Approximate Distance Oracles for Graphs with Dense Clusters

Mattias Andersson [a], Joachim Gudmundsson [b,1], Christos Levcopoulos [a]

[a] *Department of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden.*

[b] *Department of Mathematics and Computing Science, TU Eindhoven, 5600 MB, Eindhoven, the Netherlands.*

Abstract

Let $\mathcal{G}$ be a graph containing N disjoint t-spanners that are inter-connected with M edges. We present an algorithm that constructs a data structure of size $\mathcal{O}(M^2 + n \log n)$ that answers $(1+\varepsilon)$-approximate shortest path queries in $\mathcal{G}$ in constant time, where n is the number of vertices of $\mathcal{G}$.

1. Introduction

The *shortest-path* (SP) problem for weighted graphs with n vertices and m edges is a fundamental problem for which efficient solutions can now be found in any standard algorithms text, see also [6,8,12–14].

Lately the approximation version of this problem has also been studied extensively [1,5,7]. In numerous algorithms, the query version of the SP-problem frequently appears as a subroutine. In such a query, we are given two vertices and have to compute or approximate the shortest path between them. Thorup and Zwick [15] presented an algorithm for undirected weighted graphs that computes approximate solutions using a pre-computed data structure (the time of pre-processing was recently improved by Baswana and Sen [3]). Since the query time is essentially bounded by a constant, Thorup and Zwick refer to their queries as approximate *distance oracles*.

We focus on the geometric version of this problem. A geometric graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ has vertices corresponding to points in $\mathbb{R}^d$ and edge weights from a Euclidean metric, and is said to be a t-spanner for $\mathcal{V}$, if for any two points p and q in $\mathcal{V}$, there exists a path of length at most t times the Euclidean distance between p and q. Again considerable previous work exists on the shortest path and related problems for t-spanners. The geometric query version was recently studied by Gudmundsson et al. [9,10] and they presented the first data structure that answers approximate shortest-path queries in constant time, provided that the input graph is a t-spanner for some known constant $t > 1$. Their data structure uses $\mathcal{O}(n \log n)$ space and can be constructed in time $\mathcal{O}(m \log n)$.

In this paper we extend this result to hold also for "islands" of t-spanners, i.e., a set of disjoint t-spanners $\mathcal{G}_1, \ldots, \mathcal{G}_N$ inter-connected by M edges. We construct a data structure that can answer $(1+\varepsilon)$-approximate shortest path queries in constant time. The data structure uses $\mathcal{O}(M^2 + n \log n)$ space and can be constructed in time $\mathcal{O}((m + M^2) \log n)$, hence for $M = \mathcal{O}(\sqrt{n})$ the bound is essentially the same as in [9] and [10].

We claim that this generalization is natural in many applications. Consider for example the railway network in Europe where each country has a railway network which usually is a t-spanner for some small value t. The railway networks of the countries are then sparsely connected. Typically the number of inter-connecting edges is very small compared to the total number of edges in the network, see Fig. 1.

In [9] it was shown that an approximate shortest-path distance oracle can be applied to a large number of problems, for example, finding shortest obstacle-avoiding path between two vertices in a planar polygonal domain with obstacles and interesting query versions of closest pair problems. The extension presented in this paper also generalises

Email addresses: mattias@cs.lth.se (Mattias Andersson), h.j.gudmundsson@tue.nl (Joachim Gudmundsson), christos@cs.lth.se (Christos Levcopoulos).

[1] Supported by the Netherlands Organisation for Scientific Research (NWO)

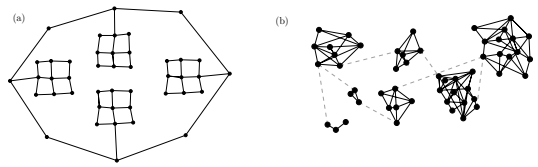


Fig. 1. (a) Many geometric networks consists of a set of "dense" graphs that are sparsely connected.(b) Example of an instance where the dashed edges are inter-connecting edges.

the results for the above mentioned problems.

We will use the following notation. For points p and q in $\mathcal{R}^d$, $|pq|$ denotes the Euclidean distance between p and q. If $\mathcal{G}$ is a geometric graph, then $\delta_{\mathcal{G}}(p,q)$ denotes the Euclidean length of a shortest path in $\mathcal{G}$ between p and q. If P is a path in $\mathcal{G}$ between p and q having length Δ with $\delta_{\mathcal{G}}(p,q) \leq \Delta \leq (1+\varepsilon)\cdot\delta_{\mathcal{G}}(p,q)$, then P is a $(1+\varepsilon)$-approximate shortest path for p and q.

The main result of this paper is stated in the following theorem:

Theorem 1 *Let $\mathcal{G}$ be a geometric graph, with n vertices and m edges, consisting of a set of disjoint t-spanners $\mathcal{G}_1=(\mathcal{V}_1,\mathcal{E}_1),\ldots,\mathcal{G}_N=(\mathcal{V}_N,\mathcal{E}_N)$ inter-connected by M edges, and let ε be a positive constant. One can construct a data structure in time $\mathcal{O}((m+M^2)\log n)$ using $\mathcal{O}(M^2+n\log n)$ space that can answer $(1+\varepsilon)$-approximate shortest path queries in constant time.*

The set of pairwise disjoint t-spanners $\mathcal{G}_1 = (\mathcal{V}_1,\mathcal{E}_1),\ldots,\mathcal{G}_N = (\mathcal{V}_N,\mathcal{E}_N)$ will be called the "islands" of $\mathcal{G}$ and, an edge $(u,v) \in \mathcal{E}$ is said to be an *inter-connecting* edge if $u \in \mathcal{V}_i$ and $v \in \mathcal{V}_j$, where $i \neq j$. A vertex $v \in \mathcal{V}_i$ incident to an inter-connecting edge is called an *harbor* and, the set of all harbors of $\mathcal{V}_i$ is denoted $\mathcal{C}_i$. Note that the total number of harbors is $\mathcal{O}(M)$ since the number of inter-connecting edges is M.

2. Tools

In the construction of the distance oracle we will need several tools, among them the well-separated pair decomposition (WSPD) by Callahan and Kosaraju [4], a graph pruning tool by Gudmundsson et al. [9] and well-separated clusters by Krznaric and Levcopoulos [11].

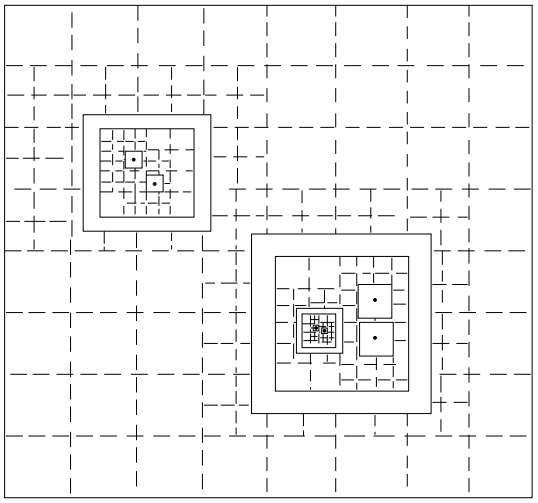


Fig. 2. An example cell partition, with respect to $\mathcal{V}'$, made by the algorithm. Doughnuts are drawn with solid lines, while inner cells are drawn with dotted ones.

In this section, given a set $\mathcal{V}$ of n points in $\mathbb{R}^d$, and a subset $\mathcal{V}' \subseteq \mathcal{V}$, we show how to associate a representative point $r \in \mathcal{V}$ to each point $p \in \mathcal{V}$, such that the distance $|pr| + |rq|$, for any point $q \in \mathcal{V}'$, is a good approximation of the distance $|pq|$. The total number of representative points is $\mathcal{O}(|V'|)$. The idea is to partition space into cells, such that all points included in a cell may share a common representative point.

We will use the fact that, given a set $\mathcal{S}$ of n points in $\mathbb{R}^d$, an approximate nearest neighbor data structure can be efficiently computed (Mount et al. [2]).

Next the algorithm for computing representative points is presented. As a pre-processing step we compute the b-cluster tree $\mathcal{T}$ ([11]) of $\mathcal{V}'$ with $b = 10/\varepsilon^2$. For a level i in $\mathcal{T}$ let $\nu(\mathcal{D}_1),\ldots,\nu(\mathcal{D}_{\ell_i})$ be the nodes at that level, where $\mathcal{D}_1,\ldots,\mathcal{D}_{\ell_i}$ are the associated clusters. Let $\mathcal{D}_{j,1}\ldots,\mathcal{D}_{j,\ell_{i+1}}$ be the cluster associated with the children of $\nu(\mathcal{D}_j)$. For each cluster $\mathcal{D}_j$ pick an arbitrary vertex d_j as the center point of $\mathcal{D}_j$. The set of the ℓ_i center points is denoted $\mathcal{D}(i)$. Perform the following four steps for each level i of $\mathcal{T}$.

(i) Compute an approximate nearest neighbor structure with $\mathcal{D}(i)$ as input, as described by Mount et al. [2].

(ii) For each center point d_j in $\mathcal{D}(i)$ compute the $(1+\varepsilon)$-approximate nearest neighbor of d_j. The point returned by the structure is denoted v_j.

(iii) For each cluster $\mathcal{D}_j$ construct two squares; $is(\mathcal{D}_j)$ and $os(\mathcal{D}_j)$ with centers at d_j and side length $2\alpha = 2(1 + 1/\varepsilon) \cdot rd(cl(\mathcal{D}_j))$ and $2\beta = 2\varepsilon|d_j, v_j|/(1 + \varepsilon)$ respectively, where $\alpha < \beta$. The two squares are called the inner and outer shells of $\mathcal{D}_j$, and the set theoretical difference between the inner and the outer shell is denoted the *doughnut* of $\mathcal{D}_j$.

(iv) The inner shell of $\mathcal{D}_j$ is recursively partitioned into four equally sized squares, until each square s either (a) is completely included in $\bigcup_{1\leq k\leq \ell_{i+1}} os(\mathcal{D}_{j,k})$ (the union of the outer shells of the children of $\nu(\mathcal{D}_j)$). In this case the square is deleted and, hence, not further partitioned. Or, (b) has diameter at most $\frac{\varepsilon}{1+\varepsilon} \cdot K$, where K is the smallest distance between a point within s and a point in $\mathcal{D}_j$. A $(1 + \varepsilon)$-approximation of K can be computed in time $\mathcal{O}(\log |\mathcal{D}_j|)$. This implies that the diameter of s is bounded by $\varepsilon \cdot K$.

The resulting cells are denoted *inner cells*. Note that, due to step 4a, every inner cell is empty of points from $\mathcal{D}_j$. An illustration of the partition is shown in Fig. 2.

Finally, after all levels of $\mathcal{T}$ has been processed, we assign a representative point to each point p in $\mathcal{V}$. Preprocess all the produced cells and perform a point-location query for each point. If p belongs to a doughnut cell then the center point of the associated cluster (see step 1) is the representative point of p. Otherwise, if p belongs to an inner cell C and p is the first point within C processed in this step then $rep(C)$ is set to p. If p is not the first point then $rep(p) = rep(C)$. Further, note that an inner cell may overlap with the union of the outer shells of the children of $v(\mathcal{D}_j)$. If a point is included in both an inner cell and an outer shell, we treat it as if it belonged to the inner cell, and assign a representative point as above.

For the above algorithm we can show the following theorem:

Theorem 2 *Given a set $\mathcal{V}$ of n points in $\mathbb{R}^d$, a subset $\mathcal{V}' \subseteq \mathcal{V}$ and a positive real value $\tau_1 < 1$, one can for each point $p \in \mathcal{V}$ associate a representative point $r(p)$ such that for any point $h \in \mathcal{V}'$, it holds that*

$$min\{|p, r(p)|, |r(p), h|\} \leq \tau_1|p, h|.$$

The number of representative points is $\mathcal{O}(|\mathcal{V}'|)$ and they can be computed in time $\mathcal{O}(n \log n)$.

3. Constructing the Oracle

In this section we consider the main result of the paper, Theorem 1. The section is divided into two subsections: first we present the construction of the structure and then how queries are answered. Note that the correctness analysis has been omitted.

3.1. *Constructing the basic structures*

In this section we show how to pre-process $\mathcal{G}$ in time $\mathcal{O}((M^2 + m) \log n)$ such that we obtain three structures that will help us answer $(1 + \varepsilon)$-approximate distance queries in constant time. We will assume that the number of edges in each subgraph is linear with respect to the number of vertices in $\mathcal{V}_i$, if not the subgraph is pruned. This is done in time $\mathcal{O}(m \log n)$ [9]. Hence, we can from now on assume that $\#\mathcal{E}_i = \mathcal{O}(\mathcal{V}_i)$.

Let $\mathcal{V}'$ be the set of vertices in $\mathcal{V}$ incident on an inter-connecting edge. Now we can apply Theorem 2 with parameters $\mathcal{V}$, $\mathcal{V}' = \Gamma'$ and τ_1 to obtain a representative point for each point in $\mathcal{V}$.

Now we are ready to present the three structures:

Oracle A: An oracle that given two points p and q answers 'yes' if p and q belongs to the same island, otherwise it will return the representative points (to be defined below) for p and q.

Oracle B: An $(1 + \varepsilon)$-approximate distance oracle for any pair of points belonging to the same island.

Matrix D: An $\mathcal{O}(M) \times \mathcal{O}(M)$ matrix. For each pair of representative points, p and q, D contains the $(1 + \varepsilon)$-approximate shortest distance between p and q.

The representative point of a point p is denoted $r(p)$, and the set of all representative points of $\mathcal{V}_i$ and $\mathcal{V}$ is denoted Γ_i and Γ, respectively. Note that $\mathcal{C}_i \subseteq \Gamma_i$. Now we turn our attention to the construction of the oracles and the matrix.

The construction of Oracles A and B are rather straightforward, with construction details omitted. However, Oracle A can be constructed in linear time, using linear space, and Oracle B can be constructed in $\mathcal{O}(m \log n)$ time, using $\mathcal{O}(n \log n)$ space.

Matrix $\mathcal{D}$ is constructed as follows. For each i, $1 \leq i \leq N$, compute the WSPD of Γ_i with separation constant $s = (\frac{1+\tau_2+\tau_3}{\tau_3-\tau_2})$ (the constants τ_2 and τ_3 are necessary for the correctness analy-

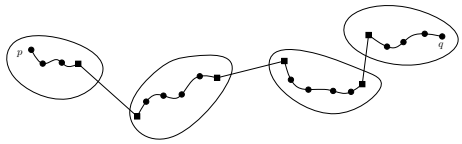


Fig. 3. Illustrating the approximate shortest path between p and q. The boxes illustrate the "harbors" along the path.

sis). As output we obtain a set of well-separated pairs $\{\{(A_1, B_1\}, \ldots, \{A_{w_i}, B_{w_i}\}\}$, such that $w_i = \mathcal{O}(\#\mathcal{C}_i)$. Next, construct the non-Euclidean graph $\mathcal{F} = (\Gamma, \mathcal{E}')$ as follows. For each Γ_i and each well-separated pair $\{A_j, B_j\}$ of the WSPD of Γ_i select two (arbitrary) representative points $a_j \in A_j$ and $b_j \in B_j$. Add the edge (a_j, b_j) to $\mathcal{E}'$ with weight $B_i(a_j, b_j)$, where $B_i(p, q)$ denotes a call to oracle B_i for $\mathcal{G}_i$ with parameters p and q. Note that the graph $\mathcal{F}$ will have $\mathcal{O}(M)$ vertices and edges.

Let D be an $\mathcal{O}(M) \times \mathcal{O}(M)$ matrix. For each representative point $p \in \Gamma$ compute the single-source shortest path in $\mathcal{F}$ to every point q in Γ and store the distance of each path in $D[p, q]$. The total time for this step is $\mathcal{O}(M^2 \log M)$, and it can be obtained by running Dijkstra's algorithm M times.

Lemma 3 *The oracles A and B, and the matrix D can be built in time $\mathcal{O}((M^2 + m) \log n)$ and the total complexity of A, B and M is $\mathcal{O}(M^2 + n \log n)$.*

3.2. *Answer a query*

Given the two oracles and the matrices presented above the query algorithm is very simple. Let $r(p)$ denote the representative point of $p \in \mathcal{V}$. Now assume that we are given two points p and q. If p and q belong to the same islands then we query Oracle B with input p, q and return the value obtained from the oracle. If p and q does not belong to the same island we return the sum of $B(p, r(p))$, $D(r(p), r(q))$ and $B(r(q), q)$. Obviously this is done in constant time.

Using Lemma 3 and analysing the correctness of the query algorithm above, we can finally show Theorem 1.

References

[1] D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999.

[2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45(6):891-923, 1998.

[3] S. Baswana and S. Sen. Approximate Distance Oracles for Unweighted Graphs in $\mathcal{O}(n^2 \log n)$ Time. In Proc. *15th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2004.

[4] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, 1995.

[5] E. Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. *SIAM Journal on Computing*, 28(1):210–236, 1998.

[6] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. In Numerische Mathmatik vol. 1, 1959.

[7] D. Dor, S. Halperin, and U. Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759, 2000.

[8] M. L. Fredman, R. E. Tarjan Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596-615, 1987.

[9] J. Gudmundsson, C. Levcopoulos, G. Narasimhan and M. Smid. Approximate Distance Oracles for Geometric graphs. In Proc. *13th ACM-SIAM Symposium on Discrete Algorithms*, 2002.

[10] J. Gudmundsson, C. Levcopoulos, G. Narasimhan and M. Smid. Approximate Distance Oracles Revisited. In Proc. *13th International Symposium on Algorithms and Computation*, 2002.

[11] D. Krznaric and C. Levcopoulos. Computing hierarchies of clusters from the Euclidean minimum spanning tree in linear time. In Proc. *15th Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, 1995.

[12] R. Raman. Recent results on the single-source shortest paths problem. SIGACT News 28:81-87,1997.

[13] M. Thorup. Floats, Integers, and Single Source Shortest Paths. *Journal of Algorithms*, 35(2):189-201, 2000.

[14] M. Thorup Undirected Single-Source Shortest Paths with Positive Integer Weights in Linear Time. *Journal of the ACM*, 46(3):362-394, 1999.

[15] M. Thorup and U. Zwick. Approximate distance oracles. In Proc. *33rd ACM Symposium on Theory of Computing*, 2001.

Geometric dilation of closed planar curves: A new lower bound

Annette Ebbers-Baumann [a], Ansgar Grüne [a], and Rolf Klein [a]
[a] *Dept. of Computer Science I, University of Bonn, D - 53117 Bonn, Germany*

Abstract

Given any simple closed curve C in the Euclidean plane, let w and D denote the minimal and the maximal caliper distances of C, correspondingly. We show that any such curve C has a geometric dilation of at least $\arcsin(\frac{w}{D}) + \sqrt{(\frac{w}{D})^2 - 1}$.

Key words: computational geometry, convex geometry, convex curves, dilation, detour, lower bound

1. Introduction

Let C be a simple closed curve C in the Euclidean plane. For any two points, p and q, on C let $\pi(p, q)$ denote the shorter of the two curve segments of C that connects p with q. Then the geometric dilation, $\delta(C)$, of C is defined as

$$\delta(C) := \sup_{p,q \in C, p \neq q} \frac{|\pi(p, q)|}{|pq|} \qquad (1)$$

The computation of the geometric dilation (then called detour) was first studied in [5], where an $O(n \log n)$ approximation algorithm for polygonal chains in the plane was given. Further efficient algorithms to compute the geometric dilation of certain classes of curves and networks were presented in [1], [11], and [9].

The question of embedding a finite point set in the plane into a network with low geometric dilation was recently studied in [4]. There it has been shown that any simple closed planar curve has dilation $\delta(C) \geq \pi/2$, using Cauchy's surface area formula.

Note, that the analogue concept on graphs, where only the point set of the vertices is taken into account for computing the dilation, was extensively studied under the notion of spanners and low dilation graphs, see e.g. [7] for a survey and [3], [2] for recent results. However, there are structural differences between the two concepts, as already mentioned e.g. in [5].

In this paper we prove a powerful generalization of the lower bound from [4]. Namely, let w and D denote the width and the diameter of the convex hull of C, correspondingly, that is, the minimal and the maximal distances of a rotating caliper measuring C; see Figure 1.

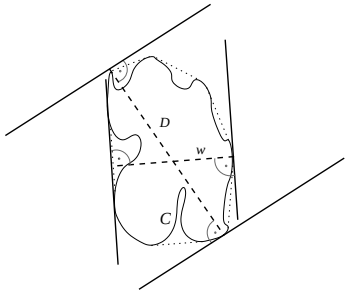


Fig. 1. Diameter D and width w of $\mathrm{ch}(C)$.

Then,

$$\delta(C) \geq \arcsin\left(\frac{w}{D}\right) + \sqrt{\left(\frac{w}{D}\right)^2 - 1} \qquad (2)$$

holds for the geometric dilation of C. This lower bound has a minimum value of $\pi/2$ if and only if $w = D$ holds. (Note, however, that the circle is not the only closed curve satisfying $w = D$.)

The proof of formula (2) uses a well-known transformation of convex curves called the central symmetrization, see e.g. [6] and [10].

The rest of this paper is organized as follows. In Section 2 we give some necessary definitions and

Email addresses: ebbers@cs.uni-bonn.de (Annette Ebbers-Baumann), gruene@cs.uni-bonn.de (Ansgar Grüne), rolf.klein@uni-bonn.de (Rolf Klein).

basic lemmata. Then, in Section 3 we cite the symmetrization transformation. Finally, Section 4 contains the proof of our lower bound.

2. Definitions and basic properties

Throughout this paper we consider *simple planar cycles* C, i.e. closed curves in the Euclidean plane without self-intersections. A simple cycle C is *convex* iff it always turns into the same direction, that is, iff C has a convex interior domain.

Definition 1 *(Dilation) Let C be a simple cycle, and let $p, q \in C$ be two points on C.*

(i) C_p^q denotes one of the two possible sub-paths of C connecting p and q, characterized by its turning direction: If one moves from p to q on C_p^q, one turns anti-clockwise ($C = C_p^q \cup C_q^p$).

(ii) The dilation of a pair of points $(p, q) \in C \times C$ *is the length of a shortest sub-path $\pi_C(p, q)$ of C connecting p and q, $|\pi_C(p, q)| = \min(|C_p^q|, |C_q^p|)$, divided by its Euclidean distance, i.e. $\delta_C(p, q) := \frac{|\pi_C(p,q)|}{|pq|}$.*

(iii) The geometric dilation of C *is the supremum of the dilation values of all pairs of points of C, i.e. $\delta(C) := \sup_{p,q \in C, p \neq q} \delta_C(p, q)$.*

By continuity and compactness arguments one can show that every finite convex curve has a pair of points attaining maximum dilation.

Definition 2 *(Partition Pair) Let $p \in C$ be a point on a cycle C. Then the unique* partition partner $\hat{p}$ *of p is characterized by $|\pi_C(p, \hat{p})| = |C|/2$. We say that $(p, \hat{p})$ is a* partition pair *of C.*

By continuity arguments it is easy to show that for every direction $v \in \mathbb{S}^1$ there exists a partition pair $(p, \hat{p})$, i.e. $\hat{p} - p = |\hat{p} - p|\, v$.

Definition 3 *(Breadths) Let C be a simple cycle, and let $v \in \mathbb{S}^1$ be an arbitrary direction.*

(i) The v-length *of C is the maximum distance of a pair of points with direction v, i.e. $l_C(v) := \max\{|pq| \mid p, q \in C, q - p = |q - p|\, v\}$.*

(ii) The v-width (v-breadth) *of C is the distance of the two supporting lines of C perpendicular to v, i.e. $w_C(v) := \max_{p \in C} p \cdot v - \min_{p \in C} p \cdot v$.*

(iii) The v-partition pair distance, *$h_C(v)$, of C is the distance of the partition pair with direction v.*

(iv) The diameter, *$D(C)$, of C is the maximal v-length, i.e. $D(C) := \max_{v \in \mathbb{S}^1} l_C(v)$. The* width, *$w(C)$, of C is the minimal v-length, i.e. $w(C) := \min_{v \in \mathbb{S}^1} l_C(v)$.*

(v) The maximal partition pair distance *is denoted by $H(C) := \max_{v \in \mathbb{S}^1} h_C(v)$ and the* minimal partition pair distance *by $h(C) := \min_{v \in \mathbb{S}^1} h_C(v)$.*

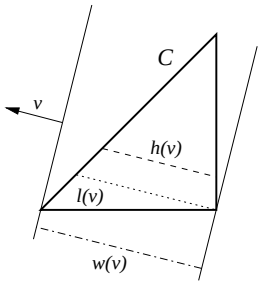


Fig. 2. Three different breadth measures.

As used in the introduction, width and diameter can also be defined using the v-width values which is proved in [8], [12] respectively:

Lemma 4 *Let C be a simple cycle, then $D(C) = \max_{v \in \mathbb{S}^1} w_C(v)$. If C is convex, then $w(C) = \min_{v \in \mathbb{S}^1} w_C(v)$.*

The next statement follows immediately from the definitions; see Figure 2.

Lemma 5 *Let C be a simple convex cycle, and let $v \in \mathbb{S}^1$ be an arbitrary direction. Then the following inequalities hold: $h_C(v) \leq l_C(v) \leq w_C(v)$.*

3. Central symmetrization

The central symmetrization (see e.g. [6], [8], [10]) is a well-known transformation which maps any convex cycle to a convex point-symmetric cycle. In our notion, the central symmetrization is based on the length values $l_C(v)$, introduced in Definition 3(i).

Amazingly, it preserves all the width values $w_C(v)$. And Cauchy's surface area formula implies that the perimeter is not changed either.

Definition 6 *Let C be a convex cycle. The* central symmetrization *of C is the cycle C' given by the parametrization $c' : \mathbb{S}^1 \to \mathbb{R}^2$, $c'(v) := \frac{l_C(v)}{2} v$.*

As depicted in Figure 3, we can construct the central symmetrization by translating all the centers of the segments of maximal length connecting pairs of points on C to the origin.

However, there is an easier and more helpful construction, described in the following lemma.

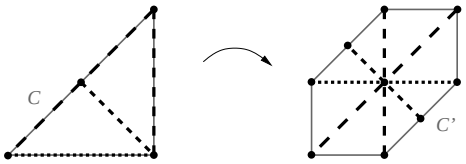


Fig. 3. The central symmetrization of an isosceles right-angled triangle.

Lemma 7 *Let $X := f(C) \cup C$ be the face bounded by C including C itself. Then define a set X' to be the* arithmetic mean *of X and $-X$; see [10]. It is the* Minkowski sum $X \oplus -X$ *scaled by* $1/2$. *Then, the central symmetrization C' is the boundary of this arithmetic mean X':*

$$X := f(C) \cup C$$
$$X' := \frac{1}{2}(X \oplus -X)) = \left\{ \frac{1}{2}(u - v) \,\middle|\, u, v \in X \right\}$$
$$\Rightarrow \quad C' = \partial X'$$

PROOF. The proof that this second way of constructing C' is also correct is straightforward. Let $v \in \mathbb{S}^1$ be an arbitrary direction. Define $l := \sup\{c \in \mathbb{R}^{>0} \mid cv \in X'\}$. Then due to X' being closed, lv is an element of X'. And for $k > l$ the point kv is not in X'. Thus, $lv \in \partial X'$.

It also follows that there are $p, q \in X$ such that $lv = (1/2)(q-p)$. On the other hand, the definition of l yields that $k > l$ implies there are no $p, q \in X$ satisfying $kv = (1/2)(q-p)$. Thus, $l = (1/2)l_C(v)$.

Hence, our analysis results in the following parametrization of $\partial X'$: $c'(v) = (1/2)l_C(v)v$. And this is exactly the parametrization we used to define C'.

The following lemma, stated here without proof, lists the most important properties of the central symmetrization. The fact that the width values are preserved is mentioned without proof in [6]. Gritzmann and Klee [8] prove the width-preserving and length-preserving property. The statement that the perimeter is preserved is also proved in [10].

Lemma 8 *Let C be a simple convex cycle, and let C' be its central symmetrization. Then, the cycle C' has the following properties:*

(i) Cycle C' is convex.
(ii) Cycle C' is point-symmetric with respect to the origin.
(iii) For every direction $v \in \mathbb{S}^1$, $h_{C'}(v) = l_{C'}(v) = l_C(v) \geq h_C(v)$, and $w_{C'}(v) = w_C(v)$.
(iv) Width, diameter and perimeter are preserved by central symmetrization, i.e. $w(C') = w(C)$, $D(C') = D(C)$ and $|C'| = |C|$.

Because we can show that the dilation of a convex cycle is always attained by a partition pair, it follows easily from those properties that the dilation of the transformed cycle cannot be larger then the original one:

Lemma 9 *The dilation of C' is not larger than the original dilation, i.e. $\delta(C') \leq \delta(C)$.*

4. The lower bound

To apply the transformation described in Section 3, we need the fact that the dilation of the boundary of the convex hull of any planar cycle C is at most the dilation of C itself. Due to space limitations we state this here without proof.

Theorem 10 *Let $C \subset \mathbb{R}^2$ be a simple closed curve. Let $\partial\mathrm{ch}(C)$ denote the boundary of the convex hull of C. Then holds*

$$\delta(C) \geq \delta(\partial\mathrm{ch}(C)).$$

Now we prove our result on the lower bound, using the central symmetrization transformation.

Theorem 11 *Let $C \subset \mathbb{R}^2$ be a simple closed curve. Let w be the width and let D be the diameter of* $\mathrm{ch}(C)$, *the convex hull of C. Then the dilation of C is bounded from below by*

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

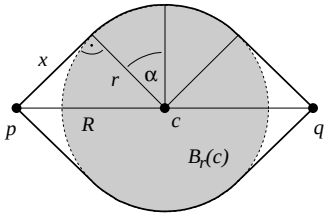


Fig. 4. The shortest cycle not intersecting the disk $B_r(c)$.

PROOF. Because of Theorem 10 we can assume w.l.o.g. that C is convex. To show the main idea of the proof we first consider a point-symmetric cycle $\tilde{C} \subset \mathbb{R}^2$ with center-point c, see Figure 4. Then, obviously, the partition pairs are also point-symmetric with respect to c. Let (p, q) be a parti-

tion pair having maximum distance $|pq| = H(\tilde{C})$. We define $R := H(\tilde{C})/2$.

Let $B_r(c)$ be the open disc with center point c and radius $r := h(\tilde{C})/2$, that is $B_r(c) := \{b \in \mathbb{R}^2 | \, |b-c| < r\}$. Then $\tilde{C}$ cannot intersect with $B_r(c)$, otherwise there would exist a partition pair having a distance smaller than $h(\tilde{C}) = 2r$.

By using the shortest possible cyclic path connecting p and q in the Euclidean plane, not intersecting $B_r(c)$ but enclosing it, we obtain a curve $\tilde{C}_{minPer}$, as shown in Figure 4. By construction $\tilde{C}_{minPer}$ is the point-symmetric curve of smallest perimeter that has minimum partition pair distance $h(\tilde{C})$ and maximum partition pair distance $H(\tilde{C})$. Due to symmetry reasons this perimeter is $P = 4x + 4r\alpha$, with x denoting the lengths of the straight path segments from p and q to the tangent points on $B_r(c)$, correspondingly, and $r\alpha$ the lengths of the path segments on $\partial B_r(c)$.

Using Pythagoras we get $x = \sqrt{R^2 - r^2}$. And by considering the angles in the rectangular triangle, we obtain $\sin\alpha = \cos(\frac{\pi}{2} - \alpha) = \frac{r}{R}$. Because the maximum dilation of the convex cycle $\tilde{C}$ is attained by a partition pair having shortest path distance $\frac{\tilde{P}}{2} \geq \frac{P}{2}$, it holds:

$$\delta(\tilde{C}) \geq \frac{\frac{P}{2}}{2r} = \frac{4x + 4r\alpha}{4r} = \frac{\sqrt{R^2 - r^2}}{r} + \arcsin\left(\frac{r}{R}\right)$$
$$= \sqrt{\left(\frac{R}{r}\right)^2 - 1} + \arcsin\left(\frac{r}{R}\right) \qquad (3)$$

Now, let C be an arbitrary convex cycle, and let C' be its central symmetrization. Then, Lemma 9 yields that $\delta(C) \geq \delta(C')$. And by Lemma 8(iv) we know that width and diameter are preserved, i.e. $w(C') = w(C)$ and $D(C') = D(C)$. However, in a point-symmetric convex cycle v-length and v-partition pair distance are equal, implying $w(C') = h(C')$ and $D(C') = H(C')$.

Thus, if we apply formula (3) to C' keeping in mind that $r = h(C')/2 = w(C')/2 = w(C)/2 = w/2$ and $R = H(C')/2 = D(C')/2 = D(C)/2 = D/2$, we get:

$$\delta(C) \geq \sqrt{\left(\frac{D}{w}\right)^2 - 1} + \arcsin\left(\frac{w}{D}\right)$$

This lower bound equals the global lower bound of $\frac{\pi}{2}$ shown in [4] only for curves of constant width ($w = D$). Further arguments show that the inequality gets strict if C is not point-symmetric or not convex. Hence, only circles have dilation $\frac{\pi}{2}$.

Remark 12 *By replacing $l_C(v)$ by $h_C(v)$ in Definition 3 we get a new transformation, the* partition pair transformation. *Ideas analogous to the ones presented here show that*

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

where $H = H(\mathrm{ch}(C))$ and $h = h(\mathrm{ch}(C))$ for every simple cycle C.

References

[1] P. Agarwal, R. Klein, Ch. Knauer, and M. Sharir. Computing the detour of polygonal curves. Tech. Report B 02-03, FU Berlin, 2002.

[2] P. Bose, J. Gudmundsson, and M. Smid. Constructing plane spanners of bounded degree and low weight. In *Proc. 10th Europ. Symp. Algo.*, LNCS 2461:234–246, 2002.

[3] D.Z. Chen, G. Das, and M. Smid. Lower bounds for computing geometric spanners and approximate shortest paths. *Discr. Appl. Math.*, 110:151–167, 2001.

[4] A. Ebbers-Baumann, A. Grüne, and R. Klein. On the geometric dilation of finite point sets. In *Proc. 14th Internat. Symp. Algo. and Comput.*, LNCS 2906:250–259, 2003.

[5] A. Ebbers-Baumann, R. Klein, E. Langetepe, and A. Lingas. A fast algorithm for approximating the detour of a polygonal chain. In *Proc. 9th Europ. Symp. Algo.*, LNCS 2161:321–332, 2001.

[6] H.G. Eggleston. *Convexity*, Cambridge University Press, 1958.

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

[8] P. Gritzmann and V. Klee. Inner and outer j-radii of convex bodies in finite-dimensional normed spaces. *Discrete Comput. Geom.*, 7:255–280, 1992.

[9] Ansgar Grüne, Rolf Klein, and Elmar Langetepe. Computing the detour of polygons. In *Abstracts 19th European Workshop Comput. Geom.*, pages 61–64. University of Bonn, 2003.

[10] I.M. Jaglom and W.G. Boltjanski. *Konvexe Figuren*, VEB Deutscher Verlag der Wissenschaften, 1956.

[11] S. Langerman, P. Morin, and M. Soss. Computing the maximum detour and spanning ratio of planar chains, trees and cycles. In *Proc. 19th Internat. Symp. Theor. Aspects of C.Sc.*, LNCS 2285:250–261, 2002.

[12] S.R. Lay. *Convex Sets and their Applications*, John Wiley & Sons, 1982.

Finding a door along a wall with an error afflicted robot

Tom Kamphans [a] and Elmar Langetepe [a]

[a] *University of Bonn*
Institute of Computer Science I
Römerstraße 164
53117 Bonn
Germany
http://web.informatik.uni-bonn.de/I/agklein.html

Abstract

We consider the problem of finding a door in a wall with a blind robot, that does not know the distance to the door or whether the door is located left hand or right hand to its start point. This problem can be solved with the well-known doubling strategy yielding an optimal competitive factor of 9 with the assumption, that the robot does not make any errors during its movements. We study the case, that the robots movement is errorneous. We give upper bounds for the movement error, such that reaching the door is guaranteed. More precisely the error range δ has to be smaller than $\frac{1}{3}$. Additionally, the corresponding competitive factor is given by $1 + 8\frac{1+\delta}{1-3\delta}$.

Keywords: Online algorithm, Online motion planning, searching, competitive ratio, errors.

1. Introduction

Online motion planning in unknown environments is theoretically well-understood and practically solved in many settings. During the last decade many different objectives where discussed under several robot models. For a general overview of theoretical online motion planning problems and its analysis see the surveys [3,9,10,7].

Theoretical correctness results and performance guarantees often suffer from idealistic assumptions, therefore in the worst case a correct implementation is impossible. On the other hand practioners analyze correctness results and performance guarantees mainly statistically. Therefore it is useful to investigate, how online algorithms with idealistic assumptions behave, if those assumptions cannot be fulfilled. More precisely, can we incorporate assumptions of errors in sensors and motion directly into the theoretical analysis? We already successfully considered the behaviour of the well-known pledge algorithm, see Abelson and diSessa [1] and Hemmerling [6], in the presence of errors [8].

The task of finding a point on a line without knowing the direction or the distance to the start point was considered by Baeza-Yates et. al. [2] and independently by Gal [5] and lead to the so called doubling strategy, which gives a basic paradigma for other searching algorithms, i.e., searching for a point on m rays or approximating the optimal search path, see [4].

Under the competitive framework doubling with a factor of two is the optimal strategy for searching a point on the line. The competitive analysis compares the cost of the strategy with the cost of a solution which is computed under full information, see [2].

In this paper we investigate how the doubling strategy behaves in the presence of errors and how the error influences the correctness and the corresponding competitive factor of the strategy.

We assume that an error range for a single step is known in advance but the agent gets no further information about the cummulated error. Therefore for the agent it makes no sense not to use the doubling heuristic.

Email addresses: kamphans@cs.uni-bonn.de (Tom Kamphans), langetep@cs.uni-bonn.de (Elmar Langetepe).

2. The lost cow problem

The task is to find a door in a wall, respectively a point, t, on a line. The robot does not know whether t is located left hand or right hand to its start position, s, nor does it know the distance from s to t. Baeza-Yates et. al. [2] describe the strategy to solve this problem using a function f. $f(i)$ is the distance the robot walks in the ith step. If i is odd, the robot moves $f(i)$ steps from the start to the right and $f(i)$ steps back; if i is even, the robot moves to the left. It is assumed, that the movement is correct, so after moving $f(i)$ steps from the start point to the right and moving $f(i)$ steps to the left, the robot has reached its start point. Baeza-Yates et. al. showed, that a strategy with $f(i) = 2^i$ is 9-competitive and this is optimal.

An online strategy that produces a path of length $|\pi_{\text{onl}}|$ is called C-*competitive* if for all scenes

$$|\pi_{\text{onl}}| \leq C \cdot |\pi_{\text{opt}}| + A$$

holds, where $|\pi_{\text{opt}}|$ denotes the path length of the optimal strategy which makes use of full information.

3. Modelling the error

The robot moves straight line segments of a certain length from the start point alternately to the left and to the right. Every movement can be afflicted with an error, that causes the robot to move more or less far than expected. However, we require the robots error in every step is within a certain error bound, δ. More precisely if the strategy moves the robot a distance ℓ we require that the covered distance is in the range $[\ell \cdot (1-\delta), \ell \cdot (1+\delta)]$ with $\delta \in [0, 1[$.

4. Reaching the door

How large can the error bound δ become under the restiction, that the robot should be able to reach the door? W. l. o. g. we consider the case, that the door is located on the same side as the first step of the agent. We assume that the door is located at $d = 2^{2j} - \varepsilon$, so an error-free robot hits the door during the iteration with step width $f(j) = 2^{2j}$.

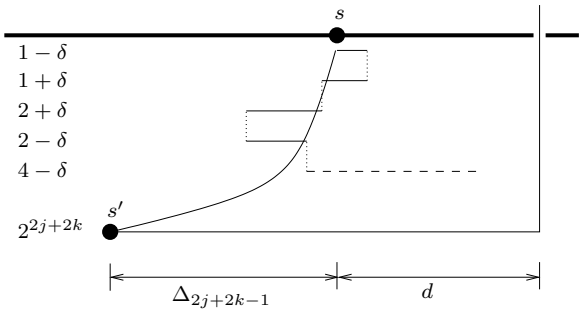


Fig. 1. In the worst case, the start point of every iteration drifts away from the door. The vertical path segments are to highlight the single iterations, the robot moves on horizontal segments only.

In the worst case, every step to the right of the error afflicted robot is too short and every step to the left is too long, so start point of the iterations drifts to the left, see Fig. 1. Let Δ_k denote the drift after k iterations.

$$\begin{aligned}\Delta_k = \sum_{i=0}^{k} \Big(& \underbrace{2^i \cdot (1+\delta)}_{\text{Steps, that are too long}} \\ & - \underbrace{2^i \cdot (1-\delta)}_{\text{Steps, that are too short}} \Big) \\ = 2\delta \cdot \sum_{i=0}^{k} 2^i &= 2\delta(2^{k+1} - 1).\end{aligned}$$

Let the error afflicted robot miss the door during the iteration with step width $f(j) = 2^{2j}$ due to the drift to the left. Now we are interested in the number of additional steps, and whether the robot is able to reach the door at all. Obviously the reachability and the number of additional steps depends on the error δ. If the robot hits the door, it will hit in the iteration with a step with of 2^{2j+2k}, $k \in \mathbb{N}^{>0}$, because the door is located right hand to the start point. To ensure, that the robot hits the door, the length of the last straight path must be at least as large as the distance to door plus the overall drift to the left. The last straight path may be error afflicted again, but its length is a least the lower bound of the error range in the iteration 2^{2j+2k}. This yields

$$
\begin{aligned}
&\Delta_{2j+2k-1} + d \le 2^{2j+2k} \cdot (1-\delta) \\
\Leftrightarrow\ & 2\delta \cdot 2^{2j+2k} - 2\delta + 2^{2j} - \varepsilon \le 2^{2j+2k} - 2^{2j+2k}\delta \\
\Rightarrow\ & 2^{2j}\left[2\delta 2^{2k} + 1 - 2^{2k} + \delta 2^{2k}\right] - 2\delta \\
& < 2^{2j}\left[3\delta 2^{2k} + 1 - 2^{2k}\right] \le \varepsilon \\
\Leftrightarrow\ & \delta \le \frac{1}{3}\frac{2^{2k}-1}{2^{2k}} + \frac{\varepsilon}{3 \cdot 2^{2j+2k}} \\
\Rightarrow\ & \delta \le \frac{1}{3}\frac{2^{2k}-1}{2^{2k}}
\end{aligned}
$$

The last term converges for $k \to \infty$ to $\frac{1}{3}$. Thus

Corollary 1 *If the error δ is not greater than $\frac{1}{3}$ the robot will hit the door.*

With the given analysis it is also possible to present relations between the error range and the number of additional iterations. For example:

Corollary 2 *If the error δ is not greater than $\frac{1}{4}$ the robot will hit the door with only one iteration step more than the error-free robot.*

5. Analyzing the performance

W. l. o. g. for the competitive setting it would be the worst, if the door is hit during the iteration with step width 2^{2j+2}, but located just a litte bit further away than the rightmost point that was reached during the iteration with step width 2^{2j}, i. e.

$$d = 2^{2j}(1-\delta) - \Delta_{2j-1} + \varepsilon = 2^{2j}(1-3\delta) + 2\delta + \varepsilon.$$

This yields a factor of

$$
\begin{aligned}
\frac{|\pi_{\text{onl}}|}{d} &= \frac{1}{d}\left(2\sum_{i=1}^{2j+1} 2^i + \Delta_{2j+1} + d\right) \\
&= \frac{1}{d}\left(2 \cdot (2^{2j+2}-1) + 2\delta(2^{2j+2}-1) + d\right) \\
&= \frac{2^{2j}\left(8(1+\delta) - \frac{1+\delta+\varepsilon}{2^{2j}}\right)}{2^{2j}\left(1-3\delta+\frac{2\delta+\varepsilon}{2^{2j}}\right)} + 1 \\
&< 8\frac{1+\delta}{1-3\delta} + 1
\end{aligned}
$$

To show that this is the worst case, we show that the robots errors in the left and in the right direction are different. Furthermore the error in each step may be a different one. Let δ_i^+ be length of the movement to the right in the i-th step and δ_i^- be the length of the movement to the left. Now

$$\sum_{i=1}^{2j-1}(\delta_i^- - \delta_i^+)$$

denotes the deviation from the start point before the $2j$-th step and

$$\sum_{i=1}^{2j-1}(\delta_i^- + \delta_i^+)$$

is the path length up to this point.

As above the door is hit during the step $2j+2$, therefore its distance is

$$
\begin{aligned}
d &= \delta_{2j}^+ - \Delta_{2j-1} + \varepsilon \\
&= \delta_{2j}^+ - \sum_{i=1}^{2j-1}(\delta_i^- - \delta_i^+) + \varepsilon
\end{aligned}
$$

For the corresponding path length we have

$$|\pi_{\text{onl}}| = \sum_{i=1}^{2j+1}(\delta_i^- + \delta_i^+) + \sum_{i=1}^{2j+1}(\delta_i^- - \delta_i^+) + d$$

due to the fact that we will finally stop at distance d, we add the deviation to d.

With this we get

$$\frac{|\pi_{\text{onl}}|}{d} = 1 + \frac{\sum_{i=1}^{2j+1}(2\delta_i^-)}{\delta_{2j}^+ - \sum_{i=1}^{2j-1}(\delta_i^- - \delta_i^+) + \varepsilon}$$

We can conclude that the ratio achieves its maximum if we exceed every δ_i^- to the greatest extend, which is $2^i(1+\delta)$. Now we only have to fix δ_i^+ in order to maximize the ratio. Obviously the nominator gets its smallest value if every δ_i^+ is very small, therefore we use $\delta_i^+ = 2^i(1-\delta)$.

Theorem 3 *If the error δ is not greater than $\frac{1}{3}$ the robot will hit the door with the doubling strategy. The generated path is not longer than $8\frac{1+\delta}{1-3\delta} + 1$ times the shortest path to the door.*

6. Summary

We have analyzed a simple doubling strategy to find a door in a wall respectively a point on a line in the presence of errors in movements. We showed that the robot is still able to reach the door if the error is less than $\frac{1}{3}$. Moreover, if the error is less than $\frac{1}{4}$ the robot will need at most two iteration steps more than the error-free robot. If the error in

the movement is bound by $\delta < \frac{1}{3}$, the competitive ratio is

$$8\frac{1+\delta}{1-3\delta} + 1.$$

Both error bounds are rather big, so it can be expected, that real robots will meet this error bounds.

The problem for m rays is a little bit different because the robot may detect the starting point if it changes from one corridor to another. Here, one may be interested in the probability of entering correctly the next corridor.

References

[1] H. Abelson and A. A. diSessa. *Turtle Geometry*. MIT Press, Cambridge, 1980.

[2] R. Baeza-Yates, J. Culberson, and G. Rawlins. Searching in the plane. *Inform. Comput.*, 106:234–252, 1993.

[3] P. Berman. On-line searching and navigation. In A. Fiat and G. Woeginger, editors, *Competitive Analysis of Algorithms*. Springer-Verlag, 1998.

[4] R. Fleischer, T. Kamphans, R. Klein, E. Langetepe, and G. Trippen. Competitive online approximation of the optimal search ratio. 2004. to appear.

[5] S. Gal. Continous search games. In *Search theory: some recent developments.*, volume 112 of *Lecture Notes in pure and applied mathematics*, pages 33–53. Dekker, New York, NY, 1989.

[6] A. Hemmerling. *Labyrinth Problems: Labyrinth-Searching Abilities of Automata*. B. G. Teubner, Leipzig, 1989.

[7] C. Icking, T. Kamphans, R. Klein, and E. Langetepe. On the competitive complexity of navigation tasks. In H. B. Hendrik I. Christensen, Gregory D. Hager and R. Klein, editors, *Sensor Based Intelligent Robots*, volume 2238 of *LNCS*, pages 245–258, Berlin, 2002. Springer.

[8] T. Kamphans and E. Langetepe. The pledge algorithm reconsidered under errors in sensors and motion. In *Proc. of the 1th Workshop on Approximation and Online Algorithms*, Lecture Notes Comput. Sci., Berlin, 2003. Springer. to appear.

[9] J. S. B. Mitchell. Geometric shortest paths and network optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 633–701. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[10] N. S. V. Rao, S. Kareti, W. Shi, and S. S. Iyengar. Robot navigation in unknown terrains: introductory survey of non-heuristic algorithms. Technical Report ORNL/TM-12410, Oak Ridge National Laboratory, 1993.

Unfolding Simple Chains Inside Circles

Marzieh Eskandari [a], Ali Mohades [a]

[a] *Math. and Computer Science Dpt., AmirKabir University of Technology, P.O.BOX 15875-4413, Tehran, Iran.*

Abstract

It is an open problem to determined whether a polygonal chain can be straightened inside a confining region if its links are not allowed to cross. In this paper we propose a special case: whether a polygonal chain can be straightened inside a circle without allowing its links to cross. We prove that this is possible if the straightened configuration can fit within circle. Then we show that these simple chains have just one equivalence class of configurations.

Key words: folding, polygonal chain, reconfiguration

1. Introduction

A *chain* is a sequence of rigid rods or links consecutively connected at their endpoints, about which they may rotate freely. The link between A_{i-1} and A_i $(1 \leq i \leq n)$ is denoted by L_i and the length of L_i is denoted by l_i. The angle at intermediate joint A_i, $\theta_i \in [0, 2\pi)$, is that determined by rotating L_i about A_i counterclockwise to bring L_i to L_{i+1}. The chain Γ is *simple* if it is non-self-crossing and non-self-touching. The subchain of Γ with joints $A_i, ..., A_j$ is denoted by $\Gamma[i, j]$.

We say a *bend operation* is performed at joint A_i, when the joint angle θ_i is changed between θ_i and π. Throughout this paper, we assume that the only bend operations allowed are *single-joint* bend operations, in which only one joint angle is altered at a time. A bend operation is *complete* if, at the end of the operation the joint angle is π. We then say that the joint has been *straightened*. A bend Operation that is not complete is called a *partial bend*. A sequence of bend operations is said to be *monotonic* if no operation increases the absolute deviation from straightness, $|\theta_i - \pi|$, for a joint A_i.

Let $\sigma = (i_1, i_2, ..., i_{n-1})$ be a permutation of the indices $\{1, 2, ..., n-1\}$. For a simple chain Γ, we say that a sequence $(A_{i_1}, A_{i_2}, ..., A_{i_{n-1}})$ of joints is *unfoldable*, if Γ can be straightened into a straight line segment L using the joints in the sequence in turn, such that Γ remains simple and all of the bend operations are complete. A simple chain Γ is called *unfoldable chain*, if it has a unfoldable sequence of joints. An intermediate joint A_i is called *unfoldable joint*, if a complete bend operation can be performed at A_i such that during the performing bend operation, Γ remains simple.

The *union chain*, Γ_U, of a chain Γ is a chain which is obtained from Γ in the following way: if none of the joints of Γ is straight joint, $\Gamma_U = \Gamma$; if Γ has at least one straight joint, for any straight joints A_i, we delete joint A_i and put $A_{i-1}A_{i+1}$ as a single link.

Reconfiguration problem and in particular, folding problem, been raised independently by several researchers. [3] has considered reconfiguration of robot arms inside a circle, with allowing its links to cross. In [4], Pei has proved that for a chain Γ inside a circle whose radius is sufficiently big, there is just one equivalence class when its links are allowed to cross. In [5] and [2], straightening a simple chain in the plane is studied and is proved that any simple chain can be straightened in the plane. And in [1] Arkin, Fekete and Mitchell have given an efficient algorithm to determined if a simple chain can be straightened by performing complete bend operations. In this paper, we study straightening a simple chain within a circle. we give a quadratic-time algorithm to straighten a simple unfoldable chain within a circle whose radius is sufficiently big. Then we prove that all of simple chains can

Email address: mohades@aut.ac.ir (Ali Mohades).

be straightened within a circle, if and only if their straightened configuration can fit in the circle. Finally we show that any two configuration of these simple chains are equivalent.

2. Preliminaries

Let Γ be a simple chain inside circle $C(O,r)$ with joints $A_0...A_n$. For fitting straightened configuration of Γ in C, we must have $\sum_{i=1}^{n} l_i \leq 2r$. From now on, we suppose Γ is a simple chain inside C such that $\sum_{i=1}^{n} l_i < 2r$.

For a circle $C(O,r)$ and two points $x, y \in \partial C$, we use $\overset{\frown}{xy}$ to denote the clockwise arc from x to y. For a point $A_i \in \partial C$ we denote the other endpoint of the diameter of C which is containing A_i, by M_i.

Definition 1 *A joint A_{r_i} is called* rim joint *if it lies on boundary of circle C. We denote the set of all rim joints of chain Γ by $A_{Rim} = \{A_{r_0}, A_{r_1}, ..., A_{r_s}\}$.*

Definition 2 *For any rim joint A_i of chain Γ, the vector $\overrightarrow{OM_i}$ is called* radius vector *of A_i and is denoted by $\mathbf{r_i}$.*

Lemma 3 *There is a diameter $s = ab$ of circle C such that all of rim joints of chain Γ belong to one of the arcs $\overset{\frown}{ab}$ or $\overset{\frown}{ba}$.*

PROOF. If $A_{Rim} = \emptyset$, there is nothing to prove. Let A_i be a rim joint of Γ and X be a moving object which is walking along arc $\overset{\frown}{A_iM_i}$ clockwise, starting at the point A_i. Suppose A_r is the last rim joint of Γ that is visited by X. Diameter $s = A_rM_r$ is a solution. Because arc $\overset{\frown}{A_rM_r}$ contains no rim joint of Γ, except A_r. □

Definition 4 *Suppose A_{Rim} has at least two point and rim joints of Γ belong to $\overset{\frown}{ab}$. The nearest rim joints to points a and b are denoted by A_f and A_e, respectively. These joints are called* limit-joints.

It is clear that all of the other rim joints of Γ are on arc $\overset{\frown}{A_fA_e}$.

Definition 5 *Let A_e and A_f be limit-joints of Γ. Vectors $\mathbf{r_e}$ and $\mathbf{r_f}$ are called* direction vectors.

Definition 6 *The sum of direction vectors, $\mathbf{r_e}$ and $\mathbf{r_f}$, is called* central direction *and denoted by $\mathbf{d}$, i.e., $\mathbf{d} = \mathbf{r_e} + \mathbf{r_f}$.*

Central Translation: We draw n vectors parallel to $\mathbf{d}$ from any joint A_i until hit circle at points N_i, then put $\varepsilon_i = ||\overrightarrow{A_iN_i}||$ and $\varepsilon = \min\{\varepsilon_i \mid 0 \leq i \leq n\}$. Translation of Γ inside C along the vector $\mathbf{d}_\varepsilon = \frac{\varepsilon}{||\mathbf{d}||}\,\mathbf{d}$, is called *central translation* of Γ. New positions of Γ and any joint A_i, after the central translation, are denoted by Γ' and A_i'. It is clear that $\varepsilon_e = \varepsilon_f$.

3. Unfoldable Simple Chains

Suppose $\sigma = (A_{i_1}, A_{i_2}, ..., A_{i_{n-1}})$ is an unfoldable sequence of joints of Γ. For straightening Γ inside circle $C(O,r)$, we propose the following algorithm which contains three steps:

Algorithm 1 *Unfolding Simple Chain Γ:*

step 1. $\Gamma' = \Gamma$; $j = 0$.
step 2. $\Gamma' = \Gamma'_U$; $j = j + 1$. *If $j = n$, stop. else* $k = i_j$;
step 3. *Straighten A_k. Go to step 2.*

Now for step 3, straightening joint A_k within circle C, we propose the following algorithm which contains four steps:

Algorithm 2 *Straightening joint A_k:*

step 1. $\Gamma_0 = \Gamma[0,k]$; $\Gamma_n = \Gamma[k,n]$;
step 2. *Rotate Γ_0 about A_k until A_k straightens or one of joints of Γ_0 hits C and Γ_0 can not rotate more about A_k. If A_k straightens, stop; else go to step 3.*
step 3. *Rotate Γ_n about A_k until A_k straightens or one of joints of Γ_n hits C and Γ_n can not rotate more about A_k. If A_k straightens, stop; else go to step 4.*
step 4. *Calculate $\mathbf{d}_\varepsilon$ and transmit Γ by $\mathbf{d}_\varepsilon$. Go to step 2.*

4. Correctness of Algorithm 2

Any repeat of algorithm 2 is called a *phase* and the joint angle at A_k, at the end of phase i, is denoted by α_i. For showing correctness of algorithm 2, we show that during the algorithm, Γ remains simple and it remains inside C. And we prove that by using algorithm 2, after a finite number of repeats, A_k straightens. Furthermore, this finite number is independent of n.

Chain Γ remains simple, because A_k is an unfold-

able joint in the plane. Now for showing that Γ remains inside C, we first prove that central translation always can be done.

Lemma 7 $\mathbf{d}_\varepsilon \neq \mathbf{0}$.

PROOF. If $\mathbf{d} = \mathbf{0}$, we have $\mathbf{r_f} = -\mathbf{r_e}$. That is implies $A_f = M_e$ and $A_e = M_f$, i.e., $A_fA_e = 2r$. Therefore $\sum_{i=1}^{n} l_i \geq \sum_{l_i \in \Gamma[e,f]} l_i \geq A_fA_e = 2r$. That is a contradiction. Thus $\mathbf{d} \neq \mathbf{0}$.

Because the angles between $\mathbf{d}$ and its components are less than $\pi/2$ and all of radius vectors lie between vectors $\mathbf{r_e}$ and $\mathbf{r_f}$, the angle between $\mathbf{d}$ and radius vectors are less than $\pi/2$, too. Thus any rim joint can transmit in direction $\mathbf{d}$ inside C. Any interior points of C also can transmit in all directions inside C. Therefore $\varepsilon \neq 0$. Consequently $\mathbf{d}_\varepsilon \neq \mathbf{0}$. □

It is clear that during the step 1 and step 2, all of the joints remain inside circle. At step 3, because $\varepsilon = \min\{\varepsilon_i \mid 0 \leq i \leq n\}$ and the angle between radius vectors and $\mathbf{d}$ are less than $\pi/2$, Γ remains inside C.

Now to show that after a finite number of repeats, algorithm 2 is terminated, we first show that at the end of any phase of algorithm 2, α_i becomes strongly close to π, i.e., $|\pi - \alpha_{i+1}| < |\pi - \alpha_i|$. So we have to prove at the end of central translation of Γ, at least one of the subchains Γ_0 or Γ_n, can rotate about A_k such that joint angle at A_k has became close to π. Note that at the end of step 1 and step 2, if A_k does not straighten, A_{Rim} has at least two points, one point from Γ_0 and the other point from Γ_n.

Lemma 8 *Let A_e and A_f be the limit-joints of Γ at phase i. If both of A_e and A_f belong to one of the subchains Γ_0 or Γ_n, then at the end of translation, none of the joints of the other subchain lie on ∂C. Furthermore, this subchain can rotate about A_k at phase $i+1$.*

PROOF. Assume without loss of generality that $A_f, A_e \in \Gamma_0$. Suppose for a contradiction, A_t is a joint of Γ_n such that $A'_t \in \partial C$. At the beginning of translation, Γ_n has at least one rim joint, A_m, which lies on arc $\overset{\frown}{A_fA_e}$. If A_k is in the exterior of closed curve $\delta = \overset{\frown}{A_fA_e} \cup \Gamma_0[e,f]$, $\Gamma_n[m,k]$ and $\Gamma_0[e,f]$ will be intersecting. Thus A_k is in the interior of δ. See figure 1. Therefore A'_k is in the interior of the closed curve $\delta' = \beta \cup \Gamma'_0[f,e]$ where

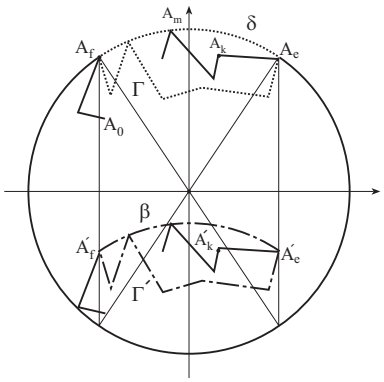


Fig. 1. *If Γ_n contains no limit-joints, Γ'_n has no rim joint.*

β is the translation of arc $\overset{\frown}{A_fA_e}$ by the vector $\mathbf{d}_\varepsilon$. But A'_t is in the exterior of δ'. So $\Gamma'_n[k,t]$ intersects boundary of δ'. That is a contradiction. Because $\Gamma_n[k,t]$ does not intersect boundary of δ. □

By lemma 8, we suppose A_e and A_f don't belong to the same subchain. From now on, the subchain which contains A_e is denoted by Γ_e and the other subchain which contains A_f is denoted by Γ_f. We have the following theorem.

Theorem 9 *At the end of translation, at least one of the subchains Γ_e and Γ_f can rotate about A_k.*

PROOF. Refer to full paper. □

Corollary 10 $|\alpha_i - \pi| > |\alpha_{i+1} - \pi|$.

By corollary 10, the configuration of Γ in two consecutive phase is different. Thus during the algorithm 2, straightening A_k is strongly progressed and cycling is not possible. Now for showing that after a finite number of repeats, algorithm 2 is terminated, we use simplicity of Γ. Assume without loss of generality that $\theta_k < \pi$. Thus according to definition of joint angle, Γ_0 must rotate about A_k clockwise. First suppose there is no confining region. So A_k can be straightened and then Γ_0 can rotate about A_k clockwise more, until first self-touching is occurred. This operation is called π-*passage motion* and the joint angle at A_k is denoted by $\pi + \tau_k$, that $\tau_k > 0$. Now suppose Γ is inside $C(O,r)$. We change the stopping criteria of algorithm 2, from achieving π to achieving $\pi + \tau_k$ and use this new algorithm on A_k. All of above proofs also hold for this new algorithm. So by corollary 10 we also have:

$$|\alpha_{i+1} - \pi - \tau_k| < |\alpha_i - \pi - \tau_k| \qquad (*)$$

Assumption $\theta_k < \pi$ yields: for every $i \geq 0$, $\alpha_i \leq \pi+\tau_k$. So $(*)$ yields $\pi+\tau_k-\alpha_{i+1} < \pi+\tau_k-\alpha_i$. In the other words, $\{\alpha_i\}_{i\geq 0}$ is a bounded and monotone sequence. Therefore it converges to its suprimum, $\pi+\tau_k$. Thus for every $\varepsilon > 0$ exists a finite natural number $N > 0$ such that for every $i \geq N$ we have $|\alpha_i - \pi - \tau_k| < \varepsilon$, i.e., for every $i \geq N$, $\pi+\tau_k-\alpha_i < \varepsilon$. Thus for $\varepsilon = \tau_k$, there is a finite number N_τ such that for all $i \geq N_\tau$, $\pi + \tau_k - \alpha_i < \tau_k$. So for $i = N_\tau$, we have $\pi + \tau_k - \alpha_{N_\tau} < \tau_k$, i.e., $\alpha_{N_\tau} > \pi$. Because N_τ is the smallest natural number that $\pi+\tau_k-\alpha_i < \tau_k$, we have $\pi+\tau_k-\alpha_{N_\tau-1} \geq \tau_k$, i.e., $\alpha_{N_\tau-1} \leq \pi$. Therefore A_k can straighten in phase N_τ or $N_\tau - 1$. Because $\alpha_{N_\tau-1} \leq \pi$ and $\alpha_{N_\tau} > \pi$. It is clear that N_τ is independent of n.

Therefore proof of correctness of algorithm 2 is terminated. Complexity of algorithm 1 is $O(n^2)$, because complexity of each step is $O(n)$ and the number of repeats is $n-1$.

5. Arbitrary Simple Chains

Now we prove that an arbitrary simple chain can be straightened inside a circle. First we have the following theorem.

Theorem 11 *Any simple chain Γ can be straightened using a finite number of monotonic single-joint bend operations.*

PROOF. See [1] and [2]. □

Theorem 11 is true, if chain is inside a circle as a confining region.

Theorem 12 *A simple chain Γ can be straightened inside a circle using a finite number of monotonic single-joint bend operations, if* $\sum_{i=0}^{n} l_i < 2r$.

PROOF. By theorem 11, Γ can be straightened in the plane using a finite number of monotonic single-joint bend operations. If all of these bend operations are complete, Γ can be straightened by using algorithm 1. But if at least one of the bend operations is not complete, these bend operations will be in accordance with a sequence of motions, $M = \{M_j\}_{j=1}^{k}$, such that M_j is a partial bend operation and Γ can be straightened by using M. Any operation M_j is single-joint, so it is in accordance with a joint A_{i_j} and this accordance is not one to one, because M_j s are not complete. Suppose any bend operation M_j is changed joint angle at A_{i_j} by $\Delta(A_{i_j}; j)$. Now note that any operation M_j is monotone; so if we change the stopping criteria of algorithm 2, from achieving π to achieving $\theta_{i_j} + \Delta(A_{i_j}; j)$, this new algorithm can be used to perform any bend operation M_j inside C. Therefore Γ can be straightened inside C by performing M_j s in turn, because k is finite. □

6. Conclusion

Assume that Γ is a simple chain such that $\sum_{i=1}^{n} l_i < 2r$ and Γ_1 and Γ_2 are two configuration of Γ inside circle $C(O, r)$. We denote their straight configurations by L_1 and L_2, respectively. Let M be a sequence of bend operations for straightening Γ_2 inside C and M^R be the reverse of Motion M. It is clear that by performing M^R, L_2 can be reconfigured to Γ_2. Now by theorem 11, we can reconfigure Γ_1 to L_1, then we can reconfigure L_1 to L_2 by translation and rotation operations and finally we can reconfigure L_2 to Γ_2 by M^R. So Γ_1 can be reconfigured to Γ_2 inside C, i.e., for a simple chain Γ, if $\sum_{i=1}^{n} l_i < 2r$, then any two configurations of Γ, inside circle $C(O, r)$ are equivalent.

References

[1] E.M. Arkin, S.P. Fekete and J.S.B. Mitchelle, An algorithmic study of manufacturing paperclips and other folded structures. *Computational Geometry: Theory and Applications* **25**, 117-138, (2003).

[2] R. Connelly, E.Demaine and G. Rote, Every polygon can be untangled. In *Proc. 41st Annu. IEEE Sympos. Found. Comput. Sci.*, pp. 432-442, (2000).

[3] J. Hopcroft, D. Josehp, and S. Whitesides, On the movement of robot arms in 2-dimensional bounded regions. *SIAM J. Compt.* **14**, 315-333, (1985).

[4] N. Pei, On the reconfiguration and reachability of chains. Ph.D. Thesis, School of Computer Science, McGill University, November, 1996.

[5] I. Streinu, A combinatorial approach to planar non-colliding robot arm motion planning. In *Proc. 41st Annu. IEEE Sympos. Found. Comput. Sci.*, pp. 443-453, (2000).

On Rectangular Cartograms

Marc van Kreveld [a] and Bettina Speckmann [b]
[a] *Inst. of Information & Computing Sciences, Utrecht University*
[b] *Dep. of Mathematics & Computer Science, TU Eindhoven*

Abstract

A rectangular cartogram is a type of map where every region is a rectangle. The size of the rectangles is chosen such that their areas represent a geographic variable (for example population). Rectangular cartograms are a useful tool to visualize statistical data. However, good cartograms are generally hard to generate: The area specifications for each rectangle may make it impossible to realize correct adjacencies between the regions and so hamper the intuitive understanding of the map.

Here we present the first fully automated algorithms for rectangular cartograms. Our algorithms depend on a precise formalization of region adjacencies and are building upon existing VLSI layout algorithms. Furthermore, we characterize a non-trivial class of rectangular subdivisions for which exact cartograms can be efficiently computed. An implementation of our algorithms and various tests show that in practice, visually pleasing rectangular cartograms with small cartographic error can be effectively generated.

1. Introduction

Cartograms. Cartograms are a useful and intuitive tool to visualize statistical data about a set of regions like countries, states or counties. The size of a region in a cartogram corresponds to a particular geographic variable [1,6]. Since the sizes of the regions are not their true sizes they generally cannot keep both their shape and their adjacencies. A good cartogram, however, preserves the recognizability in some way. Globally speaking, there are three types of cartogram. The standard type (the *contiguous area cartogram*) has deformed regions so that the desired sizes can be obtained and the adjacencies kept. Algorithms for such cartograms are described in [10,2,3]. The second type of cartogram is the non-contiguous area cartogram [7]. The regions have the true shape, but are scaled down and generally do not touch anymore. The third type of cartogram is the rectangular cartogram introduced by Raisz in 1934 [8] where each region is represented by a single rectangle.

Quality criteria. Whether a rectangular cartogram is good is determined by several factors. One of these is the *cartographic error* of the cartogram [2,3], which is defined for each region as $|A_c - A_s| / A_s$, where A_c is the area of the region in the cartogram and A_s is the area of that region as specified by the geographic variable to be displayed. The following list shows all quality criteria:

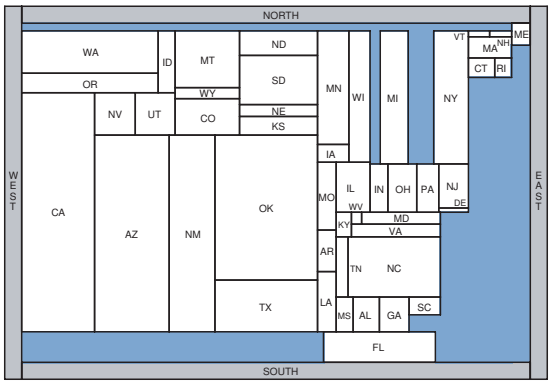


Fig. 1. A cartogram depicting the native population of the United States.

- Average cartographic error.
- Maximum cartographic error.
- Correct adjacencies of the rectangles.
- Maximum aspect ratio.
- Suitable relative positions.

For a purely rectangular cartogram we cannot expect to simultaneously satisfy all criteria well.

Related work. Rectangular cartograms are closely related to *floor plans* for electronic chips. Floor planning aims to represent a planar graph by its *rectangular dual*, defined as follows. A *rectangular partition* of a rectangle R is a partition of R into a set $\mathcal{R}$ of non-overlapping rectangles such no four rectangles in $\mathcal{R}$ meet at the same point. A rectangular dual of a planar graph (G, V) is a

rectangular partition $\mathcal{R}$, such that (i) there is a one-to-one correspondence between the rectangles in $\mathcal{R}$ and the nodes in G, and (ii) two rectangles in $\mathcal{R}$ share a common boundary if and only if the corresponding nodes in G are connected. The following theorem was proven in [5]:

Theorem 1 *A planar graph G has a rectangular dual R with four rectangles on the boundary of R if and only if*

(i) every interior face is a triangle and the exterior face is a quadrangle

(ii) G has no separating triangles.

Note that although every triangulated planar graph without separating triangles has a rectangular dual this does not imply that an error free cartogram for this graph exists.

Results. We present the first fully automated algorithms for the computation of rectangular cartograms. We formalize the region adjacencies based on their geographic location and are so able to enumerate and process all feasible *rectangular layouts* for a particular subdivision (i.e., map). The precise steps that lead us from the input data to an algorithmically processable rectangular subdivision are sketched in Section 2.

We describe three algorithms that compute a cartogram from a rectangular layout. The first is an easy and efficient (*segment moving*) heuristic which we evaluated experimentally. The visually pleasing results of our implementation can be found in Section 3. Secondly, we show how to formulate the computation of a cartogram as a bilinear programming problem. The details concerning these two methods can be found in the full paper.

For our third, exact, algorithm we introduce an effective generalization of sliceable layouts, namely *L-shape destructible* layouts. We show that:

Theorem 2 *An L-shape destructible layout with a given set of area values has exactly one or no realization as a cartogram.*

The proof of Theorem 2 (which is given in the full paper) immediately implies an efficient algorithm to compute an exact (up to an arbitrarily small error) cartogram for subdivisions that arise from actual maps.

Finally we also establish the following theorem:

Theorem 3 *A subdivision S with n vertices whose face graph is outerplanar can be represented by a rectangular cartogram with the correct adjacencies and any pre-specified area values. This cartogram can be computed in linear time.*

2. Algorithmic Outline

Assume that we are given an administrative subdivision into a set of regions. The adjacencies of the regions can be represented in a graph F, which is the face graph of the subdivision.

1. Preprocessing: The face graph F is in most cases already triangulated (except for its outer face). In order to construct a rectangular dual of F we first have to process internal vertices of degree less than four and then triangulate any remaining non-triangular faces.

2. Directed edge labels: Any two nodes in the face graph have at least one direction of adjacency which follows naturally from their geographic location. While in theory there are four different directions of adjacency any two nodes can have, in practice only one or two directions are reasonable.

Our algorithms go through all possible combinations of direction assignments and determine which one gives a correct or the best result. While in theory there can be an exponential number of options, in practice there is often only one natural choice for the direction of adjacency between two regions. We call a particular choice of adjacency directions a *directed edge labeling*.

Definition 4 (Realizable directed edge labeling)

A face graph F with a directed edge labeling can be represented by a rectangular dual if and only if

(i) every internal region has at least one North, one South, one East, and one West neighbor.

(ii) when traversing the neighbors of a node in clockwise order starting at the western most North neighbor we first encounter all North neighbors, then all East neighbors, then all South neighbors and finally all West neighbors.

3. Rectangular layout: A realizable directed edge labeling constitutes a *regular edge labeling* for F as defined in [4]. Therefore we can employ the algorithm by He and Kant [4] to construct a *rectangular layout*, i.e., the unique rectangular dual of a realizable directed edge labeling. The output of our implementation of the algorithm by He and Kant is shown in Figure 2.

4. Area assignment: For a given set of area values and a given rectangular layout we would like to decide if an exact assignment of the area values to the regions is possible without destroying

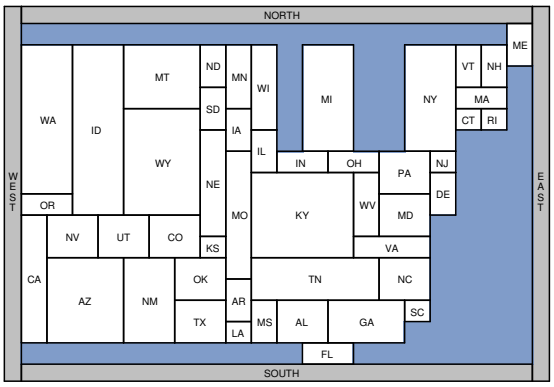


Fig. 2. One of 4608 possible rectangular layouts of the US.

the correct adjacencies. Should the answer be negative or should the question be undecidable, then we still want to compute a cartogram that has a small amount of cartographic error while maintaining reasonable aspect ratios and relative positions.

Exact algorithm. For certain types of rectangular layouts we can compute an exact cartogram. We first determine for a given rectangular layout a *maximal rectangle hierarchy*. The maximal rectangle hierarchy groups rectangles that together form a larger rectangle, as illustrated in Figure 3. It can be computed in linear time [9]. All groups in the hierarchy are independent and we will determine areas separately for each group.

A node of degree 2 in the hierarchy corresponds to a sliceable group of rectangles. If the maximal rectangle hierarchy consists of slicing cuts only, then we can (in a top-down manner) compute the unique position of each slicing cut.

Nodes of a degree higher than 2 require more complex cuts (see for example the four thick segments in Fig. 3). Here we introduce a type of non-sliceable layout for which the coordinates are still uniquely determined by the specified areas. We say a rectangular layout $\mathcal{R}$ is *irreducible* if no proper

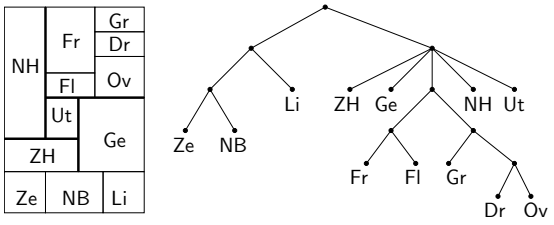


Fig. 3. Rectangular layout of the 12 provinces of the Netherlands and a corresponding maximal rectangle hierarchy.

subset of $\mathcal{R}$ (of size greater than one) forms a rectangle. Furthermore, we call a rectilinear simple polygon with at most 6 vertices *L-shaped*. We say that an L-shaped polygon is *rooted* at a vertex p if one of its convex vertices is p.

Definition 5 (L-shape destructible) *An irreducible rectangular layout $\mathcal{R}$ of a rectangle R is* L-shape destructible *if there is a sequence starting at a corner s of R in which the rectangles of $\mathcal{R}$ can be removed from R such that the remainder forms an L-shaped polygon rooted at the corner t of R opposite to s after each removal.*

Fig. 4. An L-shape destructible layout of the Eastern US. The shaded area shows the L-shaped polygon after the removal of rectangles $1-4$.

3. Implementation and experiments

We have implemented the segment moving heuristic and tested it on some data sets. The main objective was to discover whether rectangular cartograms with reasonably small cartographic error exist, given that they are rather restrictive in the possibilities to represent all rectangle areas correctly. Secondary objectives were to determine how the cartographic error depends on maximum aspect ratio and correct or false adjacencies.

Our layout data sets consist of the 12 provinces of the Netherlands and the 48 contiguous states of the USA. Here we present only some of the results pertaining to the US data sets, all other results can be found in the full paper. We allowed 13 pairs of adjacent states of the USA to be in different relative positions leading to 8192 possible layouts. Only 4608 of these correspond to a realizable directed edge labeling. We considered all 4608 layouts and chose the one giving the lowest average error as the cartogram.

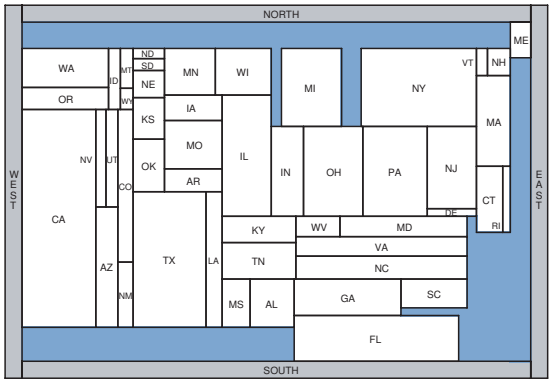


Fig. 5. The population of the US.

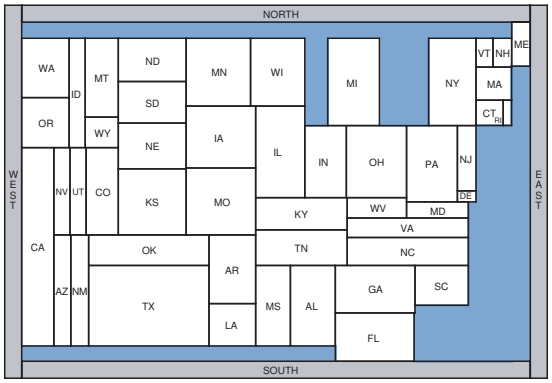


Fig. 6. The highway kilometers of the US.

Data set	Adjacency	Aspect ratio	Av. error	Max.
US pop.	false	8	0.104	0.278
US pop.	false	9	0.085	0.193
US pop.	false	10	0.052	0.295
US pop.	false	11	0.030	0.091
US pop.	false	12	0.022	0.056
US pop.	correct	12	0.327	0.618
US pop.	correct	13	0.319	0.608
US pop.	correct	14	0.317	0.612
US pop.	correct	15	0.314	0.569
US pop.	correct	16	0.308	0.612
US hw.	correct	6	0.073	0.188
US hw.	correct	7	0.059	0.111
US hw.	correct	8	0.058	0.101
US hw.	correct	9	0.058	0.101
US hw.	correct	10	0.058	0.101

Table 1. Errors for different aspect ratios, and correct/false adjacencies. Sea 20%.

As numeric data we considered *population*, *native population*, number of *farms*, number of *electorial college votes*, and total length of *highways*.

Table 1 shows errors for various settings for two US data sets. Only the average error is significant in the table, since the rectangular layout chosen for the table is the one with lowest average error. The corresponding maximum error is shown only for completeness. Since cartograms are interpreted visually and show a global picture, errors of a few percent on the average are acceptable. Such errors are also present in standard contiguous cartograms.

The error decreases with a larger aspect ratio, as expected. For the native population data set an aspect ratio of 7 combined with false adjacency results in a cartogram with average error below 0.04 (see Fig. 1). Figures 5 and 6 show rectangular cartograms for two of the five US data sets. The data sets allowed an aspect ratio of 10 or lower to yield an average error between 0.03 and 0.06.

References

[1] B. Dent. *Cartography - thematic map design*. McGraw-Hill, 5th edition, 1999.

[2] J. A. Dougenik, N. R. Chrisman, and D. R. Niemeyer. An algorithm to construct continous area cartograms. *Professional Geographer*, 37:75–81, 1985.

[3] H. Edelsbrunner and E. Waupotitsch. A combinatorial approach to cartograms. *Comput. Geom. Theory Appl.*, 7:343–360, 1997.

[4] G. Kant and X. He. Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems. *Theor. Comp. Sci.*, 172:175–193, 1997.

[5] K. Koźmiński and E. Kinnen. Rectangular dual of planar graphs. *Networks*, 5:145–157, 1985.

[6] NCGIA / USGS. Cartogram Central, 2002. `http://www.ncgia.ucsb.edu/projects/Cartogram_Central/index.html`.

[7] J. Olson. Noncontiguous area cartograms. *Professional Geographer*, 28:371–380, 1976.

[8] E. Raisz. The rectangular statistical cartogram. *The Geographical Review*, 24:292–296, 1934.

[9] S. Sur-Kolay and B. Bhattacharya. The cycle structure of channel graphs in nonslicable floorplans and a unified algorithm for feasible routing order. In *Proc. IEEE International Conference on Computer Design*, pages 524–527, 1991.

[10] W.Tobler. Pseudo-cartograms. *The American Cartographer*, 13:43–50, 1986.

Farthest-Point Queries with Geometric and Combinatorial Constraints

Ovidiu Daescu [a], Ningfang Mi [a], Chan-Su Shin [b], and Alexander Wolff [c]

[a] *Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083, USA*
[b] *School of Electronics and Information Engineering, Hankuk University of Foreign Studies, Korea*
[c] *Faculty of Computer Science, Karlsruhe University, Germany. WWW: i11www.ilkd.uka.de/algo/people/awolff*

1. Introduction

In this paper we discuss farthest-point problems, in which a sequence $S = (p_1, p_2, \ldots, p_n)$ of n points in the plane is given in advance and can be preprocessed to answer various queries efficiently. We first consider the general setting where query points can be arbitrary, then we investigate a special setting where each point in S is queried exactly once.

To describe our problems, we use the following notation. Given two points $p \neq q$, let pq denote the line through the points p and q, let $\overline{pq}$ denote the line segment joining p and q, and let $|pq|$ be the Euclidean distance of p and q. We will use ε to denote a fixed arbitrarily small positive number.

FarthestPointAboveLine (FPAL): Given a pair (q, l_q), where q is a point and l_q is a line through q, decide whether there is a point in S above l_q, and if yes report the one farthest from q.

Our solution to this problem is a data structure based on [6] that takes $O(n \log n)$ space, $O(n^{1+\varepsilon})$ preprocessing time and $O(n^{1/2+\varepsilon})$ query time. We can do better if S is in convex position:

FPALinConvexPolygon (FPALCP): Given a convex n-gon C and a pair (q, l_q), where q is a point and l_q is a line through q, decide whether there is a point in C above l_q, and if yes report the one farthest from q.

Our solution takes $O(n \log n)$ space, $O(n \log^2 n)$ preprocessing time and $O(\log^2 n)$ query time.

FarthestPointSameSide (FPSS): Given a triplet $(q, \mathcal{L}_q, \Delta)$, where q is a point and $\mathcal{L}_q$ is a line through q such that all points in S are within distance Δ from $\mathcal{L}_q$, decide whether there is a point $p \in S$ such that (i) $|qp| > \Delta$, and (ii) p is above the line l_q orthogonal to $\mathcal{L}_q$ at q. If yes, report the point p farthest from q that fulfills (ii).

Our data structure for this problem has the same time and space bounds as that for FPALCP.

FarthestIndexedPointOtherSide (FIPOS): Given a triplet (i, j, Δ), such that $1 \leq i < j \leq n$ and all points $p_k \in S$ with $i < k < j$ are within distance Δ from $p_i p_j$, decide whether there is a point p_k with $i < k < j$ such that (i) $|p_i p_k| > \Delta$, and (ii) p_k and p_j lie on different sides of the line that goes through p_i and is orthogonal to $p_i p_j$. If yes, report the point p_k farthest from p_i that fulfills (ii).

Our time and space bounds for this problem are by a log-factor above those for FPALCP.

In the special setting where each point in S is queried exactly once we investigate the following problem. We assume the existence of a point $p \notin S$.

BatchedFarthestIndexedPointSameSide (BFIPSS): For each point $p_i \in S$ decide whether there is a point $p_f \in \{p_1, \ldots, p_i\}$ that lies on the same side as p with respect to the perpendicular bisector of p and p_i. If yes report the point p_f farthest from p that has the above property.

Our algorithm for this problem runs in $O(n \log^2 n)$ time and uses $O(n \log n)$ space.

The problems we study are related to the nearest-point query problem, to the all-pairs farthest and closest neighbors problem, and to the closest-point-to-line query problem. Although these problems are well understood, we are not aware of work on the problems we consider.

Applications of our data structures are polygonal chain approximation [2], approximate solutions for the one-cylinder problem [1], and geometric spanning trees [5].

Email addresses: daescu@utdallas.edu (Ovidiu Daescu), nxm024100@utdallas.edu (Ningfang Mi), cssin@hufs.ac.kr (Chan-Su Shin).

2. Farthest-Point Queries

We first tackle FPALCP, since it is part of the solutions for the problems FPSS and FIPOS.

In the preprocessing phase, we construct a balanced binary tree T in $O(n \log^2 n)$ time as follows. The vertices of the convex polygon C, in counter-clockwise order from the rightmost vertex, are associated with the leaves of T. At each internal node u, we compute and store the convex hull and the farthest-point Voronoi diagram V_u of the leaf descendants of u from the information available at the children of u. We then preprocess V_u for planar point-location queries, which takes a total of $O(n \log n)$ time for each level of T. Thus, the overall computation of T takes $O(n \log^2 n)$ time.

Given a query pair (q, l_q), we find in $O(\log n)$ time the intersection points of l_q with the boundary ∂C of C by binary search. We assume that l_q has non-empty intersection with the interior of C (the four other cases are trivial). The sought point is either one of the two intersection points of l_q with ∂C or a vertex of C that is above l_q. Without loss of generality, we assume that no vertex of C lies on l_q and that l_q has positive slope.

We query T as follows. We select the two endpoints of the segments intersected by l_q that are above l_q. Let s be the first and t the second endpoint in counter-clockwise order on ∂C. We walk in T from s to t and collect a set $\mathcal{V}$ of $O(\log n)$ farthest-point Voronoi diagrams in two phases. In the ascending phase we go upwards from s. Whenever we get to a node u from its *left* child, we add to $\mathcal{V}$ the Voronoi diagram stored at the right child of u. In the descending phase we go down towards t. Whenever we go to the *right* child of a node u, we add to $\mathcal{V}$ the Voronoi diagram stored at the left child of u. Clearly, all points associated with these Voronoi diagrams are above l_q and thus the sought vertex is either s, t or one of these points. We locate q in $O(\log n)$ time in each Voronoi diagram in $\mathcal{V}$ and keep track of the point farthest from q. Thus we can answer the query in $O(\log^2 n)$ time.

Theorem 1 *There is a data structure for FPALCP that takes $O(n \log n)$ space, $O(n \log^2 n)$ preprocessing time and $O(\log^2 n)$ query time.*

One can answer queries for FPSS using the same approach as for FPAL: construct a partition tree based on a fine simplicial partition in $O(n^{1+\varepsilon})$ time [6] and enhance it with a second-level data structure consisting of the farthest-point Voronoi diagram (of the points at each internal node) preprocessed for planar point location. This data structure can be used to obtain a collection of disjoint subsets of points representing all points in S which are above the line l_q, and then query the farthest-point Voronoi diagrams for the points in each subset in order to answer the query. Note that the point farthest from the query point q may lie *inside* the convex hull C of S, so it seems our solution of FPALCP cannot be applied here. The following lemma, however, does give us a way to use the FPALCP data structures to solve FPSS.

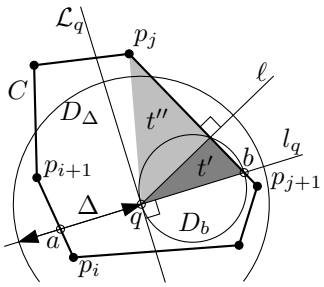


Fig. 1. The point p farthest from q must be a vertex of the convex hull C of S if $|qp| > \Delta$.

Lemma 2 *Given $S \subset \mathbb{R}^2$ and a triplet $(q, \mathcal{L}_q, \Delta)$, where q is a point and $\mathcal{L}_q$ is a line through q such that all points in S are within distance Δ from $\mathcal{L}_q$, if there is a point $p \in S$ such that (i) $|qp| > \Delta$, and (ii) p is above the line l_q orthogonal to $\mathcal{L}_q$ at q, then the point in S farthest from q is a vertex of the convex hull C of S.*

PROOF. Wlog. we assume that l_q intersects ∂C in two line segments $\overline{p_i p_{i+1}}$ and $\overline{p_j p_{j+1}}$ with $1 \leq i < j < n$ and that $p_{i+1}, \ldots, p_j$ all lie above l_q. Let a and b be the intersection points. Clearly for each triangle qp_kp_{k+1} above l_q either p_k or p_{k+1} is farthest from q. We now consider the triangle $t = qp_jb$, the triangle qap_{i+1} is analogous.

Let ℓ be the line through q that is orthogonal to p_jp_{j+1}. Consider the right-angled triangle t' (shaded dark in Figure 1) that is defined by ℓ, p_jp_{j+1} and qb. Due to Thales' theorem t' is contained in the disk D_b whose diameter is $\overline{qb}$. Since $|qb| \leq \Delta$, D_b (and thus t') is contained in the radius-Δ disk D_Δ centered at q.

Now if $q_j \in t'$ then $t \subseteq t' \subseteq D_\Delta$. Otherwise ℓ splits t into t' and another right-angled triangle t'' (shaded lightly in Figure 1) that is defined by qp_j, p_jp_{j+1} and ℓ. Since $\overline{qp_j}$ is the hypothenuse of t'',

p_j is farthest from q in t''. Thus the point farthest from q either lies in D_Δ or is a vertex of C. □

Theorem 3 *There is a data structure for FPSS that takes $O(n \log n)$ space, $O(n \log^2 n)$ preprocessing time and $O(\log^2 n)$ query time.*

To solve FIPOS, the indexed version of FPSS, in the preprocessing phase we construct an $O(n \log^2 n)$-size balanced binary tree T as follows. Each leaf of T is associated with a point in S such that the point $p_i \in S$ is stored at the i-th leaf of T. We go up the tree T and, at each internal node v, we compute and store the convex hull C_v of the leaf descendants of v. We also compute and store at v a secondary level data structure in the form of a balanced binary tree T_v of the farthest-point Voronoi diagrams on the vertices of C_v, enhanced with a planar point-location data structure, similar to the one used in solving FPSS. Then, the overall computation of T takes $O(n \log^3 n)$ time and requires $O(n \log^2 n)$ space. For a query pair (i, j), let π_{ij} be the path in T from p_i to p_j. We use this path to obtain a set $\mathcal{C}$ of $O(\log n)$ convex hulls whose union contains only the points p_k with $i < k < j$ as follows. In the ascending phase we add to $\mathcal{C}$ the convex hull stored at the right child of v if π_{ij} gets to v from its left child. In the descending phase we add to $\mathcal{C}$ the convex hull stored at the left child of v if π_{ij} goes from v to its right child. For each convex hull $C \in \mathcal{C}$, querying the secondary data structure reduces to FPSS. Let t be the number of vertices on C. Then we can determine in $O(\log^2 t)$ time whether there is a k, $i < k < j$, such that the point $p_k \in S$ associated with C satisfies the two FIPOS conditions. Since the size of the set $\mathcal{C}$ is $O(\log n)$, the overall query time is $O(\log^3 n)$.

Theorem 4 *There is a data structure for FIPOS that takes $O(n \log^2 n)$ space, $O(n \log^3 n)$ preprocessing time and $O(\log^3 n)$ query time.*

Building on our solutions of FPALCP, FPSS, and FIPOS, we extend a recent result in [3].

Theorem 5 *Given a polygonal chain $P = (p_1, p_2, \ldots, p_n)$ in the plane, the min-# problem with the tolerance zone criterion and L_2 distance metric can be solved in $O(F(m)\, n \log^3 n)$ time with $O(n \log^2 n)$ space, where $F(m)$ is the number of vertices of the path approximation graph reachable from p_1 with at most $m - 2$ edges and m is the number of vertices of an optimal approximating path.*

A version of BFIPSS without the index restriction has been considered in [5]. There the problem was to report for each point $p_i \in S$ a point farthest from the fixed point $p \notin S$ that lies on the same side as p with respect to the perpendicular bisector of p and p_i. In [5] the problem is reduced to the problem of finding for each $p_i \in S$ the first disk in a sequence of disks that does *not* contain p_i. This problem has been addressed in [2] under the name *off-line ball exclusion search (OLBES)*. The authors set up a tree data structure with a space requirement of $O(n \log n)$ and then query this structure with each point in S. This results in a total running time of $O(n \log n)$ for OLBES in dimension $d = 2$. For $d > 2$ the problem is solved differently in $O(n^{2-2/(\lfloor d/2 \rfloor)+1})$ time. In [5], a version of OLBES where all disks intersect a common point has been solved for $d = 2$ in $O(n \log n)$ time and $O(n)$ space by sweeping an arrangement of circular arcs. To solve BFIPSS in dimension 2 we set up a tree data structure similar to [4, proof of Lemma 2]. Here, however, we must solve a different OLBES problem in each query and thus need to modify our tree successively.

Theorem 6 *BFIPSS can be solved in $O(n \log^2 n)$ time and $O(n \log n)$ space for a sequence S of n points and a point $p \notin S$ in the plane.*

PROOF. Let $D_{n+1} = \emptyset$ and let $D_1, \ldots, D_n$ be the sequence of disks in order of non-increasing radius that are centered on the points in S and touch p. We build a binary tree $\mathcal{B}$ that we query with the points in S, and the answer of a query will correspond to the index of the first disk in the sequence $D_1, \ldots, D_{n+1}$ that does *not* contain the query point. The leaves of $\mathcal{B}$ correspond to these answers from left to right. Each inner node v stores the intersection I_v of all disks (except D_{n+1}) that correspond to the leaves in the subtree rooted at v. We label each node v with a pair $[a_v, b_v]$ encoding the set $S_v = \{a_v, \ldots, b_v\}$ of consecutive indices that correspond to these disks. In Figure 2 a tree with $n = 13$ is depicted. We build $\mathcal{B}$ in a bottom-up fashion. Each inner node has two children in the previous level, except possibly a level's rightmost node that can have a right child in an earlier level, see the node with label $[9, 13]$ in Figure 2.

Querying $\mathcal{B}$ with a query point q means to follow a path from the root to a leaf. In each inner node v with left child ℓ the test $q \in I_\ell$ is performed. If $q \in$

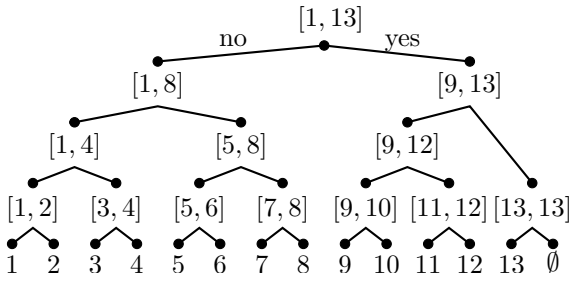


Fig. 2. The tree data structure $\mathcal{B}$ for $n = 13$.

I_ℓ, the query continues with the right, otherwise with the left child of v.

Other than in [2,4], we start with an empty skeleton of $\mathcal{B}$, i.e. all inner nodes v are labeled by $[a_v, b_v]$, but all intersections I_v are set to $\mathbb{R}^2$. Also other than in [4], the order in which we query becomes crucial. We go through the points $p_1, \ldots, p_n \in S$ in order of increasing index. Before querying $\mathcal{B}$ with p_i we update $\mathcal{B}$ by adding the new disk D_k centered on p_i (note that usually $k \neq i$) to the intersection I_v for each node v on the path from the root to the leaf that corresponds to D_k.

The result of a query with p_i is the disk D_j that corresponds to the leaf at the end of the query path π. If $j = n + 1$ then π is the rightmost root–leaf path. Consider the left children of the nodes on π. The sets S_ℓ that belong to these left children partition $\{1, \ldots, n\}$. In other words, the intersection of I_ℓ over these children is $D_1 \cap \cdots \cap D_n$. Since π is the rightmost root–leaf path, the containment queries in all nodes on π were answered positively. Thus q_i is contained in all disks currently in $\mathcal{B}$, i.e. $q_i \in D(q_1, p) \cap \cdots \cap D(q_i, p)$, where $D(a, b)$ is the disk centered at a that touches b. This means that none of $q_1, \ldots, q_i$ lies in the halfplane $h(p, q_i)$ that contains p and whose boundary is the perpendicular bisector of p and q_i. Otherwise [5, Lemma 1] would guarantee that $q_i \notin D(q_k, p)$ for the point q_k in $\{q_1, \ldots, q_i\}$ farthest from p in $h(p, q_i)$.

If $j \leq n$ we again consider the left children of the nodes on the query path π of q_i. The sets S_ℓ partition $\{1, \ldots, j-1\}$ if we take only those left children ℓ into account that do not themselves lie on π. Similarly to above, the intersection of I_ℓ over these children is $D_1 \cap \cdots \cap D_{j-1}$. Thus q_i is contained in all D_k with $k < j$ that are currently in $\mathcal{B}$. On the other hand, since π is not the rightmost root–leaf path, there must be left children that lie on π. The last such left child v is the root of the subtree whose rightmost leaf corresponds to D_j. Thus v is associated with some set $S_v = \{i_v, \ldots, j\}$, where $1 \leq i_v \leq j$. Since we have already observed that q_i is contained in all D_k with $k < j$ that are currently in $\mathcal{B}$, but π came to v via a "no"-branch, we now know that $q_i \notin D_j$. Let m be such that $D_j = D(q_m, p)$. Note that $q_i \notin D(q_m, p)$ means that $D(q_m, p)$ was inserted in $\mathcal{B}$ before querying with q_i, and thus $m \leq i$. Since $q_i \notin D(q_m, p)$, and $q_i \in D(q_r, p)$ for all $r \leq i$ with $|pq_r| > |pq_m|$, [5, Lemma 1] yields that q_m is farthest from p in $\{q_1, \ldots, q_i\} \cap h(p, q_i)$.

The running time is as follows. Querying $\mathcal{B}$ takes $O(\log^2 n)$ time since the height of $\mathcal{B}$ is $O(\log n)$ and in each node of the query path the query point has to be located in the intersection I_v of some disks, which can be done in $O(\log n)$ time.

When we update $\mathcal{B}$ by adding a new disk D_j, we have to go from the root to the leaf that corresponds to D_j. In each node on this path we must compute $I_v \cap D_j$ and update our data structure for I_v. This can be done in $O(\log n)$ time per node by a procedure detailed in [5, proof of Lemma 4]. Thus each update also takes $O(\log^2 n)$ time.

Now the running time of $O(n \log^2 n)$ is obvious. The space consumption is $O(n \log n)$ since a) each disk contributes only to intersections stored on the path from the root to "its" leaf, and b) a disk that contributes to some intersection I_v adds at most one arc to the boundary of I_v [5, Fact 2]. □

References

[1] M. Bădoiu, S. Har-Peled, and P. Indyk. Approximate clustering via core-sets. In *Proc. STOC'02*, pages 250–257, 2002.

[2] D. Z. Chen, O. Daescu, J. Hershberger, P. M. Kogge, and J. Snoeyink. Polygonal path approximation with angle constraints. In *Proc. SODA'01*, pages 342–343, Washington, D.C., 2001.

[3] O. Daescu and N. Mi. Polygonal path approximation: A query based approach. In *Proc. ISAAC'03*, volume 2906 of *LNCS*, pages 36–46, 2003. Springer.

[4] J. Gudmundsson, H. Haverkort, S.-M. Park, C.-S. Shin, and A. Wolff. Facility location and the geometric minimum-diameter spanning tree. In *Proc. APPROX '02*, vol. 2462 of *LNCS*, pages 146–160, 2002. Springer.

[5] J. Gudmundsson, H. Haverkort, S.-M. Park, C.-S. Shin, and A. Wolff. Facility location and the geometric minimum-diameter spanning tree. *Computational Geometry: Theory and Appl.*, 27(1):87–106, 2004.

[6] J. Matoušek. Efficient partition trees. *Discrete and Computational Geometry*, 8:315–334, 1992.

A quadratic distance bound on sliding between crossing-free spanning trees

Oswin Aichholzer [a,1] and Klaus Reinhardt [b]

[a] Institute for Softwaretechnology
Graz University of Technology, Austria

[b] Wilhelm-Schickard-Institut für Informatik
Universität Tübingen, Germany

Abstract

Let S be a set of n points in the plane and let $\mathcal{T}_S$ be the set of all crossing-free spanning trees of S. We show that any two trees in $\mathcal{T}_S$ can be transformed into each other by $O(n^2)$ local and constant-size edge slide operations. No polynomial upper bound for this task has been known, but in [1] a bound of $O(n^2 \log n)$ operations was conjectured.

Key words: crossing-free spanning tree, local transformation, edge slide

1. Introduction

Let S be a set of n points in the Euclidean plane. W.l.o.g. we assume that no two points of S have the same x-coordinate, otherwise we rotate the coordinate system appropriately. A *crossing-free spanning tree* of S is a tree whose edges connect all points in S (and no others) with straight line segments that pairwise do not cross. With $\mathcal{T}_S$ we denote the set of all crossing-free spanning trees of S.

An interesting question is whether, and how fast, two members of $\mathcal{T}_S$ can be transformed into each other by means of predefined rules, often called flips. A common operation is what is called an *edge move*, which relates two trees in the set $\mathcal{T}_S$ iff they have all but one edge in common (one edge is 'flipped'). For this general setting Avis and Fukuda [2] showed that the corresponding tree graph is connected and has a diameter bounded by $2n - 4$. If we restrict the set of allowed flips to planar, length-improving edge moves then in [1] a way to transform any tree $T \in \mathcal{T}_S$ into the minimum spanning tree of S in only $O(n \log n)$ steps was given. For a more detailed discussion and some historical background see [1].

Our interest is focused on a local edge move that keeps one endpoint of the moved edge fixed and moves the other one along an adjacent tree edge. Following [3], we will call this constant-size operation an edge slide. More formally the central operation we consider is defined as follows [1]: Consider a tree $T' \in \mathcal{T}_S$. A *(planar) edge slide* on T' takes some edge $e \in T'$ and moves one of its endpoints along some edge adjacent to e in T', without generating any edge crossings. This gives a new edge f and a new tree $T'' = T' \cup \{f\} \setminus \{e\}$ such that $T'' \in \mathcal{T}_S$. An edge slide is a special kind of planar edge move: T'' is obtained by closing with f a 3-cycle C in T' and by removing e from C, in a way such that T' avoids the interior of the triangle C. Intuitively speaking, an edge slide is an edge operation as local as it can be.

In this paper we investigate the questions of how fast two crossing-free spanning trees of $\mathcal{T}_S$ can be transformed into each other by means of the edge slide operation. To this end consider the *tree graph* $\mathcal{TG}(S)$ which is an undirected graph that has $\mathcal{T}_S$ as its set of nodes. It realizes an arc between two nodes (trees) T' and T'' if and only

Email addresses: oaich@ist.tugraz.at (Oswin Aichholzer), reinhard@informatik.uni-tuebingen.de (Klaus Reinhardt).

[1] Research supported by Acciones Integradas 2003-2004, Proj.Nr.1/2003

if T' can be transformed into T'' by an edge slide (and vice versa). In [1] it was shown that $\mathcal{T}G(S)$ is connected. The length of a shortest path in $\mathcal{T}G(S)$ corresponds to the distance between the two respective trees. However, for the edge slide operation no polynomial upper bound on this length has been known. It was conjectured that 'if two trees are part of the same triangulation of S then they can be transformed into each other by $O(n^2)$ edge slides'. By results in [1], this would give a diameter of $O(n^2 \log n)$ for the corresponding tree graph $\mathcal{T}G(S)$. We are able to prove the following, stronger result:

Theorem 1 *Let T' and T'' be any two crossing-free spanning trees of S. Then T' can be transformed into T'' by $O(n^2)$ edge slides.*

As mentioned in [1] the edge slide operation could also prove useful in enumerating all simple polygons on a point set S via constant-size local transformations. This question is still unsettled; see e.g. Hernando et al. [5]. Our upper bound on the diameter of $\mathcal{T}G(S)$ might be useful in this respect.

2. Upper Bound Construction

Let S and $T \in \mathcal{T}_S$ be as defined in Section 1. We call a pair (e, p_j), where $e = p_i p_k$ is an edge of T and $p_i, p_j, p_k \in S$ are sorted in x-order, a *slide triangle* if the open triangle $\Delta = p_i p_j p_k$ is free of points from S and edges from T, that is, the interior of Δ is empty.

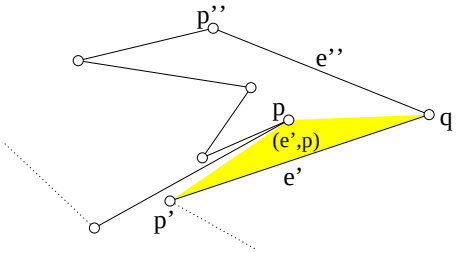


Fig. 1. A slide triangle (e', p), see Lemma 2.

Lemma 2 *Let P be a simple polygon with vertex set S, and let δP be the boundary of P with one marked edge e^*. If $\delta P \setminus \{e^*\}$ is no x-monotonous path then in the interior of P there always exists a slide triangle $(e, p) \subset P$, $e \neq e^*$.*

Proof Since $\delta P \setminus \{e^*\}$ is not x-monotonous there exists a vertex $q \in S$ with two edges from $\delta P \setminus \{e^*\}$ both emanating to the same side, i.e., both to the left or right of q. Let e' and e'', respectively, be these edges. W.l.o.g. we assume that they emanate from q to the left, e' lies below e'' and the left endpoint p' of e' lies to the left of the left endpoint $p''$of e'', see Figure 1 (all other cases are symmetric). If the open triangle $\Delta = qp'p''$ is empty we have a slide triangle (e', p''). Otherwise consider the point p which among all points of P in the interior of Δ minimizes the angle $\angle pp'q$ at p'. Note that there are no edges (partially) inside Δ that have q as an endpoint or intersect e' or e''. Therefore p provides a slide triangle (e', p). □

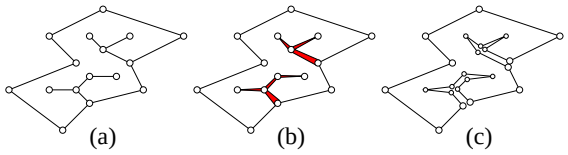


Fig. 2. Cutting a tree polygon (a) along interior edges (b) to obtain a simple polygon (c).

A *tree polygon* P is a simple polygon with interior points, each point connected to the boundary δP via a unique (simple) path such that the resulting graph is planar, see Figure 2(a). In other words, the graph without the edges of δP is a forest. We claim that we can handle this more general situation like a simple polygon: Cut along interior edges and move them apart at the cuts infinitesimally, i.e., duplicate the related vertices, see Figure 2(b) and (c). Observe that the proof of Lemma 2 still holds for this setting by considering edges e' and e'' that are neighboring in the cyclic order around q.

We call the x-monotonous path connecting all vertices of S in their x-sorted order the *canonical spanning tree* $T_c \in \mathcal{T}_S$ of S.

Theorem 3 *For a point set S and a crossing-free spanning tree $T \in \mathcal{T}_S$, $T \neq T_c$, there always exists a slide triangle (e, p), $p \in S$ and $e \in T$ such that the path $\pi \in T$ connecting p to e, say at point q, is x-monotonous. Moreover $\pi \cup pq$ is a simple polygon without interior points.*

Proof We first show that there always exists some slide triangle (e, p). The union of T and the boundary of the convex hull of S partitions S into $k \geq 1$ tree polygons P_i, $i = 1, \ldots, k$. Since T is a connected spanning tree each P_i has a unique edge which stems from the boundary of the convex hull of S. We mark these edges. From Lemma 2 and the discussion afterwards we know that we get a slide triangle inside some P_i unless for all P_i the remaining (non marked) part is x-monotonous. But in the latter case T must be x-monotonous, too, that is, $T = T_c$, a contradiction.

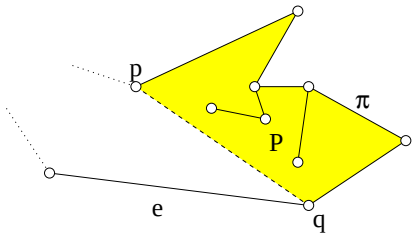


Fig. 3. A path π connecting p to q.

Let q be the (first) endpoint of e to which p is connected. If the edge pq belongs to T we are done. Thus assume that p is connected to q via a path π of length greater than 1, see Figure 3. Since (e, p) is a slide triangle the edge pq does not cross an edge of T. Thus the 'pocket' formed by π together with the edge pq and possible interior edges and points is a tree polygon P. If $\delta P \setminus pq$ is an x-monotonous path we are done. Otherwise we mark the edge pq and apply induction on P. Note that only one edge of P is marked, since T does not contain cycles. Moreover, in every induction step we obtain a smaller instance, since we get rid of at least one edge of T. □

For an edge e its weight is defined as the number of points from S which lie in the open x-interval spanned by e, that is, the number of points which lie between the endpoints of e in the x-sorted order. The weight of a tree T, denoted by $w(T)$, is the sum of the weights of its edges. Obviously T_c has weight zero and is the only tree with this property. Since each of the $n-1$ edges of T has at most weight $n-2$ the weight of a tree with n points is bounded by $(n-1)(n-2) < n^2$. A tight bound is given by the following lemma, for which we omit the proof in this extended abstract.

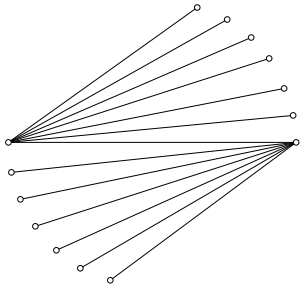


Fig. 4. A tree with maximum weight of $\frac{3n^2-10n+8}{4}$.

Lemma 4 *The weight $w(T)$ of a crossing-free spanning tree $T \in \mathcal{T}_S$ is bounded by $0 \leq w(T) \leq \lfloor \frac{3n^2-10n+8}{4} \rfloor$, $n \geq 2$, and these bounds are tight.*

Lemma 5 *Any crossing-free spanning tree $T \in \mathcal{T}_S$ can be transformed into T_c by at most $2 \cdot w(T)$ edge slides.*

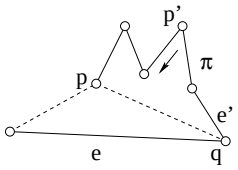


Fig. 5. A slide triangle (e, p) with x-monotonous path π connecting p to q.

Proof If $T = T_c$ the statement is obviously true, so let $T \neq T_c$. Let (e, p) be a slide triangle as provided by Theorem 3, see Figure 5. Let $k \geq 1$ be the number of edges of the x-monotonous path π connecting p to some endpoint q of e. We claim that we can reduce the weight of e by at least k by performing $2k-1$ edge slides. To this end let e' be the edge of π incident to q. Our first task is to slide e' along π to obtain the edge qp.

Assume that $k > 1$. Then π avoids the interior of the slide triangle (e, p) and thus contains at least one vertex p' pointed away from the edge qp. Since π is x-monotonous we can slide the edge of π which has p' as its left endpoint 'towards' p along the edge of π which has p' as its right endpoint. We repeat this process until we obtain the edge qp, i.e., $k = 1$. Since each edge slide reduces the length of the current path from p to q by one, we carry out exactly $k-1$ steps.

Now we can slide e along qp, reducing its weight by at least k (the vertices of π different from q). Finally we slide qp back to e' by reversing the steps of the first phase.

As long as the resulting tree is not T_c we repeat all above steps. After each iteration the weight of a single edge has been decreased by at least half of the number of the involved edge slide operations. We thus can transform T into T_c with at most $2w(T)$ edge slides. □

We are now ready to prove our main result as proposed in Section 1. We give here a more explicit statement and Theorem 1 then follows as a corollary.

Theorem 6 *For any pair $T', T'' \in \mathcal{T}_S$ we can transform T' into T'' by at most $2(w(T') + w(T'')) \leq 3n^2$ edge slides.*

Proof Lemma 5 shows that we can transform any tree $T' \in \mathcal{T}_S$ into T_c with at most $2w(T')$ edge slides. By symmetry of the edge slide operation we can use the reverse transformation for T''. Together with the upper bound $w(T'), w(T'') \leq \frac{3n^2}{4}$ from Lemma 4, the theorem follows. □

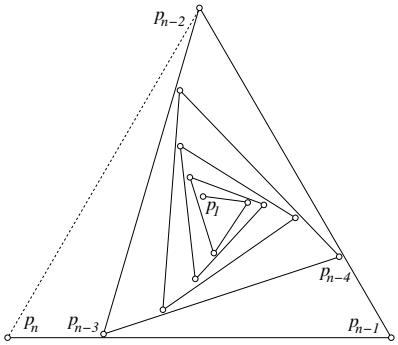


Fig. 6. To obtain the edge $p_n p_{n-2}$ requires $(n-1)(n-2)/2$ edge slides for odd $n \geq 3$.

Figure 6 shows that there are examples requiring $\Omega(n^2)$ edge slides to transform two spanning trees into each other. Thus the bound of Theorem 1 is tight. We omit the details on the lower bound construction in this extended abstract.

3. Discussion and Open Problems

One might wonder whether the slide-distance between two spanning trees which do not intersect each other is smaller than in the general case. A similar result holds for triangulations, where the flip-distance can be bounded by the number of crossing edges [4]. However, from the example in Figure 6 it follows that even for two trees differing in only one edge the slide-distance is quadratic.

Another observation is that the weight of a spanning-tree is direction-sensitive. So an obvious question is whether there always exists a 'nice' direction with sub-quadratic weight? Again a negative answer is given by the example of Figure 6, having weight $\Theta(n^2)$ for any direction of the x-axis.

So far we only obtained results on the number of necessary slide operations. On the algorithmic side we are also interested in the time complexity to compute the $O(n^2)$ slide sequence. We plan to investigate this question in the near future.

A related algorithmic question is how fast we can compute a direction to minimize the weight of a given tree. This can be done in time $O(n^2 \log n)$, but we omit the details in this extended abstract.

Acknowledgments

We would like to thank Ferran Hurtado for his presentation 'Flip' as the Paul Erdös lecture at CCCG 2003 in Halifax, Canada. This inspired us to (continue) work on the presented topic. We are grateful to Franz Aurenhammer and Hannes Krasser for enjoyable discussions and carefully reading the manuscript.

References

[1] O.Aichholzer, F.Aurenhammer, F.Hurtado *Sequences of spanning trees and a fixed tree theorem.* Computational Geometry: Theory and Applications, 21(1-2):3–20, 2002.

[2] D.Avis, K.Fukuda *Reverse search for enumeration.* Discrete Applied Mathematics 65: 618-632, 1996.

[3] W.Goddard, H.C.Swart *Distances between graphs under edge operations.* Discrete Mathematics 161: 121–132, 1996.

[4] S.Hanke, T.Ottmann, S.Schuierer *The edge-flipping distance of triangulations.* Journal of Universal Computer Science 2 (1996), 570-579.

[5] M.C.Hernando, M.E.Houle, F.Hurtado *On local transformation of polygons with visibility properties.* Proc. 6th International Computing and Combinatorics Conference COCOON'00, Springer LNCS 1858: 54–63, 2000.

Curvature criteria to fit curves to discrete data

Lyuba Alboul *,[a] Gilberto Echeverria [a] Marcos Rodrigues [a]
[a] *Sheffield Hallam University, Sheffield S1 1WB, UK*
http:www.shu.ac.uk/scis/artificial_intelligence

Abstract

Several geometric criteria to fit a polygonal closed curve to discrete two-dimensional data are considered and analysed. Most of these criteria are related to the concept of curvature, for example, one criterion is minimisation of total absolute curvature. On the basis of these criteria algorithms to construct a polygonal curve which is optimal with respect to a specific criterion, are designed.

Key words: Curve reconstruction, total absolute curvature, polygonisation

1. Introduction

The work presented in this communication is dedicated to the following problem: *Given discrete point data set S in 2D, find a curve that spans all these points and best satisfies a specific criterion.* We call this problem the *Curve modelling problem*, and in what follows we abbreviate it as the *Curvmod* problem. This problem occurs in particular forms in various applications. For example, an important problem in Computer Graphics is to find a planar curve that is shaped in the way as you want it to be shaped [8]. A well–known form of the *Curvmod* problem is the *Curve Reconstruction Problem* which is concerned with finding a piecewise linear approximation to a curve from a set of given sample points [6]. An algorithm for curve reconstruction should preserve the order of the points sampled from the curve, *i.e.*, the points are connected only if they are adjacent along the curve. Applications of this problem can be found in many areas, for example, detecting boundaries in Image Processing, determining patterns in Computer Vision, and reconstruction of topographic objects from aerial images.

Another form of the *Curvmod* problem is related to polygonisation of a given planar point set S. Studying the set of polygonisations of S is an active area of research in computational geometry. For example, one of the polygonisation problems is to determine a polygonisation (or polygonisations) with the minimum number of *reflex* vertices, as the number of reflex vertices quantifies in a combinatorial sense the degree to which the point set S is in convex position [5].

We restrict ourselves to polygonal curves. There are various algorithms for curve reconstructing, which give good results if the sampling of data points is dense enough [4,6]. We look at the problem from another point of view. We omit any additional information on the data points, and attempt to determine what 'reasonable' curves can be constructed for the given data and what characteristics these curves possess and how these characteristics distinguish one curve from another. In this setting the *Curvmod* problem can be viewed as an optimisation problem: a 'reasonable' curve can be defined as an *optimal curve* with respect to a certain criterion.

We introduce several geometrical criteria, mostly related to the concept of curvature, as curvatures truly describe the shape of objects. Research on discrete curvatures (*i.e.*, curvatures that can be computed on a set of points) is of growing interest in geometric modelling (some overview is given in [2]). An important measurement of an object is its size, some criteria related to this measurement are also considered.

* Corresponding author
Email address: L.Alboul@shu.ac.uk (Lyuba Alboul).

2. Preliminaries. Notions related to the concept of curvature

2.1. *Problem definition*

The definition of the curvature of a curve is in general given under the assumption that the curve is of C^2–class, and thus posseses two continuous derivatives. However, the notion of curvature can be generalised to the class of curves which possess continuous second derivatives except at a finite number of points where a jump discontinuity in the first derivative can occur. We can introduce the *exterior angle* $\alpha(s)$ formed by the right and left tangents at a point of discontinuity s [9]. In the case of a polygonal curve points of discontinuity are its vertices. In this paper we discuss only curves, which are continuous images of the circle S^1 into the plane, *i.e.*, *closed polygons*. Let us denote with $\alpha(v_i)$ the exterior angle at the vertex v_i of a curve. Then expression

$$\omega = \sum_{i=1}^{n} \alpha(v_i) \tag{1}$$

represents the total curvature of a closed polygonal curve. If this curve is the boundary of a closed simple polygon, ω is always equal to 2π. Its absolute value $|\omega|$, or *absolute total curvature* is more informative. An important fact is that $|\omega|$ reaches its minimal value 2π on convex curves (polygons). With respect to the concept of *total absolute curvature* we can formulate the following problem:

Problem 1. Given discrete two–dimensional data. Among all closed polygonal curves that span these data find a curve (or curves) with the minimal total absolute curvature.

Motivation to study polygonal curves of *minimal total absolute curvature* is related to the concept of Tight submanifolds [7]. One of the main properties of one– and two–dimensional tight submanifolds in $\mathbb{R}^3$ is that they possess the minimal total absolute curvature. For non–regular curves and surfaces such as polygonal curves and polyhedral surfaces the concept of curvature is also determined, and surface triangulations of minimal total absolute curvature (MTAC) were introduced. MTAC triangulation coincides with the convex triangulation of the data if the data are in convex position, but the properties of triangulations with MTAC for non-convex data are still mostly unknown (see [3] for a review). As surfaces are much more complex objects than curves, the study of the one–dimensional version of the MTAC triangulation provides useful insights for research on triangulations. Besides this, the study of polygonal curves with respect to their total absolute curvature is also interesting for its own sake, as the total absolute curvature is related to the global properties of the curve and might be used to characterise diverse curve profiles. We are interested to determine *to what extent a polygonal curve is represented by its total absolute curvature.*

2.2. *Some simple formulae and examples*

The total curvature and total absolute curvature coincide for a closed convex polygonal curve and both equal to 2π. Therefore for a non–convex curve the excess in the total absolute curvature with respect to the curvature of the convex curve can be used as a measure of deviation from the convex curve.

Given the data, we take as a *convex curve of reference* the boundary of the convex hull of the data, (referred to as the *CB–curve*). The statement of Problem 1 allows self–intersecting curves, but in this communication we presume that polygonal curves represent boundaries of simple polygons, *i.e.*, are not self–intersecting.

A simple polygonal curve L, that spans the data, is represented by its vertices V_1, V_2, ..., V_{n-1}, V_n. V_iV_j, where $j = i + 1$, is a line segment, and its length is denoted with l_{ij}, $l_{ij} = l_{ji}$. A *star* of a vertex V_i is the union of the vertex and its two adjacent line segments $V_{i-1}V_i$ and V_iV_{i+1}.

We designate the vertices that lie on *CB–curve* with V^i_{CB}, and with $\gamma_i, i = 1, ..., l$ the exterior angles at these vertices with respect to the *CB–curve*. It is clear that $\sum \gamma_i = 2\pi$.

Let us denote with V^j_{conv} convex vertices of L and with $\alpha_j, j = 1, ..., m$ the exterior angles at these vertices with respect to L, and with V^k_{reflex} – reflex vertices of L and with $\beta_k, i = 1, ..., p$, the corresponding exterior angles at these vertices with respect to L; $m + p = n$.

For a simple closed polygon the following equality holds:

$$\sum \alpha_j - \sum \beta_k = 2\pi \tag{2}$$

As $|\omega| = \sum \alpha_j + \sum \beta_k$, the total absolute curvature of a polygonal closed simple curve can be rep-

resented as

$$|\omega| = 2\pi + 2\sum_k \beta_k \tag{3}$$

The set of convex vertices V^j_{conv} can be further split into three disjoint subsets, namely, the subset $V^{j_1}_{conv-CB}$ of vertices that lie on the CB–curve and such that their stars belong to the CB–curve; the subset $V^{j_2}_{conv-int}$ of vertices that are convex but do not belong to the CB–curve; and the subset $V^{j_3}_{conv-corn}$ of vertices that lie on the CB–curve but their stars do not belong to the CB–curve. Vertices of the last type are called *corner vertices* as at these vertices the curve is deviated from the CB–curve. The corresponding exterior angles are denoted as α_{j_1}, α_{j_2} and α_{j_3}. Obviously, any part of the curve that is deviated from the CB–curve starts and ends at the neighbouring vertices of $V^{j_3}_{conv-corn}$, since our curve has no self–intersections. Each α_{j_3} is equal to $\gamma_{j_3} + \alpha(CB)_{j_3}$; where by $\alpha(CB)_{j_3}$ we denote the angle at a corner vertex of the curve L with respect to the CB–curve, or in other words, the angle between the edge of L that has this corner vertex as one of the end-vertices and the (imaginary) edge of the CB-curve, that would have the same corner vertex as one of the end-vertices. We call such an angle a *deviating angle*. After some computation, we obtain the following expression:

$$\sum(\alpha_{j_2} + \alpha(CB)_{j_3}) = \sum \beta_k \tag{4}$$

Therefore, to find the curve (or curves) of minimal total absolute curvature among all curves that span the given data, it is sufficient to minimise either the sum of exterior angles at reflex vertices or the sum of exterior angles at the internal convex vertices and the deviating angles.

From formula 4 it follows that if the curve does not have internal convex vertices then the amount of deviation of the curve from the convex curve CB is concentrated in deviating angles. Each deviated part L_d, $d = 1, ..., f$ contains in its turn some convex internal vertices $V^{j_2^d}_{conv_d}$ (its number may be equal to zero) and reflex vertices $V^{k^d}_{reflex_d}$

Equality 4 holds for each deviated part:

$$\sum(\alpha_{j_2^d} + \alpha(CB)_{j_3^d}) = \sum \beta_{k^d} \tag{5}$$

Let us define a convex region R_{conv} as a part $V_i, V_{i+1}, ..., V_{i+j}$ of a curve L that satisfies the condition that each vertex that belongs to R_{conv} is convex, but vertices V_{i-1} and V_{i+j+1} are reflex.

A concave region $R_{concave}$ is defined analogously. The following statement is true:

Suppose we construct a curve L_1 which has g convex regions and h concave ones. If we manage to add new vertices in such a way that the obtained regions are preserved, than a new curve L_2 will possess the same total absolute curvature as L_1.

The *spherical image* of a curve visualises the concept of *total absolute curvature*. The spherical image for a polygonal curve is constructed by means of outwards units normals to the line segments of the curve, all of them are 'translated' to the same origin. The ends of the unit normals of a planar curve will lie on the unit circle. Let us suppose that we walk around the boundary of a polygon, for example, in anticlockwise direction starting from vertex V_1, passing through all the vertices according to their order until we arrive again at the vertex V_1. To this walk a corresponding walk on the circle is generated, some of its parts are walked several time for a non–convex curve.

The spherical image provides a vertex classification of the curve as for a reflex vertex the direction will be opposite to the chosen one. The spherical images can be put in one–to–one correspondence for two curves of the same data set if the numbers of concavities/convexities and corresponding 'incorporated' curvatures for the both curves are the same. We say in this case that two curves are *curvature identical*. Examples of two *curvature identical* curves and their corresponding spherical images (schematically depicted), are given in Fig. 1, Fig. 2, Fig. 3 and Fig. 4.

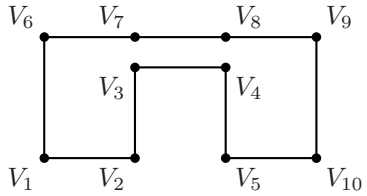


Fig. 1. Ten–points data set. Curve 1

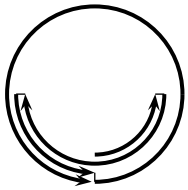


Fig. 2. A spherical image of Curve one

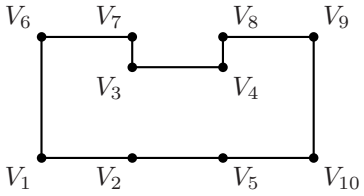


Fig. 3. Ten–points data set. Curve 2

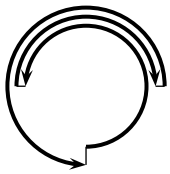


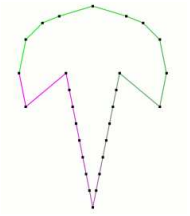
Fig. 4. A spherical image of Curve two

3. Short overview of research

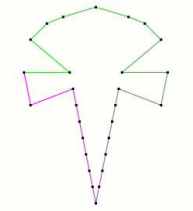
In order to construct 'reasonable' curves we consider the following criteria:

(i) minimisation of total absolute curvature;
(ii) minimisation of the length of a curve;
(iii) minimisation of the average curvature of a curve, *i.e.*, $|\omega|/\sum l_{ij}$;
(iv) minimisation of the total absolute curvature of a curve with only one 'deviation';
(v) minimisation of various weighted curvatures, such as $\sum_i(|\alpha(v_i)|(l_{(i-1)i} + l_{i(i+1)})$, $\sum_i |\alpha(v_i)|l_{(i-1)i}l_{i(i+1)}$ and some other.

Several algorithms are designed in order to obtain optimal curves with respect to these criteria. The objective of this research is to study how various criteria influence the shape of a curve. Therefore, we experimented with algorithms as well, at this moment we are not really interested in their costs. For certain criteria more than one algorithm is designed, using local or global approaches. See Fig. 5 for examples of outputs of two different algorithms to construct a curve of minimal total absolute curvature. We tested the algorithms on many data sets, among them data from curves with sharp corners, and onions with several layers when the minimum distance between the layers is greater than a certain value ϵ. In order to compare curves obtained by different criteria various geometric shape parameters are computed. The research is still in its development. One of the goals is to optimise some of obtained algorithms. All details and illustrative examples are given in the full version of the paper [1].



(i)



(ii)

Fig. 5. Outputs of two different algorithms for the same input data. The curve in (i) is the curve of minimal total absolute curvatue

References

[1] L. Alboul, G. Echeverria, and M. Rodrigues, Curvature criteria to fit curves to discrete data. Manuscript in preparation.

[2] L. Alboul, Optimising Triangulated Polyhedral Surfaces with Self–intersections, in: N. J. Wilson and R. R. Martin, eds., *Mathematics of Surfaces*, LNCS 2768, (2003), 48–73.

[3] L. Alboul and R. van Damme: Polyhedral metrics in surface reconstruction. In: Mullineux, G. (ed.), *Mathematics of Surfaces VI*, pp. 171–200, Oxford, 1996.

[4] N. Amenta, M. Bern, D. Eppstein, The Crust and the b–Skeleton: Combinatorial Curve Reconstruction, *Graphical Models and Image Processing* **60** (1998), 125–135.

[5] E. M. Arkin, S. P. Fekete, F. Hurtado, J. S. B. Mitchell, M. Noy, V. Sacristán, S. Sethia, On the Reflexifity of Point Sets, in: B. Aronov, S. Basu, J. Pach, and M. Sharir, eds., *Discrete and Computational Geometry*, (Springer, Berlin, 2003) 139–156.

[6] T. K. Dey and R. Wenger, Reconstructiong Curves with Sharp Corners, *Comp. Geometry and Appl.*, **19**(2–3) (2001), 89–99.

[7] N. H. Kuiper, Minimal total absolute curvature for immersions, *Invent. math*, **10** (1970), 209–238.

[8] Coaxing a planar curve to comply, *J. of Comp. and Appl. Math*, **140**, (2002), 599–618

[9] A. C. M. van Rooij, The Total Curvature of Curves, *Duke Math.*, **32** (1965), 313–324.

Minimum number of different distances defined by a finite number of points

Albujer, A. [a] and Segura Gomis, S. [a]

[a] *Departamento de Análisis Matemático, Universidad de Alicante, Campus de San Vicente del Raspeig, E-03080-Alicante, Spain*

Abstract

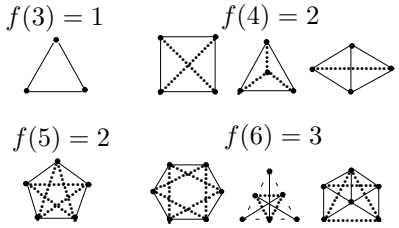
We study the minimum number of different distances defined by a finite number of points in the following cases: a) we consider metrics different from the euclidean distance in the plane, b) we consider the euclidean distance but restricted to subsets of the plane of special interest, c) we consider other topological surfaces: the cylinder and the flat torus. All these results extend those obtained by Erdös and other mathematicians for the euclidean distance in the plane.

Key words: distances, metrics, integer lattice, flat torus, cylinder

1. Introduction

In 1946 [4] Erdös posed the following problem: let $f(n)$ denote the minimum number of distinct distances that can occur among the $\frac{n(n-1)}{2}$ distances between n distinct points in the plane; what can we know about $f(n)$?

For small values of n it is easy to compute $f(n)$ and many times the number $f(n)$ is attained for different configurations. Let us see the first examples.



Erdös obtained asymptotic estimates for $f(n)$. The first asymptotic estimate was

$$cn^{1/2} < f(n) < c\frac{n}{(\ln n)^{1/2}}$$

Erdös conjectured that $f(n) > cn^{1-\varepsilon}$ for each $\varepsilon > 0$ and offered 500$ for a proof or a disproof.

Erdös conjecture is still open. The last improvement was made by Szemerédi (1992) who proved that $f(n) > cn^{4/5}$ ([2]).

$f(n)$ can be investigated in dimension $\mathbb{R}^d$ with $d > 2$. For $d = 3$ the best bounds are $cn^{4/3}\log\log n < f(n)$ ([5]) and $f(n) < n^{3/2+o(1)}$ ([3]).

Recently Braß([1]) determined $f(n)$ exactly in dimension $d = 4$.

There are also bounds for general dimensions.

The aim of this communication is to extend Erdös problem to other ambient spaces.

a) First we consider metrics different from the euclidean distance in the plane.
b) Then we consider also the euclidean distance but restricted to subsets of the plane of special interest (points of the integer lattice and points of rational coordinates).
c) Finally we consider Erdös problem in other topological spaces.

Email addresses: alab@alu.ua.es (Albujer, A.), Salvador.Segura@ua.es (Segura Gomis, S.).

2. Erdös problem considering arbitrary distances in the plane

In the plane we can define many distances. Let us recall that a distance in the plane is a map

$$d : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$$

such that

i) $d(x, y) \geq 0$, $d(x, y) = 0$ iff $x = y$
ii) $d(x, y) = d(y, x)$
iii) $d(x, z) \leq d(x, y) + d(y, z)$

An interesting way to define a metric in the plane is as follows.

Let us define the gauge function $g(K, x)$ of a closed, convex set K relative to an origin 0 as

$$g(K, x) = \inf\{\lambda : x \in \lambda K, \lambda > 0\}$$

It is easy to prove that if K is a proper convex body and $0 \in intK$, the function $m(x, y) = g(K, x - y)$ almost defines a distance in the plane. The condition that is violated is the symmetry condition $m(x, y) = m(y, x)$ that holds only if K is centrally symmetric.

So a closed, centrally symmetric, convex set K with $0 \in intK$ defines a metric ([7]).

All these metrics generate the euclidean topology.

In the particular case that the set K that induce this metric is a square of the type $\langle(p, 0), (0, p), (-p, 0), (0, -p)\rangle$, then we obtain the famous taxi-cab metric which is also known as the Manhattan metric. This metric is particularly relevant for our analysis.

The taxi-cab metric can also be defined as

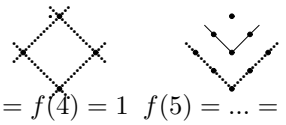
$$d_T((x_1, y_1), (x_2, y_2)) = |x_2 - x_1| + |y_2 - y_1|$$

We are going first to estimate the minimum number of distinct distances with the taxi-cab metric.

We begin showing in table 1 that the taxi-cab metric provides different values for $f(n)$ than those values attained with the euclidean metric.

$f(n)$	Euclidean metric	Taxi-cab metric
3	1	1
4	2	1
5	2	2
6	3	2
7	3	2

Table 1
Some values of $f(n)$ with the taxi-cab metric



$f(3) = f(4) = 1 \quad f(5) = ... = f(9) = 3$

Now we are going to estimate precisely $f(n)$.

Lemma 1 *Considering the taxi-cab metric*

$$f(n) \leq \lceil \sqrt{n} \rceil - 1$$

PROOF. Let L be the lattice generated by the vectors (1,1) and (1,-1). Let p be an integer number and let K_p be the point set determined by $K_p = L \cap \langle(0, 0), (p-1, -(p-1)), (2(p-1), 0), (p-1, p-1)\rangle$.

Consider $K_{\lceil \sqrt{n} \rceil}$. This set contains at least n points since $\lceil \sqrt{n} \rceil \cdot \lceil \sqrt{n} \rceil \geq \sqrt{n} \cdot \sqrt{n} = n$, so we can always choose n points in $K_{\lceil \sqrt{n} \rceil}$. As the points in $K_{\lceil \sqrt{n} \rceil}$ determine $\lceil \sqrt{n} \rceil - 1$ different distances among them, $f(n) \leq \lceil \sqrt{n} \rceil - 1$

□

This estimate is not only an upper bound but a precise determination of the minimum number of distinct distances:

Lemma 2 *Considering the taxi-cab metric*

$$f(n) \geq \lceil \sqrt{n} \rceil - 1$$

PROOF. We are going to give a sketch of the proof:

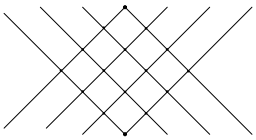
We consider the "metric circumferences", which in our case are square-shaped, as the locus of the points that are at distance r from a point $p \in \mathbb{R}^2$:

$$S(p, r) = \{x \in \mathbb{R}^2 : d_T(p, x) = r\}$$

Now we are going to locate the maximum number of points who define at most m distances among

them. As we are always dealing with a finite number of points, there would be at least two points a, b such that the distance between them is the maximum possible. All the others points should be located in the intersection of m "metric circumferences" centered at a and m "metric circumferences" centered at b. If we do not want to increase the number of different distances, the radius of these "metric circumferences" are $\{i\frac{d(a,b)}{m}\}_{i=1}^{m}$. There are two possible situations:

i) All the "metrics circumferences" intersect properly. In this case as they have the pseudodisc property (there are at most two proper intersections for each pair of pseudodiscs), then we obtain that the maximum number of possible points is given by $(m+1)^2$ and they are distributed in a translated of the square K_{m+1}. From this configuration we obtain our lower bound.



ii) There are some pairs of "metric circumferences" that do not intersect properly but are tangent. In this case they are tangent along a straight line segment. Considering the ends of this straight line segment and taking the "metric circumferences" centered at these ends we conclude that there are no more than $(m+1)^2$ possible locations. So we also reach the same lower estimate.

□

It is easy to see that the argument in the proof of lemma 2 holds for all metrics generated by the gauge function of a centrally symmetric, convex set; as the equality sign is attained in the case that the body which generates the metric is a square, we can conclude the following corollary:

Corollary 3 *Among all metrics generated by a closed, centrally symmetric, convex set K, the metrics that give the smallest values for $f(n)$ are those in which the metric is generated by the gauge function corresponding to a square.*

For instance the taxi-cab metric or the maximum metric defined as

$$d((x_1,y_1)(x_2,y_2)) = \max\{|x_2-x_1|,|y_2-y_1|\}$$

Corollary 3 does not hold for general metrics. For example, let us consider the so called post office metric

$$d((x_1,y_1)(x_2,y_2)) = d_2((x_1,y_1),(0,0)) + d_2((0,0),(x_2,y_2))$$

where d_2 stands for the euclidean distance in the plane. In this case it is easy to see that we can arrange infinite points so that each pair of them is at distance one; so $f(n)=1$ for all n.

3. The minimum number of distinct distances problem with the euclidean distance in the integer lattice

If we consider Erdös problem restricted to the integer lattice we obtained different values for $f(n)$ as the following table shows:

$f(n)$	Plane	Integer lattice
3	1	2
4	2	2
5	2	3
6	3	4
7	3	4

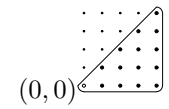
Table 2
Some values of $f(n)$ in the integer lattice

In general, we have the following upper bound:
Lemma 4 *In the integer lattice*

$$f(n) \leq \frac{(\lceil\sqrt{n}\rceil-2)(\lceil\sqrt{n}\rceil+1)}{2}$$

PROOF. Let L be the integer lattice and let P_n be the point set $P_n = L \cap \langle(0,0),(0,\lceil\sqrt{n}\rceil-1),(\lceil\sqrt{n}\rceil-1,0),(\lceil\sqrt{n}\rceil-1,\lceil\sqrt{n}\rceil-1)\rangle$. P_n contains at least n points.We can compute the number of distinct distances by considering only the distances between $(0,0)$ and the the rest of the points below the diagonal of the square. Then we have at most $2+3+...+(\lceil\sqrt{n}\rceil-1)$ distinct distances,

that is the sum of the terms of an arithmetic progression, so $f(n) \geq 2 + 3 + ... + (\lceil\sqrt{n}\rceil - 1) = \frac{\lceil\sqrt{n}\rceil(\lceil\sqrt{n}\rceil-1)}{2} - 1 = \frac{(\lceil\sqrt{n}\rceil-2)(\lceil\sqrt{n}\rceil+1)}{2}$



□

The values of $f(n)$ if we restrict to the integer lattice are the same that if we restrict to the points with rational coordinates because a finite number of points with rational coordinates are included in an integer lattice generated by the vectors $(q, 0)$,$(0, q)$ where q is the least common denominator of the coordinates of the points that we are considering.

This estimate of $f(n)$ can be applied to computers, because the computer screen can be represented as a finite number of points with rational coordinates.

4. Erdös problem for other topological spaces

The first mathematicians to consider this problem in other topological spaces were Erdös, Hickerson and Pach ([6]) who studied the case of the sphere.

We are going to consider two other particular topological surfaces: the cylinder and the flat torus.

As we can see in table 3, for small values of n we obtain the same values for $f(n)$ in both topological surfaces, but this do not occur for greater values of n.

$f(n)$	Plane	Cylinder	Flat torus
3	1	1	1
4	2	1	1
5	2	2	2
6	3	2	2

Table 3
Some values of $f(n)$ in different topological surfaces

Now, we can give upper bounds for both topological surfaces.

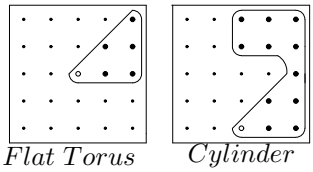
Lemma 5 *In the flat torus*

$$f(n) \leq \frac{(\lfloor\lceil\sqrt{n}\rceil/2+1\rfloor+2)(\lfloor\lceil\sqrt{n}\rceil/2+1\rfloor-1)}{2}$$

Lemma 6 *In the cylinder*

$$f(n) \leq \frac{(\lfloor\lceil\sqrt{n}\rceil/2+1\rfloor+2)(\lfloor\lceil\sqrt{n}\rceil/2+1\rfloor-1)}{2}$$
$$+(\lfloor\lceil\sqrt{n}\rceil/2+1\rfloor)(\lceil\sqrt{n}\rceil - \lfloor\lceil\sqrt{n}\rceil/2+1\rfloor)$$

We omit the proofs because they are similar to the proof of lemma 3. We have to consider a particular lattice and intersect it with the topological surfaces, as the figure shows. Then, by some arithmetic computations, we obtain the bounds.



References

[1] Braß, P.: On the maximum number of unit distances among n points in dimension four, *Intuitive geometry* (Budapest, 1995), 277–290, Bolyai Soc. Math. Stud., 6, János Bolyai Math. Soc., Budapest, 1997.

[2] Chung, F. R. K., Szemerdi, E. and Trotter, W. T.: The number of different distances determined by a set of points in the Euclidean plane, *Discrete Comp. Geometry* **7** (1992), 1–11.

[3] Clarkson, K., Edelsbrunner, H., Guibas, L., Sharir, M. and Welzl, E.:Combinatorial complexity bounds for arrangements of curves and spheres, *Discrete Comp. Geometry* **5** (1990), 90–160.

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

[5] Erdös, P.:On sets of distances of n points in Euclidean space, *Magyar Tudományos Akadémia Mátematikai Kutató Intézet Közleményei* **7** (1960), 165–169.

[6] Erdös, P. , Hickerson, D. and Pach J.: A problem of Leo Moser about repeated distances on the sphere, *Amer. Math. Monthly* **96** (1989), 569–575.

[7] Guggenheimer, H. W.: *Applicable geometry*, Robert E. Krieger Publishing Co., Inc. , Huntington, New York, 1977.

Computing the Hausdorff Distance Between Curved Objects [1]

Helmut Alt Ludmila Scharf
Institute for Computer Science, Freie Universität Berlin, Takustr.9, D-14195 Berlin

Key words: shape comparison, Hausdorff distance, parametric curves

1. Introduction

Analysis and comparison of geometric shapes are of importance in various application areas within computer science, such as pattern recognition and computer vision, but also in other disciplines concerned with the form of objects such as cartography, molecular biology, medicine, or biometric signal processing.

The general situation is that we are given two objects A, B modelled as subsets of 2- or 3-dimensional space and we want to know how much they resemble each other [1].

For this purpose we need a similarity measure defined on pairs of shapes indicating the degree of resemblance of these shapes. A frequently used similarity measure is the Hausdorff distance, which is defined for arbitrary non-empty compact sets A and B. It assigns to each point of one set the distance to its closest point in the other and takes the maximum of all these values. Formally, we define the *one-sided Hausdorff distance* from A to B as

$$\vec{\delta}_H(A,B) = \max_{a\in A} \min_{b\in B} d(a,b), \qquad (1)$$

where $d(x,y)$ denotes a distance measure between points x and y.

Here we will assume the planar case, i.e., that $A, B \subset \mathbb{R}^2$ and that d is the Euclidean distance.

The (bidirectional) *Hausdorff distance* between A and B is defined as

$$\delta_H(A,B) = \max(\vec{\delta}_H(A,B), \vec{\delta}_H(B,A)) \qquad (2)$$

We will only describe the computation of the one-sided Hausdorff distance from A to B, the computation of the bidirectional Hausdorff distance is then straightforward.

[1] This research was supported by the European Union under contract No. IST-2000-26473, Project ECG.

The aim of this paper is to find an algorithm for general shapes which are modelled by two sets of n *algebraic curves*. When we speak about the Hausdorff distance between these two sets we actually mean the Hausdorff distance between the two sets of points lying on these curves. We will restrict to curves that are given by *rational parameterizations*, i.e., each curve is represented by a parameterization

$$c : I \to \mathbb{R}^2, \quad c(t) = (x(t),\ y(t)) \qquad (3)$$

where $I \subset \mathbb{R}$ is a closed interval and $x(t), y(t)$ are rational functions with no poles in I.

Observe that this definition includes some important families of free-form parametric curves, for example B-splines.

For simplicity, we assume a certain general position of the input curves. In particular, we assume that any two curves intersect in at most finitely many points.

2. Basic cases

In this section we will investigate how the directed Hausdorff distance between two single objects (curves or points) can be computed.

2.1. *Point-curve and curve-point*

The Hausdorff distance from a point $p = (u,v)$ to a curve $c(t) = (x(t), y(t)), t \in I$ is $\min_{t\in I} d(p, c(t))$.

The Hausdorff distance from a curve $c(t)$ to a point p is the maximum Euclidean distance from any point on c to p.

In order to find those parameters t where the minimum or maximum is attained, we consider the

zeroes of the derivative of the squared distance $\frac{\mathrm{d}}{\mathrm{d}\,t}[d^2(p, c(t))]$, i.e., the equation

$$2 \cdot (u - x(t)) \cdot x'(t) + 2 \cdot (v - y(t)) \cdot y'(t) = 0 \quad (4)$$

This equation has constantly many solutions if the degree of c is bounded. We call a point satisfying this equation a *footpoint* of p on c. In addition to the points given by the solutions of equation 4 the minimum or maximum distance can be attained at the endpoints of c (Fig. 1).

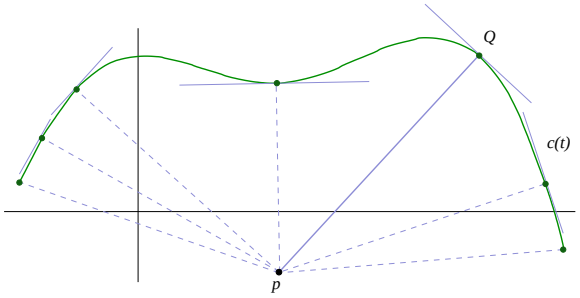


Fig. 1. Hausdorff distance from a curve $c(t)$ to a point p is the distance from p to the farthest point on c.

2.2. *Curve-curve*

We reduce the problem of determining the Hausdorff distance from a curve a, $a(t) = (x_a(t), y_a(t))$, $t \in I_a$ to a curve b, $b(s) = (x_b(s), y_b(s))$, $s \in I_b$ to determining the distances of constantly many candidate points on a to the curve b.

There are four different types of candidate points. Firstly, the Hausdorff distance can be assumed at one of the endpoints of curve a (*type EA*).

Secondly, it can happen that the Hausdorff distance is attained between an endpoint Q of b and a point on a. Therefore, we determine on a all footpoints of the endpoints of b by equation (4) (*type EB*).

For the third type of candidate points we consider the *self-bisector* (or *medial axis*) of a curve, which is the set of all points whose minimal distance to the curve is attained at more than one point on the curve, cf. Fig. 2.

The Hausdorff distance from a to b can be attained at an intersection point of a with the self-bisector of b. We will only give a system of equations describing such a point here for the part of the self-bisector where the two closest points are interior points of b, see Fig. 3.

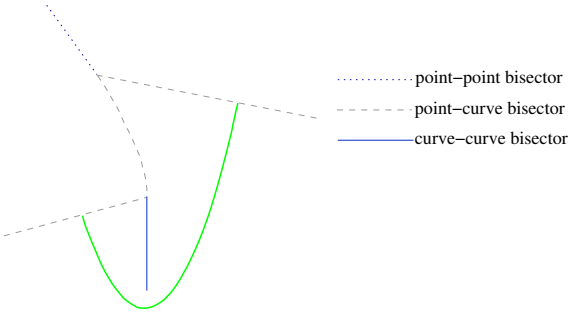


Fig. 2. Self-bisector of a parabola segment.

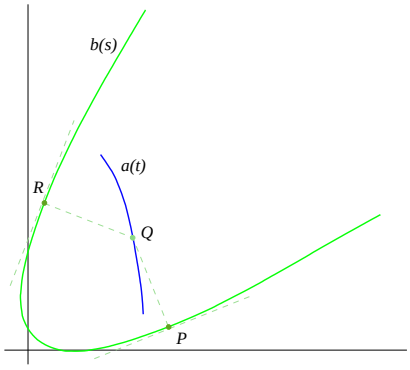


Fig. 3. The Hausdorff distance from the curve $a(t)$ to the curve $b(s)$ is assumed at the intersection point Q of the curve a with the self-bisector of the curve b. The point Q has two different internal foot-points P and R on b.

Suppose $Q = a(t)$ and that $P = b(s)$ and $R = b(r)$ with $r \neq s$ are the points on b closest to Q. Then we obtain the following system of equations for t, s, and r:

$$\begin{aligned}
&\left[(x_a(t) - x_b(s))^2 + (y_a(t) - y_b(s))^2\right] - \\
&\qquad \left[(x_a(t) - x_b(r))^2 + (y_a(t) - y_b(r))^2\right] = 0 \quad (5)\\
&(x_a(t) - x_b(s)) \cdot x_b'(s) + (y_a(t) - y_b(s)) \cdot y_b'(s) = 0 \quad (6)\\
&(x_a(t) - x_b(r)) \cdot x_b'(r) + (y_a(t) - y_b(r)) \cdot y_b'(r) = 0 \quad (7)
\end{aligned}$$

In order to enforce the condition $r \neq s$ we introduce a new variable u and add one more equation to the system:

$$1 - u(s - r) = 0 \quad (8)$$

We add all points determined by the finitely many solutions of this system to the candidate list (*type SB*).

For the fourth type of possible candidate points we observe that the Hausdorff distance can occur between two interior points $p \in a$ and $q \in b$ with-

out one of them lying on the self-bisector of the other curve, see Fig. 4.

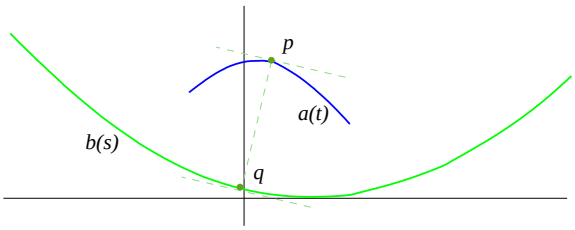


Fig. 4. Hausdorff distance at interior points.

Since q is a footpoint on b for point p, the line segment $\overline{pq}$ must be perpendicular to the tangent line to curve b at point q. In addition it can be shown that the tangent line to the curve a at point p must be parallel to the tangent line to the curve b at point q and, thus, also perpendicular to the line segment $\overline{pq}$. The remaining candidate points (*type I*) for the Hausdorff distance are, therefore, among the solutions of the following system:

$$(x_a(t) - x_b(s)) \cdot x_b'(s) + (y_a(t) - y_b(s)) \cdot y_b'(s) = 0 \qquad (9)$$

$$(x_a(t) - x_b(s)) \cdot x_a'(t) + (y_a(t) - y_b(s)) \cdot y_a'(t) = 0 \qquad (10)$$

A detailed geometric proof for this fact is omitted due to space limitations and can be found in [7].

3. The general case

As was said before, the general problem we consider is to find the Hausdorff distance between two point sets given by two sets A, B of rationally parameterized algebraic curves.

In order to find $\tilde{\delta}(A, B)$ we split the curves of A at their intersection points with the Voronoi diagram of B. The resulting set A' of curves has the property that each curve lies in one Voronoi cell of B, so its distance to B is the distance to one curve and can be determined using the techniques of section 2.

Computing the complete Voronoi diagram of algebraic curves is a very difficult task in practice and there are some open questions about the bisector of algebraic curves [3–5]. In [5] an approximation algorithm for Voronoi diagrams of curves is given.

Instead, we just compute the intersection points described. The splitting of each curve a of A is done incrementally. Suppose that we have list of intersection points of the Voronoi diagram of a subset of B with a and we want to add a new curve $b \in B$. We do this by scanning the current segments of a. Each segment s belongs to some curve $c \in B$ which has already been processed. First we determine all intersection points of the bisector between b and c with a and find out which ones lie inside s. s is split further with these points and some portions are labelled with b as nearest neighbor, others with c. After this has been done it might be necessary to merge neighboring segments which are both marked with b but are separated by a split-point from previous steps.

Similar to the case of self-bisectors, the intersection points of with the bisector between b and c can be found as solutions of systems of equations. We only give this system here for the "interior" bisector of two curves b and c, not the ones for the bisector between an endpoint of one curve and another curve or between two endpoints. These systems, however have to be considered, as well.

The system for the "interior" bisector uses the property that if a point $a(t)$ lies on the bisector between b and c and the corresponding footpoints are $b(s)$ and $c(r)$ then the line segment $\overline{a(t)b(s)}$ is perpendicular to the tangent vector $b'(s)$, and $\overline{a(t)c(r)}$ is perpendicular to $c'(r)$.

$$d(a(t), b(s)) - d(a(t), c(r)) = 0 \qquad (11)$$

$$\langle (a(t) - b(s)), b'(s) \rangle = 0 \qquad (12)$$

$$\langle (a(t) - c(r)), c'(r) \rangle = 0 \qquad (13)$$

The worst case running time of our algorithm as stated is $O(nm^2)$ if A consists of n and B of m curves, assuming that the degrees of the curves are bounded. It can be easily improved to $O(nm \log m)$ by using a divide-and-conquer approach for the splitting of the curves of A. It should be worthwhile to further improve the combinatorial complexity of the algorithm (see also [2]), but our major intent was to find a simple algorithm that can be put into practice with a reasonable amount of effort.

4. Implementation

We implemented the algorithm described in C++ using the computer algebra software library SYNAPS (SYmbolic Numeric APplicationS), see [8,6], for solving the systems of polynomial equations.

For visualization purposes a graphical user interface was developed using the GTK library. It includes visualization of the curves, the input and editing of the parameterization and the intervals if the parameter values, the computation of the Hausdorff distance and the graphical indication of the candidate points considered for the computation. Figure 5 shows the interface with an example.

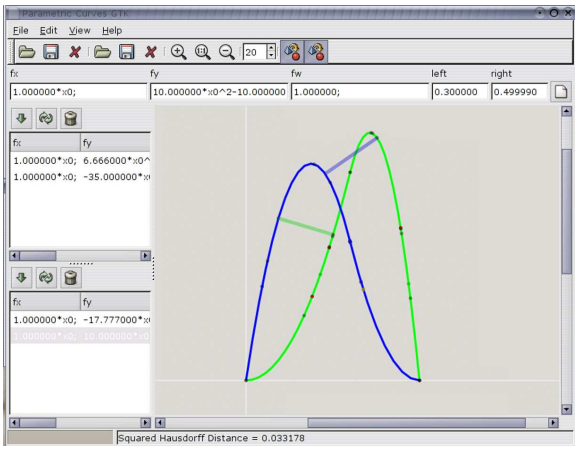


Fig. 5. Result of a Hausdorff distance computation with two B-splines of degree 2. Both one-way distances and all candidate points are shown.

References

[1] Helmut Alt and Leonidas J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In *Handbook of computational geometry*. Elsiever Science B.V., 1999.

[2] Helmut Alt and Otfried Schwarzkopf. The Voronoi diagram of curved objects. In *Proc. 11th ACM Comput. Geom. Symp.*, pages 89–97, Vancouver, BC, 1995.

[3] Gershon Elber and Myung-Soo Kim. Bisector curves of planar rational curves. *Computer-Aided Design*, 30(14):1089–1096, 1998.

[4] Rida T. Farouki and Rajesh Ramamurthy. Degenerate point/curve and curve/curve bisectors arising in medial axis computations for planar domains with curved boundaries. *Internat. J. Comput. Geom. Appl.*, 8:599–617, 1998.

[5] Rida T. Farouki and Rajesh Ramamurthy. Voronoi diagram and medial axis algorithm for planar domains with curved boundaries, I. Theoretical foundations. *Journal of Computational and Applied Mathematics*, 102(1):119–141, February 1999.

[6] G. Dos Reis, B. Mourrain, R. Rouillier, and Ph. Trébuchet. An environment for symbolic and numeric computation. In *Proc. of the International Conference on Mathematical Software*, pages 239–249, 2003.

[7] Ludmila Scharf. Computing the Hausdorff distance between sets of curves. Master's thesis, Freie Universität Berlin, Germany, 2003.

[8] http://www-sop.inria.fr/galaad/logiciels/synaps/.

A certified conflict locator for the incremental maintenance of the Delaunay graph of semi-algebraic sets

François Anton,
Department of Computer Science, University of British Columbia,
201-2366 Main Mall, Vancouver, B.C., Canada, V6T 1Z4

Abstract

Most of the curves and surfaces encountered in geometric modelling are defined as the set of solutions of a system of algebraic equations or inequalities (semi-algebraic sets). The Voronoi diagram of a set of sites is a decomposition of the space into proximal regions (one for each site). Voronoi diagrams have been used to answer proximity queries. The dual graph of the Voronoi diagram is called the Delaunay graph. Only approximations by conics can guarantee a proper continuity of the first order derivative at contact points, which is necessary for guaranteeing the exactness of the Delaunay graph. The central idea of this paper is that a (one time) symbolic preprocessing may accelerate the certified numerical evaluation of the Delaunay graph conflict locator. The symbolic preprocessing is the computation of the implicit equation of the generalised offset to conics. The certified computation of the Delaunay graph conflict locator relies on theorems on the uniqueness of a root in given intervals (Kantorovich, Moore-Krawczyk). For conics, the computations get much faster by considering only the implicit equations of the generalised offsets.

Key words: Delaunay graph, semi-algebraic sets, conics, conflict locator

1. Introduction

Most of the curves and surfaces encountered in geometric modelling are defined as the set of common zeroes of a set of polynomials (*algebraic varieties*) or subsets of algebraic varieties defined by one or more algebraic inequalities (*semi-algebraic sets*). Many problems from different fields involve proximity queries like finding the nearest neighbour, finding all the neighbours, or quantifying the neighbourliness of two objects. The retraction planning [ÓY85] problem (that addresses the optimal trajectory of a robot around obstacles) in robotics and spatial analysis and influence zones in Geographic Information Systems [VC90] are strongly linked to questions of proximity among real-world objects in real-world environments.

The Voronoi diagram [Vor08] (see Fig. 1) of a set of sites is a decomposition of the space into proximal regions (one for each site). The proximal region (or Voronoi zone) of a site is the locus of points closer to that site than to any other one. Voronoi diagrams allow one to answer proximity queries after a query point has been located in the Voronoi zone it belongs to. The Voronoi diagram defines a neighbourhood relationship among sites: two sites are neighbours if, and only if, their Voronoi regions are adjacent. The graph of this neighbourhood relationship is called the Delaunay graph. The Delaunay graph of sites in the plane satisfies the following empty circle criterion (see Fig. 2): no site intersects the interior of the circles touching (tangent to without intersecting the interior of) the sites that are the vertices of any triangle of the Delaunay graph (see Fig. 3). There have been attempts [OBS92] to compute Voronoi diagrams of curves by approximating curves by line segments or circular arcs, but the exactness of the Delaunay graph is not guaranteed [RF99a]. Indeed, the Voronoi diagram is very sensitive to the order of continuity at contact points (see [RF99a]). Only approxima-

Email address: anton@medicis.polytechnique.fr (François Anton).

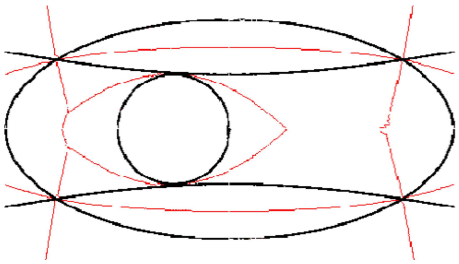


Fig. 1. The Voronoi diagram (light) of a circle, an ellipse and a hyperbola (dark)

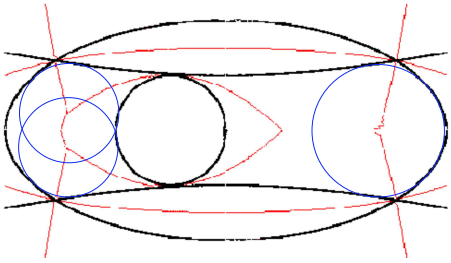


Fig. 2. The 3 empty circles for the sites of Fig. 1

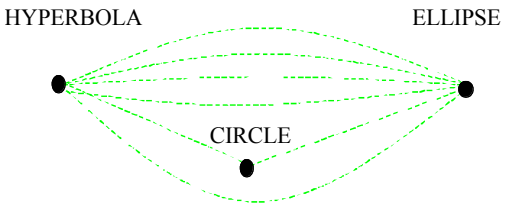


Fig. 3. The Delaunay graph of the sites of Fig. 1

tions by conics can guarantee a proper continuity of the first order derivative at contact points, which is necessary for guaranteeing the exactness of the Delaunay graph [RF99a]. Other approximation algorithms have used a Newton-Raphson scheme to compute and classify Voronoi vertices for curves with a rational parameterisation [RF99a,RF99b]. These do not directly address the exactness of the Delaunay graph.

2. Preliminaries

We will recall now the formal definitions of the Voronoi diagram and of the Delaunay graph. For this purpose, we need to recall some basic definitions.

Definition 1 *(Metric) Let M be an arbitrary set. A metric on M is a mapping $d : M \times M \rightarrow \mathbb{R}_+$ such that for any elements a, b, and c of M, the following conditions are fulfilled: $d(a,b) = 0 \Leftrightarrow a = b$, $d(a,b) = d(b,a)$, and $d(a,c) \leq d(a,b) + d(b,c)$. (M,d) is then called a metric space, and $d(a,b)$ is the distance between a and b.*

Let $M = \mathbb{R}^N$, and δ denote the Euclidean distance between points. Let $\mathcal{S} = \{s_1, ..., s_m\} \subset M, m \geq 2$ be a set of m different subsets of M, which we call *sites*. The distance between a point x and a site $s_i \subset M$ is defined as $d(x, s_i) = \inf_{y \in s_i} \{\delta(x,y)\}$.

Definition 2 *(Influence zone) For $s_i, s_j \in \mathcal{S}, s_i \neq s_j$, the influence zone $D(s_i, s_j)$ of s_i with respect to s_j is: $D(s_i, s_j) = \{x \in M | d(x, s_i) < d(x, s_j)\}$.*

Definition 3 *(Voronoi region) The Voronoi region $V(s_i, \mathcal{S})$ of $s_i \in \mathcal{S}$ with respect to the set $\mathcal{S}$ is: $V(s_i, \mathcal{S}) = \bigcap_{s_j \in \mathcal{S}, s_j \neq s_i} D(s_i, s_j)$.*

Definition 4 *(Voronoi diagram) The Voronoi diagram of $\mathcal{S}$ is the union $V(\mathcal{S}) = \bigcup_{s_i \in \mathcal{S}} \partial V(s_i, \mathcal{S})$ of all region boundaries.*

Definition 5 *(Delaunay graph) The Delaunay graph $DG(\mathcal{S})$ of $\mathcal{S}$ is the dual graph of $V(\mathcal{S})$ defined as follows:*

- *the set of vertices of $DG(\mathcal{S})$ is $\mathcal{S}$,*
- *for each $N-1-$dimensional facet of $V(\mathcal{S})$ that belongs to the common boundary of $V(s_i, \mathcal{S})$ and of $V(s_j, \mathcal{S})$ with $s_i, s_j \in \mathcal{S}$ and $s_i \neq s_j$, there is an edge of $DG(\mathcal{S})$ between s_i and s_j and reciprocally, and*
- *for each vertex of $V(\mathcal{S})$ that belongs to the common boundary of $V(s_{i_1}, \mathcal{S}), \ldots, V(s_{i_{N+2}}, \mathcal{S})$, with $\forall k \in \{1, ..., N+2\}, s_{i_k} \in \mathcal{S}$ all distinct, there exists a complete graph K_{N+2} between the $s_{i_k}, k \in \{1, ..., N+2\}$, and reciprocally. (see example on Fig. 3).*

Let us introduce the generalised offset and the generalised Voronoi vertex. We place ourselves in the affine space K^2 where $K = \mathbb{C}$ for the sake of introducing those notions in an easier way. While the $R-$generalised offset to ν is the locus of the centres of circles of radius R that are tangent to ν, the true $R-$offset to ν is the locus of the centres of circles of radius R that are tangent to ν and do not contain any point of ν in its interior (see Fig. 4).

Definition 6 *(generalised Voronoi vertex) A generalised Voronoi vertex of three semi-algebraic sets S_1, S_2, and S_3 is a point of intersection of the $R-$generalised offsets of S_1, S_2, and S_3 (see Exam-*

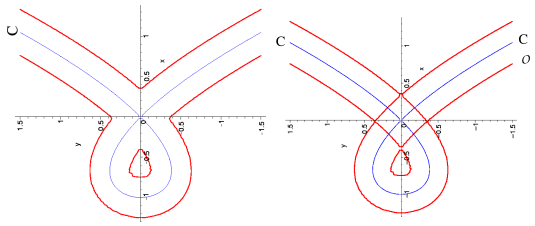


Fig. 4. The strophoid and its true (left) and generalised (right) offsets

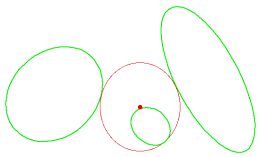


Fig. 5. A generalised Voronoi vertex (dot) of three conics (thick lines)

ple on Fig. 5).

3. The Delaunay graph conflict locator for semi-algebraic sets

Let $X_1, ..., X_{N+2}$, be semi-algebraic sets [BR90,BCR98]. A semi-algebraic set X_i is defined as: $\bigcup_{j=1}^{s_i} \bigcap_{k=1}^{r_{i,j}} \left\{x \in \mathbb{R}^N | f_{i,j,k} \star_{i,j,k} 0\right\}$, where $f_{i,j,k}$ is a polynomial with real coefficients in the variables $x_{i_1}, ..., x_{i_N}$ and $\star_{i,j,k}$ is either $<$ or $=$, for $i = 1, 2, 3, 4$, $j = 1, ..., s_i$ and $k = 1, ..., r_{i,j}$. The Delaunay graph conflict locator determines which ones of the maximal dimensional facets of the Delaunay graph of $N+1$ semi-algebraic sets $X_1, ..., X_{N+1}$ would be changed by the addition of the semi-algebraic set X_{N+2}.

Let us assume without loose of generality that each $\bigcap_{k=1}^{r_{i,j}} \left\{x \in \mathbb{R}^N | f_{i,j,k} \star_{i,j,k} 0\right\}$ for each X_i is defined by at least one non-trivial algebraic equation (i.e. different from the zero polynomial). If our starting assumption is not valid in the case we treat, we can make it valid by adding the equations corresponding to $f_{i,j,k} = 0$ for each (i, j, k) such that j is the index of a component that is not defined as in the assumption and i is the index of the semi-algebraic set to which the component belongs. Let us denote V_i as the intersection of all the $V(f_{i,j,k})$ such that $\star_{i,j,k}$ is $=$ for each $i = 1, 2, 3, 4$. Let $\mathcal{N}_i$ be the normal space to V_i at the point $x_i = (x_{i_1}, ..., x_{i_N})$. Each $f_{i,j,k}$ defining V_i induces $N-1$ polynomials $n_{i,j,k,l}$ with $l = 1, ..., N-1$ that are the equations defining the normal to $V(f_{i,j,k})$ at x_i. A point $q = (y_1, ..., y_N)$ belongs to $\mathcal{N}_i$ if its coordinates satisfy all the equations of the normal spaces to $V(f_{i,j,k})$ at x_i such that $\star_{i,j,k}$ is $=$.

For a given $q = (y_1, ..., y_N)$, let $\mathcal{M}_i$ be the the set of points $m_i = (z_{i_1}, ..., z_{i_N}) \in X_i$ such that q belongs to the normal space to V_i at the point m_i. In the general case, each set $\mathcal{M}_i$ is a finite set of points. However, if V_i contains a portion of hypersphere $PHS(q, \rho)$ centered on q, then $\mathcal{M}_i$ contains that portion of hypersphere. To get in all cases a finite set of points m_i of V_i, we use $\mathfrak{S}_i = \mathcal{M}_i$ when $\mathcal{M}_i$ is finite, and $\mathfrak{S}_i \bigcap PHS(q, \rho) = \{w_i\}$ for an arbitrary point w_i of $PHS(q, \rho)$ when V_i contains a portion of hypersphere $PHS(q, \rho)$ centered on q.

We are now able to write the system of algebraic equations and inequalities that define the outcome of the Delaunay graph conflict locator. Let us consider the map $\pi : K^{3N} \rightarrow K^N$ defined by $\pi(x_i, q, m_i) = q$.

The point q is at the distance r from the point x_i if, and only if, the distance between q and x_i is r. This is expressed algebraically by the equation $d_i(q, x_i) = (y_1 - x_{i_1})^2 + ... + (y_N - x_{i_N})^2 - r^2 = 0$.

The generalised r-offset $\mathcal{O}_i$ to X_i is the image by π of the points of K^{3N} defined by the following system of equations and inequalities:

$$\begin{cases} \exists j \in [1, s_i], \forall k \in [1, r_{i,j}], \\ \quad \begin{cases} f_{i,j,k}(x_i) \star_{i,j,k} 0 \\ \text{if } \star_{i,j,k} \text{ is } \text{“}=\text{”}, \\ \quad \begin{cases} d_i(x_i, q) = 0 \\ \forall l = 1, .., N-1, n_{i,j,k,l}(x_i, q) = 0 \\ f_{x_{i_1}}(x_i) \neq 0 \text{ or } \ldots \text{ or } f_{x_{i_N}}(x_i) \neq 0 \end{cases} \end{cases} \end{cases}$$

The true r-offset to X_i is obtained as the difference of the generalised r-offset $\mathcal{O}_i$ to X_i and the union of each one of the images by π of the semi-algebraic sets defined by the following system of equations and inequalities for each point m_i of $\mathfrak{S}_i$:

$$\begin{cases} \exists j \in [1, s_i], \forall k \in [1, r_{i,j}], \\ \quad \begin{cases} f_{i,j,k}(m_i) \star_{i,j,k} 0 \\ \text{if } \star_{i,j,k} \text{ is } \text{“}=\text{”}, \\ \quad \begin{cases} f(m_i) = 0 \\ \forall l = 1, .., N-1, n_{i,j,k,l}(m_i, q) = 0 \\ d(m_i, q) < 0 \end{cases} \end{cases} \end{cases}$$

It is obvious that a true Voronoi vertex of

Running time	above systems	generalised offsets
General Solve	6 min 38 s	12 min 37 s
Gradient Solve	2 h 56 min 10 s	2 min 26 s
Hessian Solve	20 h 17 min 42 s	3 min 42 s

Table 1
Some running time results for ellipses

$X_1, ..., X_{N+1}$ is a point of intersection of the true $r-$offsets to $X_1, ..., X_{N+1}$ respectively. A true Voronoi vertex of $X_1, ..., X_{N+1}$ is at the distance R from X_{N+2}, or alternatively, a true Voronoi vertex of $X_1, ..., X_{N+1}$ belongs to the true $R-$offset to X_{N+2}. Consider the $N+2-$dimensional points whose first N coordinates are the coordinates of a true Voronoi vertex of $X_1, ..., X_{N+1}$, and the remaining two are the distances r between that true Voronoi vertex and $X_1, ..., X_{N+1}$, and R between that true Voronoi vertex and X_{N+2}. The Delaunay graph conflict locator should report all the true Voronoi vertices such as the corresponding $N+2-$dimensional points satisfy $R - r < 0$.

We have evaluated the Delaunay graph conflict locator without solving any intermediary system by using an interval analysis based library (ALIAS [Mer00]) for solving zero-dimensional systems of equations and inequalities. The certified computation of the Delaunay graph conflict locator relies on theorems on the uniqueness of a root in given intervals (Kantorovich and Moore-Krawczyk). This computation uses a bisection process on one or all the variables using either only the equations of the system, or using the Jacobian of the system (Moore-Krawczyk test for finding "exactly" the solutions), or using the Jacobian and the Hessian of the system (with Kantorovich, Moore-Krawczyk tests). We first used ALIAS on the above system of algebraic equations and inequalities that specify the Delaunay graph conflict locator for semi-algebraic sets. Then for conics, we used ALIAS on the system simplified by replacing the equations f_i, n_i and d_i of the conics, normals and distances between the points on the conics and the true Voronoi vertex by the implicit equations of the generalised offsets to the conics (see [Ant04]). This induces much faster computations (see Table 1).

4. Conclusions

We have presented what we believe is the first certified conflict locator for the incremental maintenance of the Delaunay graph for semi-algebraic sets. Further research will try to improve the running time of the computations.

References

[Ant04] François Anton. *Voronoi diagrams of semi-algebraic sets.* PhD thesis, University of British Columbia, January 2004.

[BCR98] Jacek Bochnak, Michel Coste, and Marie-Françoise Roy. *Real algebraic geometry.* Springer-Verlag, Berlin, 1998. Translated from the 1987 French original, Revised by the authors.

[BR90] Riccardo Benedetti and Jean-Jacques Risler. *Real algebraic and semi-algebraic sets.* Hermann, Paris, 1990.

[Mer00] Jean-Pierre Merlet. Alias: an interval analysis based library for solving and analyzing system of equations. In *SEA*, Toulouse, France, 14–16 June 2000.

[OBS92] Atsuyuki Okabe, Barry Boots, and Kōkichi Sugihara. *Spatial tessellations: concepts and applications of Voronoï diagrams.* John Wiley & Sons Ltd., Chichester, 1992. With a foreword by D. G. Kendall.

[ÓY85] Colm Ó'Dúnlaing and Chee-K. Yap. A "retraction" method for planning the motion of a disc. *J. Algorithms*, 6(1):104–111, 1985.

[RF99a] Rajesh Ramamurthy and Rida T. Farouki. Voronoi diagram and medial axis algorithm for planar domains with curved boundaries. I. Theoretical foundations. *J. Comput. Appl. Math.*, 102(1):119–141, 1999. Special issue: computational methods in computer graphics.

[RF99b] Rajesh Ramamurthy and Rida T. Farouki. Voronoi diagram and medial axis algorithm for planar domains with curved boundaries. II. Detailed algorithm description. *J. Comput. Appl. Math.*, 102(2):253–277, 1999.

[VC90] Christine Voiron-Canicio. *Analyse spatiale et analyse d'images par la morphologie mathématique.* RECLUS, 1990.

[Vor08] Georgiï Feodosevich Voronoï. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire. recherches sur les parallélоèdres primitifs. première partie. partition uniforme de l'espace analytique à n dimensions à l'aide des translations d'un même polyèdre convexe. *Journal für die reine und angewandte Mathematik*, 134:198–287, 1908.

3D realization of two triangulations of a convex polygon.

Sergey Bereg [a] ,

[a] *Department of Computer Science, University of Texas at Dallas, Box 830688, Richardson, TX 75083, USA.*

Abstract

We study the problem of construction of a convex 3-polytope whose (i) shadow boundary has n vertices and (ii) two hulls, upper and lower, are isomorphic to two given triangulations of a convex n-gon. Barnette [1] proved the existence of a convex 3-polytope in general case. We show that, in our case, a polytope can be constructed using an operation of edge creation.

Key words: triangulation, convex polytope, Steinitz theorem

1. Introduction

Let P be a convex polygon in the xy-plane with n vertices. Two triangulations of P are called *distinct* if the only edges they share are the edges of P. Let T_1 and T_2 be two distinct triangulations of P. At the *First Canadian Conference on Computational Geometry* Leo Guibas conjectured that it is always possible to perturb the vertices of P vertically out (i.e., by displacements parallel to the z-axis) so that the polygon P becomes a spatial polygon P' such that the convex hull of P' is a convex polyhedron consisting ot two triangulated cups glued along P', *and* the triangulation of the upper cup (i.e., those faces oriented toward $+z$) is that specified as T_1, and the triangulation of the lower cup is that specified as T_2 [4].

Boris Bekster [2] disproved Guibas' conjecture by showing a counterexample, a convex hexagon with two triangulations. Marlin and Toussaint [3] considered the computational problem of deciding whether a triple (P, T_1, T_2) admits a realization in $\mathbb{R}^3$. They reduced the problem to a linear programming problem with $O(n^2)$ inequality constraints and n variables. The variables are z-coordinates of lifted vertices of P and the constraints correpond to vertex-face relations: the vertices must be below/above the planes passing through faces of the upper/lower cup of P'. The number of constraints can be dropped to $2n-6 = |T_1|+|T_2|$ by considering dihedral angles corresponding to diagonals of the triangulations [7].

Guibas conjecture is related to Steinitz's theorem [5].

Steinitz's Theorem: *A graph G is isomorphic to the edge graph of a convex 3-polytope if and only if G is 3-connected and planar.*

By Steinitz's theorem the graph $(P, T_1 \cup T_2)$ is the edge graph of a convex 3-polytope [3]. According to Barnette's theorem [1], every 3-polytope with a Hamiltonian circuit has realization such that the Hamiltonian circuit is a shadow boundary. This implies that Guibas' conjecture is true up to a combinatorial deformation [2]. Formally this can be stated as follows.

Theorem 1 *For any two distinct triangulations T_1 and T_2 of a convex polygon P_2 in $\mathbb{R}^2$ with n vertices, there is a convex polytope P_3 in $\mathbb{R}^3$ with n vertices such that* (i) *the xy-shadow S of P_3 contains all its vertices, and* (ii) *there is a isomorphism $\tau : P_2 \to S$ that maps the edges of T_1 (resp. T_2) to the edges of the upper hull of P_3 (resp. the lower hull).*

Barnette's proof deals with general faces (not just triangles) due to its generality. In this paper we give a different proof of Theorem 1 that uses only triangular faces of polytopes which can be turned into a more robust algorithm for finding a combinatorial realization of (P, T_1, T_3) in $\mathbb{R}^3$.

Realization questions have been studied in com-

Email address: besp@utdallas.edu (Sergey Bereg).

URL: http://utdallas.edu/~sxb027100 (Sergey Bereg).

puter graphics and scene analysis as well. Sugihara [6] established necessary and sufficient conditions whether a line drawing in the plane can be realized in $\mathbb{R}^3$ by lifting.

We call a triple (P, T_1, T_2) a *configuration*. We call a map τ satisfying the conditions of Theorem 1 a *realization*.

2. Edge contraction

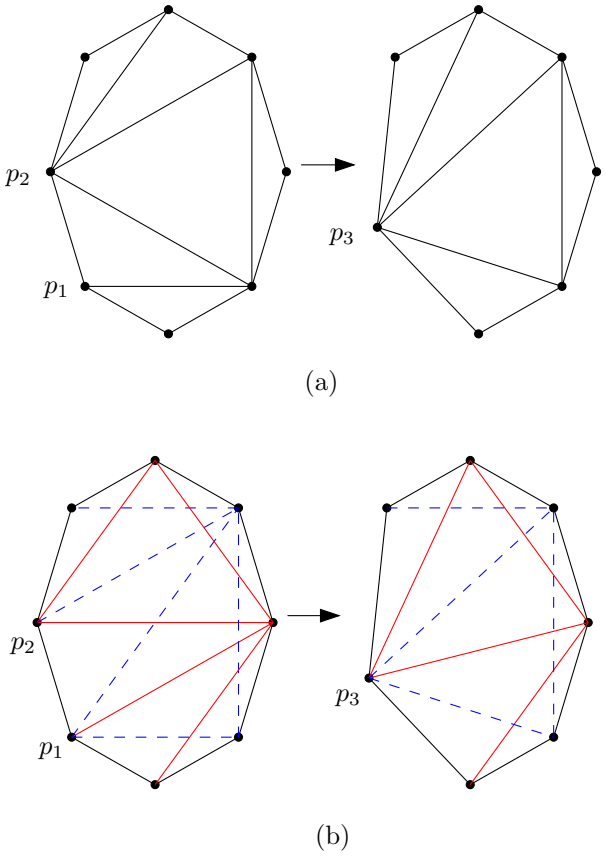


Fig. 1. (a) Edge contraction of a triangulation. (b) Edge contraction of two triangulations. The diagonals of one triangulation are solid and the diagonals of the other triangulation are dashed.

As in Barnette's proof we use the operation of edge removal. The difference is that we will not apply it for diagonals of P. This prevents the appearence of faces with more than three vertices. The *edge contraction* in a configuration is defined by idetifying the edge enpoints. If applied to one triangulation of P, it produces a triangulation, see Fig. 1 (a) for example where the edge p_1p_2 is contracted. When applied for two triangulations, we want the reduced triangulations to be distinct. An edge e of a configuration (P, T_1, T_2) is *contractible* if the new triangulations T_1' and T_2'' are distinct. In general, not all edges are contractible. For example, the edge p_1p_2 in the Fig. 2 (a) is not contractible since two edges p_1p_6 and p_2p_6 from different triangulations coincide after the contraction of p_1p_2.

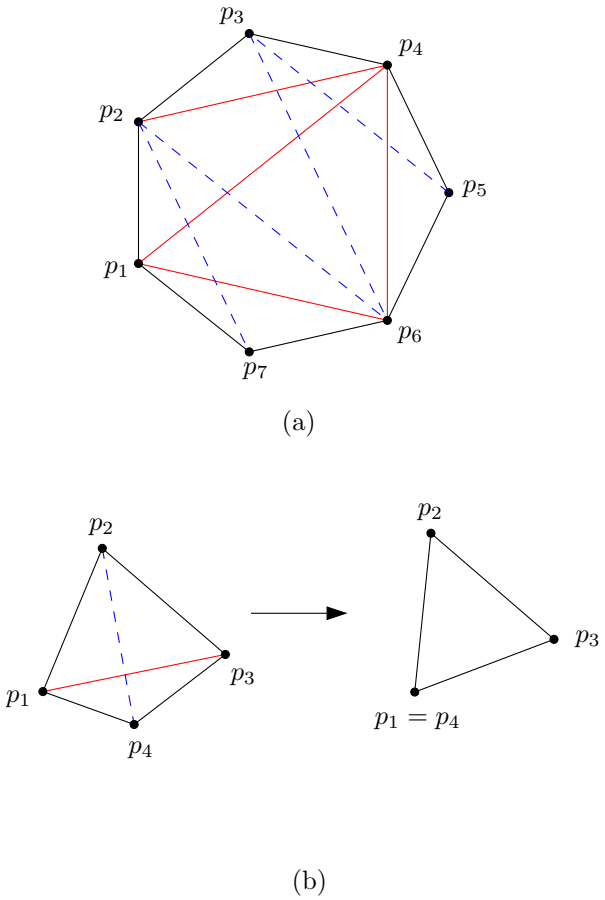


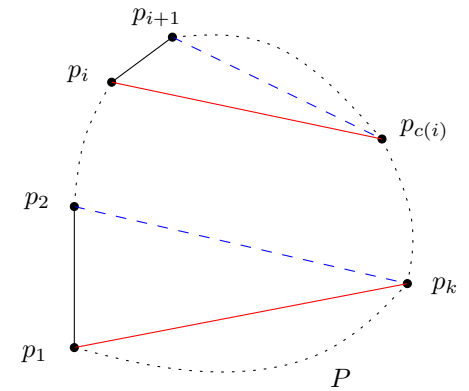
Fig. 2. (a) The edge (p_1, p_2) is not contractible. (b) The edge contraction for $n = 4$.

Lemma 2 *Let $\mathcal{C}$ be a configuration with $n \geq 4$ vertices. There is a contractible edge of $\mathcal{C}$ among the edges of the convex polygon.*

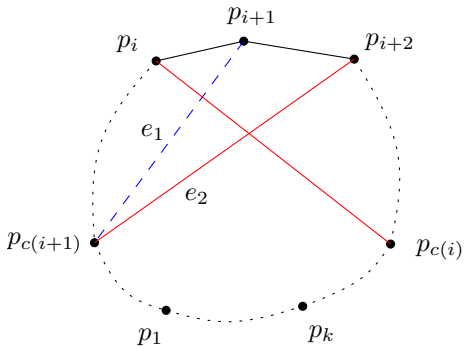
PROOF. If $n = 4$ then every edge of the convex polygon is contractible, see Fig. 2 (b). We prove the lemma for $n \geq 5$. Suppose to the contrary that there is a configuration (P, T_1, T_2) such that all edges of P are not contractible. Let $p_1, \ldots, p_n$ be

the vertices of P in clockwise order. The edge p_1p_2 is not contractible. Then there is a vertex $p_k, 4 \leq k \leq n-1$ such that p_1p_k is an edge of one triangulation, say T_1, and p_2p_k is an edge of T_2, see Fig. 3 (a).

Consider an edge $p_ip_{i+1}, 2 \leq i \leq k-1$. Since p_ip_{i+1} is not contractible, there is a vertex $p_{c(i)}$ such that $p_ip_{c(i)}$ is a diagonal of $T_j, j = 1, 2$ and $p_{i+1}p_{c(i)}$ is a diagonal of T_{3-j}, see Fig. 3 (a). We call $p_{c(i)}$ a *witness* since it indicates that p_ip_{i+1} is not contractible. At least one vertex of $\{p_i, p_{i+1}\}$, say p_l, is different from p_2 and p_k. Then the edge $p_lp_{c(i)}$ does not cross one of the edges p_1p_k or p_2p_k. Therefore $c(i)$ is an index in the range $1, \ldots, k$.



(a)



(b)

Fig. 3. Lemma 2.

We call $p_{c(i)}$ a *left witness* if $c(i) < i$. We call $p_{c(i)}$ a *right witness* if $c(i) > i+1$. Each witness is either left or right since $c(i) \neq i, i+1$. Note that $p_{c(2)}$ is a right witness and $p_{c(k-1)}$ is a left witness. Thus there is an index $i, 2 \leq i \leq k-2$ such that $p_{c(i)}$ is the right index and $p_{c(i+1)}$ is the left index, see Fig. 3 (b). Then $p_ip_{c(i)}$, a diagonal of a triangulation T_j, intersects both diagonals $e_1 = (p_{i+1}, p_{c(i+1)})$ and $e_2 = (p_{i+2}, p_{c(i+1)})$. Either e_1 or e_2 is a diagonal of T_j. Contradiction.

3. Edge creation

We define an operation of *edge creation* as the reverse operation of the edge contraction. The following lemma chracterizes the change of the configuration when an edge is created. We denote the sequence of indices from i to j in clockwise order by $\{i, i+1, \ldots, j\}$.

Lemma 3 (Edge creation) *Let $\mathcal{C} = (P, T_1, T_2)$ be a configuration with $n \geq 3$ vertices where $P = \{p_1, \ldots, p_n\}$. Suppose that an edge $e = (q_1, q_2)$ is created in place of a vertex $p_i \in P$. Let $\mathcal{C}' = (P', T_1', T_2')$ be the configuration obtained by replacing a vertex p_i by an edge $e = (q_1, q_2)$ in clockwise order. Then there are two edges $(p_i, p_j) \in T_1$ and $(p_i, p_k) \in T_2$ such that*

- *an edge $(p_l, p_i) \in T_1, l \in \{i+1, i+2, \ldots, j\}$ is replaced by the edge $(p_l, q_1) \in T_1'$, and*
- *an edge $(p_l, p_i) \in T_1, l \in \{j, j+1, \ldots, i-1\}$ is replaced by the edge $(p_l, q_2) \in T_1'$, and*
- *an edge $(p_l, p_i) \in T_2, l \in \{i+1, i+2, \ldots, p_k\}$ is replaced by the edge $(p_l, q_1) \in T_2'$, and*
- *an edge $(p_l, p_i) \in T_2, l \in \{k, k+1, \ldots, i-1\}$ is replaced by the edge $(p_l, q_2) \in T_2'$.*

We show that an edge can be always created.

Theorem 4 *Let $\mathcal{C} = (P, T_1, T_2)$ be a configuration with $n \geq 3$ vertices and let $\mu : P \to \mathbb{R}^3$ be its realization in $\mathbb{R}^3$. Let $\mathcal{C}' = (P', T_1', T_2')$ be the configuration obtained by an edge creation. There is a realization of $\mathcal{C}'$ if there is a realization of $\mathcal{C}$.*

Theorem 1 follows from Theorem 4.

References

[1] D. W. Barnette. Projections of 3-polytopes. *Israel J. Math.*, 8:304–308, 1970.

[2] B. V. Dekster. Convex hulls of spatial polygons with a fixed convex projection. *Beiträge zur Algebra und Geometrie/Contributions to Algebra and Geometry*, 36:123–124, 1995.

[3] B. Marlin and G. Toussaint. Constructing convex 3-polytopes from two triangulations of a polygon. In *Proc.*

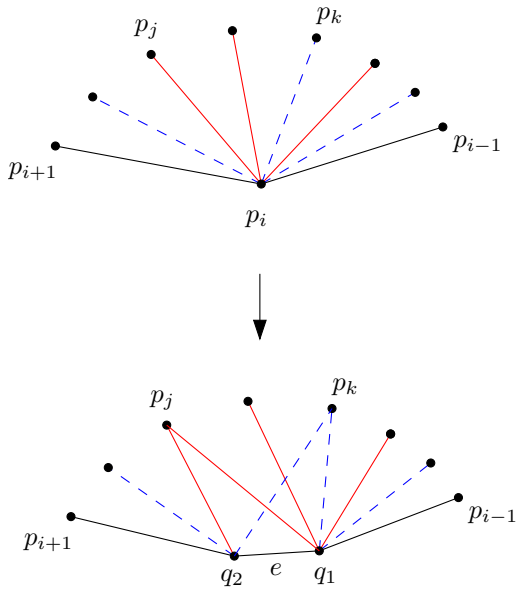


Fig. 4. Edge creation.

14th Canad. Conf. Comput. Geom., pp. 36–39, 2002, `http://www.cccg.ca/proceedings/2002/28.ps`.

[4] W. J. Moser. Problems, problems, problems. *Discrete Applied Mathematics*, 31:201–225, 1991.

[5] E. Steinitz and H. Rademacher. *Vorlesungen uber die Theorie der Polyeder*. Julius Springer, Berlin, Germany, 1934.

[6] K. Sugihara. A necessary and sufficient condition for a picture to represent a polyhedral scene. *IEEE Trans. on Pattern Analysis and Mach. Intelligence*, 6(5):578–586, 1984.

[7] G. Toussaint. Personal communication.

Planar embeddability of the vertices of a graph using a fixed point set is NP-hard [1]

Sergio Cabello
Institute of Information and Computing Sciences
Utrecht University
The Netherlands

Abstract

Let $G = (V, E)$ be a graph with n vertices and let P be a set of n points in the plane. We show that deciding whether there is a planar straight-line embedding of G such that the vertices V are embedded onto the points P is NP-complete, even when G is 2-connected and 2-outerplanar. This settles an open problem posed in [2,4,14].

1. Introduction

A *geometric graph* H is a graph $G(H)$ together with an injective mapping of its vertices into the plane. An edge of the graph is drawn as a straight-line segment joining its vertices. We use $V(H)$ for the set of points where the vertices of $G(H)$ are mapped to, and we do not make a distinction between the edges of $G(H)$ and H. A *planar* geometric graph is a geometric graph such that its edges intersect only at common vertices. In this case, we say that H is a *geometric planar embedding* of $G(H)$. See [15] for a survey on geometric graphs.

Let P be a set of n points in the plane, and let G be a graph with n vertices. What is the complexity of deciding if there is a straight-line planar embedding of G such that the vertices of G are mapped onto P? This question has been posed as open problem in [2,4,14], and here we show that this decision problem is NP-complete. Let us rephrase the result in terms of geometric graphs.

Theorem 1 *Let P be a set of n points, and let G be a graph on n vertices. Deciding if there exists a geometric planar embedding H of G such that $V(H) = P$ is an NP-complete problem.*

The reduction is from 3-partition, a strongly NP-hard problem to be described below, and it constructs a 2-connected graph G. The main ideas of the proof are given in Section 2, and we use that the maximal 3-connected blocks of a 2-connected planar graph can be embedded in different faces. In a 3-connected planar graph, all planar embeddings are topologically equivalent due to Whitney's theorem [10, Chapter 6]. Therefore, it does not seem possible to extend our technique to show the hardness for 3-connected planar graphs.

Related work A few variations of the problem of embedding a planar graph into a fixed point set have been considered. The problem of characterizing what class of graphs can be embedded into any point set in general position (no three points being collinear) was posed in [12]. They showed that the answer is the class of *outerplanar* graphs, that is, graphs that admit a straight-line planar embedding with all vertices in the outerface. This result was rediscovered in [6], and efficient algorithms for constructing such an embedding for a given graph and a given point set are described in [2]. The currently best algorithm runs in $O(n \log^3 n)$ time, although the best known lower bound is $\Omega(n \log n)$.

A tree is a special case of outerplanar graph. In this case, we also can choose to which point the root should be mapped. See [3,13,16] for the evolution on this problem, also from the algorithmical point of view. For this setting, there are algorithms running in $O(n \log n)$ time, which is worst case optimal. Bipartite embeddings of trees were considered in [1].

Email address: sergio@cs.uu.nl (Sergio Cabello).

[1] Partially supported by the Cornelis Lely Stichting. A full version is available as [5].

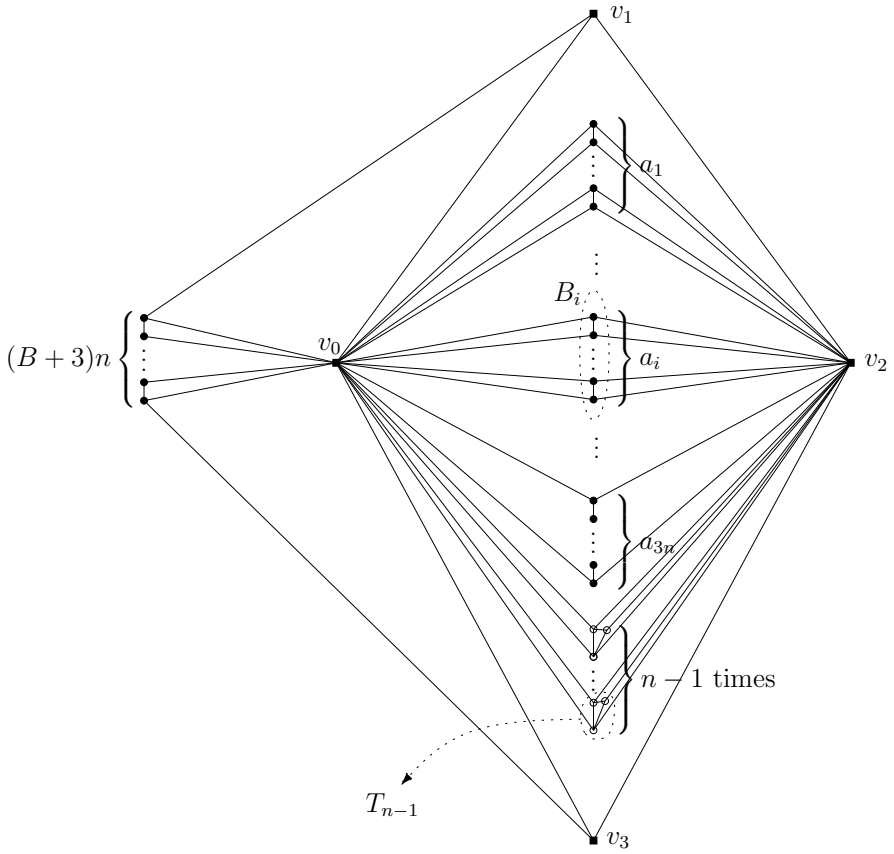


Fig. 1. Graph G for the NP-hardness reduction.

If we allow each edge to be represented by a polygonal path with at most two bends, then it is always possible to get a planar embedding of a planar graph that maps the vertices to a fixed point set [14]. If a bijection between the vertices and the point set is fixed, then we need $O(n^2)$ bends in total to get a planar embedding of the graph, which is also asymptotically tight in the worst case [17].

We finish by mentioning a related problem, which was the initial motivation for this research. A universal set for graphs with n vertices is a set of points S_n such that any planar graph with n vertices has a straight-line planar embedding whose vertices are a subset of S_n. Asymptotically, the smallest universal set is known to have size at least $1.098n$ [7], and it is bounded by $O(n^2)$ [8,18]. Characterizing the asymptotic size of the smallest universal set is an interesting open problem [9, Problem 45].

2. Planar embeddability is NP-complete

It is clear that the problem belongs to NP: a geometric graph H with $V(H) = P$ and $G(H) \equiv G$ can be described by the bijection between $V(G)$ and P, and for a given bijection we can test in polynomial time whether it actually is a planar geometric graph; therefore, we can take as certificate the bijection between $V(G)$ and P.

For showing the NP-hardness, the reduction is from 3-partition.

Problem: 3-partition

Input: A natural number B, and $3n$ natural numbers $a_1, \ldots, a_{3n}$ with $\frac{B}{4} < a_i < \frac{B}{2}$.

Output: n disjoint sets $S_1, \ldots, S_n$ such that for all S_j we have $S_j \subset \{a_1, \ldots, a_{3n}\}$, $|S_j| = 3$, and $\sum_{a \in S_j} a = B$.

We will use that 3-partition is a strongly NP-hard problem, that is, it is NP-hard even if B is bounded by a polynomial in n [11]. Observe that because $\frac{B}{4} < a_i < \frac{B}{2}$, it does not make sense to have sets

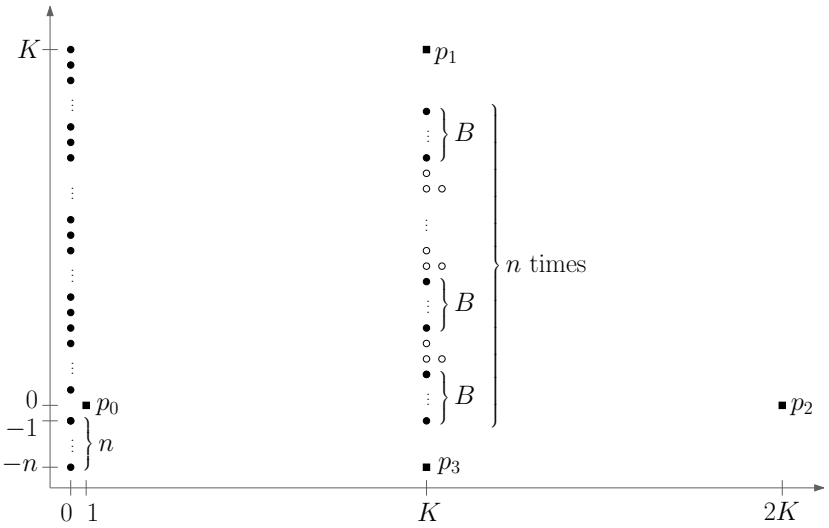


Fig. 2. Point set P for the NP-hardness reduction. $K = (B+2)n$.

S_j with fewer or more than 3 elements. That is, it is equivalent to ask for subdividing all the numbers into disjoint sets that sum to B. Of course, we can assume that $\sum_{i=1}^{3n} a_i = Bn$, as otherwise it is impossible that a solution exists.

In the following, we only give the reduction, and do not discuss how the solution to the constructed problem relates to the original problem. Details can be found in the full version [5]. Given a 3-partition instance, we construct the following graph G (see Figure 1):

- Start with a 4-cycle with vertices $v_0, \ldots, v_3$, and edges $(v_{i-1}, v_{i \bmod 4})$. The vertices v_0 and v_2 will play a special role.
- For each a_i in the input, make a path B_i consisting of a_i vertices, and put an edge between each of those vertices and the vertices v_0, v_2.
- Construct $n-1$ triangles $T_1, \ldots, T_{n-1}$. For each triangle T_i, put edges between each of its vertices and v_2, and edges between two of its vertices and v_0. We call each of these structures a separator (the reason for this will become clear later).
- Make a path C of $(B+3)n$ vertices, and put edges between each of the vertices in C and v_0. Furthermore, put an edge between one end of the path and v_1, and another edge between the other end and v_3.

It is easy to see that G is planar; in fact, we are giving a planar embedding of it in Figure 1. The idea is to design a point set P such that $G \setminus (B_1 \cup \cdots \cup B_{3n})$ can be embedded onto P in essentially one way. Furthermore, the embedding of $G \setminus (B_1 \cup \cdots \cup B_{3n})$ will decompose the rest of the points into n groups, each of B vertices and lying in a different face. The embedding of the remaining vertices $B_1 \cup \cdots \cup B_{3n}$ will be possible in a planar way if and only if the paths B_i can be decomposed into groups of exactly B vertices, which is equivalent to the original 3-partition instance. The following point set P does the work (see Figure 2):

- Let $K := (B+2)n$.
- Place $(B+3)n$ points at the coordinates $(0,-n), (0,-(n-1)), \ldots, (0,-1)$ and at the coordinates $(0,1), (0,2), \ldots, (0,K)$.
- Place points $p_0 := (1,0), p_1 := (K,K), p_2 := (2K,0), p_3 := (K,-n)$. In the figure, these points are shown as boxes and are labeled.
- For each $i \in \{0, \ldots, n-1\}$, place the group of B points $(K, -2+(B+2)i+1), (K, -2+(B+2)i+2), \ldots, (K, -2+(B+2)i+B)$.
- For each $i \in \{1, \ldots, n-1\}$, place the group of three points $(K, (B+2)i-3), (K, (B+2)i-2), \ldots, (K+1, (B+2)i-3)$. In the figure, these points are shown as empty circles.

3. Concluding remarks

The point set P that we have constructed has many collinear points. However, in the proof we do not use this fact, and so it is easy to modify the reduction in such a way that no three points of P are collinear. Probably, the easiest way for keeping

integer coordinates is replacing each of the points lying in a vertical line by points lying in a parabola, and adjusting the value K accordingly. Therefore, the result remains valid even if P is in general position, meaning that no 3 points are collinear.

In the proof, the graph G that we constructed is 2-outerplanar, as shown in Figure 1. k-outerplanarity is a generalization of outerplanarity that is defined inductively. A planar embedding of a graph is *k-outerplanar* if removing the vertices of the outer face produces a $(k-1)$-outerplanar embedding, where 1-outerplanar stands for an outerplanar embedding. A graph is k-outerplanar if it admits a k-outerplanar embedding. For outerplanar graphs, the embedding problem is polynomially solvable [2], but for 2-outerplanar we showed that it is NP-complete. Therefore, regarding outerplanarity, our result is tight.

The graph G that we constructed in the proof is 2-connected; removing the vertices v_1, v_3 disconnects the graph. As mentioned in the introduction, we use this fact in the proof because in a 2-connected graph the maximal 3-connected blocks can flip from one face to another one. Therefore, it would be interesting to find out the complexity of the problem when the graph G is 3-connected, and more generally, the complexity when the topology of the embedding is specified beforehand.

Acknowledgements

I would like to thank Marc van Kreveld for careful reading of the manuscript and pointing out that the reduction can use a 2-outerplanar graph.

References

[1] M. Abellanas, J. García-López, G. Hernández, M. Noy, and P. A. Ramos. Bipartite embeddings of trees in the plane. *Discrete Applied Mathematics*, 93:141–148, 1999. A preliminary version appeared in *Graph Drawing (Proc. GD '96)*, LNCS 1190, pg. 1–10.

[2] P. Bose. On embedding an outer-planar graph in a point set. *Comput. Geom. Theory Appl.*, 23:303–312, November 2002. A preliminary version appeared in *Graph Drawing (Proc. GD '97)*, LNCS 1353, pg. 25–36.

[3] P. Bose, M. McAllister, and J. Snoeyink. Optimal algorithms to embed trees in a point set. *Journal of Graph Algorithms and Applications*, 1(2):1–15, 1997. A preliminary version appeared in *Graph Drawing (Proc. GD '95)*, LNCS 1027, pg. 64–75.

[4] F. Brandenberg, D. Eppstein, M.T. Goodrich, S.G. Kobourov, G. Liotta, and P. Mutzel. Selected open problems in graph drawing. In *Graph Drawing (Proc. GD'03)*, LNCS, 2003. To appear.

[5] S. Cabello. Planar embeddability of the vertices of a graph using a fixed point set is NP-hard. Technical report UU-CS-2003-031. Available at `http://www.cs.uu.nl/research/techreps/UU-CS-2003-031.html`, 2003.

[6] N. Castaneda and J. Urrutia. Straight line embeddings of planar graphs on point sets. In *Proc. 8th Canad. Conf. Comput. Geom.*, pages 312–318, 1996.

[7] M. Chrobak and H. Karloff. A lower bound on the size of universal sets for planar graphs. *SIGACT News*, 20:83–86, 1989.

[8] H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.

[9] E. D. Demaine, J. S. B. Mitchell, and J. O'Rourke (eds.). The Open Problems Project. `http://cs.smith.edu/~orourke/TOPP/Welcome.html`.

[10] R. Diestel. *Graph Theory*. Springer-Verlag, New York, 2nd edition, 2000.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.

[12] P. Gritzmann, B. Mohar, J. Pach, and R. Pollack. Embedding a planar triangulation with vertices at specified points. *Amer. Math. Monthly*, 98(2):165–166, 1991.

[13] Y. Ikebe, M. A. Perles, A. Tamura, and S. Tokunaga. The rooted tree embedding problem into points in the plane. *Discrete Comput. Geom.*, 11:51–63, 1994.

[14] M. Kaufmann and R. Wiese. Embedding vertices at points: Few bends suffice for planar graphs. *Journal of Graph Algorithms and Applications*, 6(1):115–129, 2002. A preliminary version appeared in *Graph Drawing (Proc. GD '99)*, LNCS 1731, pg. 165–174.

[15] J. Pach. Geometric graph theory. In J. D. Lamb and D. A. Preece, editors, *Surveys in Combinatorics*, number 267 in London Mathematical Society Lecture Note Series, pages 167–200. Cambridge University Press, 1999.

[16] J. Pach and J. Töumlrőcsik. Layout of rooted trees. In W.T. Trotter, editor, *Planar Graphs*, volume 9 of *DIMACS Series*, pages 131–137. Amer. Math. Soc., Providence, 1993.

[17] J. Pach and R. Wenger. Embedding planar graphs at fixed vertex locations. *Graphs Combin.*, 17:717–728, 2001. A preliminary version appeared in *Graph Drawing (Proc. GD '98)*, LNCS 1547, pg. 263–274.

[18] W. Schnyder. Embedding planar graphs on the grid. In *Proc. 1st ACM-SIAM Sympos. Discrete Algorithms*, pages 138–148, 1990.

New Bound for Incremental Constructing Arrangements of Curves

Manuel Abellanas [a,1], Aymée Calatayud [*,b,1], Jesús García-López [c,2],

[a] *Dpt. Mat. Aplic., FI, UPM, Boadilla del Monte, 28660 Madrid, Spain*
[b] *Dpt. Mat. Aplic., FI, UPM, Boadilla del Monte, 28660 Madrid, Spain*
[c] *Dpt. Mat. Aplic., EUI, UPM, Ctra. Valencia Km 7, 28031 Madrid, Spain*

Abstract

Let $A(\Gamma)$ be the arrangement induced by a set Γ of n unbounded Jordan curves in the plane that intersect each other in at most two points. The upper bound for constructing those arrangements by an incremental method is, up to now, $O(n\lambda_4(n))$. In this paper we improve this bound to $O(n\lambda_3(n))$.

Key words: arrangements of curves, incremental algorithms, zones

1. Introduction

The arrangement of a set Γ of n unbounded Jordan curves, denoted by $A(\Gamma)$, is the plane subdivision induced by the curves in Γ. Let γ_0 be a curve not belonging to Γ. The *zone* of γ_0 in $A(\Gamma)$, denoted by $z(\gamma_0)$, is the set of faces of $A(\Gamma)$ intersected by γ_0.

Up to now, the best known upper bound for the zone of γ_0 in $A(\Gamma)$ is $O(\lambda_4(n)) = O(n\,2^{\alpha(n)})$, obtained from its relation with the complexity of a face in an arrangement of Jordan arcs: by deleting small pieces in the curves of Γ just containing the intersection points with γ_0, all faces of $z(\gamma_0)$ become a part of the same face of an arrangement of Jordan arcs. The total amount of new elements in the arrangement is a constant factor of the size of the arrangement ([3], pg. 23). By using an incremental algorithm, in which curves are inserted in the arrangement one by one, the construction of $A(\Gamma)$ takes $O(n\,\lambda_4(n))$ time. In this paper we improve this bound to $O(n\,\lambda_3(n))$ if the curves intersect each other in at most two points (remember that $\lambda_4(n) = O(n2^{\alpha(n)})$ and $\lambda_3(n) = O(n\alpha(n))$).

The algorithm that computes $A(\Gamma)$ in $O(n\lambda_3(n))$ time is based in three lemmas.

The first lemma considers the frontier of a face in the external zone composed by the edges of the face not belonging to γ_0 and says that if p and q are the extreme points of the frontier of a face in the external zone, then the segment $\overline{pq}$ in γ_0 is an edge of the arrangement. This property allows to travel through the frontiers of the successive faces of the external zone thus obtaining the intersection points of γ_0 with the curves in the arrangement just in the order they appear in γ_0. Therefore, the time complexity for the insertion of γ_0 depends on the complexity of the external zone which is $O(\lambda_3(n))$.

The second lemma considers the case in which the infinite extremes of all the curves that intersect γ_0 in two points are in the same side of γ_0, and proves that the complexity of the zone in this side of γ_0 is $O(\lambda_3(n))$. We call *external zone* to this part of the zone.

Last lemma shows that it is always possible to list the curves of Γ in order $\gamma_1, \gamma_2, \ldots, \gamma_n$, such that each γ_i with $2 \leq i \leq n$ verifies that the infinite extremes of every curve in the set $\{\gamma_1, \gamma_2, \ldots, \gamma_{i-1}\}$ that intersects γ_i in zero or two points are in the same side of γ_i.

* Corresponding author
Email addresses: mabellanas@fi.upm.es (Manuel Abellanas), acalatayud@fi.upm.es (Aymée Calatayud), jglopez@eui.upm.es (Jesús García-López).

[1] Partially supported by *MCYT* TIC2003-08933-C02-01
[2] Partially supported by *MCYT* TIC2002-01541

2. Notation and definitions

Every curve in Γ intersecting γ_0 is decomposed by γ_0 in two or three connected pieces. The unbounded pieces will be called *branches*. Branches will be oriented, being the starting point the intersection point with γ_0. This point will be called the *base* of the branch. Base of a branch r will be noted by $\overline{r}$. Every curve in Γ intersecting γ_0 in two points give rise to two branches.

If all branches of the curves intersecting γ_0 in two points are in the same side of γ_0 we will say that γ_0 is a pseudo-convex curve in the arrangement and the *external zone* of γ_0 will be the half-zone containing all these branches. From now on γ_0 will be pseudo-convex in the arrangement $A(\Gamma)$ and we will only consider the external zone of γ_0, noted by $z^+(\gamma_0)$.

We will call *external frontier* every connected component obtained by deleting edges contained in γ_0 in the frontier of a face of the external zone. As a consequence, extreme points of the external frontiers belong to γ_0.

Without lost of generality, we will suppose that every face of the external zone, but the first and the last, are bounded. If this where not the case, one can add a curve to Γ that intersects only the unbounded faces.

3. The external frontier

Lemma 1 *The extreme points of an external frontier are the extreme points of an edge contained in γ_0 in the arrangement* $\mathrm{A}(\Gamma \cup \{\gamma_0\})$.

PROOF. Let F^* be the external frontier of a face in the external zone. Let p and q be the extreme points of F^*. If $\overline{pq}$, the arc of γ_0 with extreme points p and q, were not an edge in the mentioned arrangement, $\overline{pq}$ would contain at least one intersection point c in its interior. As $\overline{pq}$ is contained in γ_0, then necessarily c is the base of a branch. Because branches are not bounded, c and F^* intersect each other. This is a contradiction with the fact that F^* is an external frontier of a face.

4. External zone complexity

In this section one proves the $O(\lambda_3(n))$ complexity of the external zone of γ_0 in $A(\Gamma)$. It is proved by bounding the number of its edges.

The idea is to visit the external frontiers in the order they appear along γ_0 and write down, for every edge encountered, the branch or supporting curve. Let $U = \langle u_1, u_2, \ldots, u_m \rangle$ be the obtained list of symbols. One realizes that U is a Davenport-Schinzel sequence.

(n, s) Davenport-Schinzel sequences have to satisfy the following properties (see [4], pg. 1):

(i) To contain at most n different symbols.

(ii) $u_i \neq u_{i+1}$ for every $1 \leq i < m$.

(iii) there cannot be $s+2$ indices $1 \leq i_1 < i_2 < \ldots < i_{s+2} \leq m$ such that $u_{i_1} = u_{i_3} = u_{i_5} \ldots = a$, $u_{i_2} = u_{i_4} = u_{i_6} \ldots = b$ and $a \neq b$.

In order to have condition (ii) satisfied one uses different symbols for each side of the same branch. In this way, n curves give rise to $4n$ symbols at most (two for every of the $2n$ possible branches).

Proposition 2 *U is a $(4n, 3)$ Davenport-Schinzel sequence.*

PROOF.

(i) $4n$ symbols corresponds to the, at most, $2n$ branches.

(ii) In the same face there are not two consecutive edges from the same branch (nor curve, if it does not intersect γ_0). This is due to the fact that curves intersect in a transversal way. When passing from one face to the consecutive one, there are neither two equal consecutive symbols because they corresponds to the two sides of the same branch, marked with different symbols.

(iii) Third condition is due to the fact that two branches intersect in at most two points. One can prove that, under this condition, the maximum length of an alternating sub-chain of two symbols is 4.

Lemma 3 *The external zone of γ_0 has $O(\lambda_3(n))$ complexity.*

PROOF. By proposition 2, it suffices to note that $\lambda_3(4n) = O(\lambda_3(n))$.

5. Insertion order

In this section it's seen how to sort the set of curves Γ in such a way that, if they are inserted one by one in this order in the arrangement, curve γ_i is pseudo-convex with respect to the previous γ_j inserted curves, $2 \leq j \leq i-1$.

Without lost of generality one can suppose that the external zone of every curve is always to its left.

The idea for sorting is the following: a curve γ is fixed while the others are classified with respect to γ into two sets that will be recursively sorted: the set of curves that must be inserted before γ in the arrangement and the set of curves that have to be inserted after γ. We insert before γ all the curves that give rise to unbounded branches to the left of γ (γ must be oriented before doing this classification). Let $|\delta \cap \gamma|$ denote the number of intersection points between δ and γ.

Procedure for sorting the set Γ;

(1) If $\Gamma = \emptyset$ nothing to do. Return $\emptyset$.
(2) Select at random a curve $\gamma \in \Gamma$.
(3) If it is still not oriented, select an orientation for γ.
(4) Classify the rest of curves into two sets $C_1(\gamma)$ and $C_2(\gamma)$ containing the curves of Γ - $\{\gamma\}$ that have to be inserted in the arrangement before and after γ respectively:
- Curve δ is included in $C_1(\gamma)$ whenever one of the three following conditions holds:
  $|\delta \cap \gamma| = 0$ and δ is to the left of γ.
  $|\delta \cap \gamma| = 1$.
  $|\delta \cap \gamma| = 2$ and the branches of δ are to the left of γ.
- Curve δ is included in $C_2(\gamma)$ if it has not been included in $C_1(\gamma)$, that is:
  $|\delta \cap \gamma| = 0$ and δ is to the right of γ or
  $|\delta \cap \gamma| = 2$ and the branches of δ are to the right of γ.

(5) Orient the curves in $C_2(\gamma)$ depending on the orientation of γ in such a way that the infinite extremes of γ lay to the left of each curve.
(6) Recursively repeat the procedure for $C_1(\gamma)$ and $C_2(\gamma)$, thus obtaining the sorted lists $L_1(\gamma)$ and $L_2(\gamma)$ respectively.
(7) The final sorting in Γ is $L_1(\gamma) + [\gamma] + L_2(\gamma)$, being "+" the concatenation of lists operation.

Definition 4 *Curve γ is called reference curve of the process. Sets $C_1(\gamma)$ and $C_2(\gamma)$ determined by γ are called associated sets.*

One can proof that the orientation that a reference curve γ determine in its associated set $C_2(\gamma)$ do not prevent a future sorting of this set. No care is necessary with curves in $C_1(\gamma)$ because they appear in the arrangement before γ.

Observation: Every curve is, at some point during the execution of the procedure, a reference curve having the corresponding associated sets (may be empty some of them).

In order to facilitate the proof of the pseudo-convexity of every curve with respect to the preceding ones, one consider a surrounding pseudo-circumference containing in its interior all intersection points of the arrangement. This extra curve represents the infinity and intersects every curve in two points.

Definition 5 *The intersection points between a curve $\gamma \in \Gamma$ and the pseudo-circumference are called the extreme points of γ. They are considered points at infinity.*

Definition 6 *Every curve γ divides the pseudo-circumference in two pieces called vaults. Once the curve γ is oriented, the corresponding vaults becomes left vault and right vault of γ. The left vault is denoted by $\widehat{\gamma}$.*

Vault concept allow us to define pseudo-convexity in the following way:

Definition 7 *γ is pseudo-convex with respect to a set of curves if all of them have at least one intersection point with $\widehat{\gamma}$.*

Proposition 8 *A curve γ and its associated sets verify the following properties:*
(i) γ is pseudo-convex with respect to curves in $C_1(\gamma)$.
(ii) Curves in $C_2(\gamma)$ are pseudo-convex with respect to γ.
(iii) Curves in $C_2(\gamma)$ are pseudo-convex with respect to those in $C_1(\gamma)$.

PROOF. Items (i) and (ii) are verified by construction. For item (iii), let be $\gamma_1 \in C_1(\gamma)$. By construction, γ_1 have at least an intersection point with $\widehat{\gamma}$. Let be $\gamma_2 \in C_2(\gamma)$. By construction, $\widehat{\gamma} \subset \widehat{\gamma_2}$. That means that γ_1 have at least an intersection point with $\widehat{\gamma_2}$. Therefore, by definition 7, γ_2 is pseudo-convex with respect to γ_1.

Finally, we can proof the third lemma:

Lemma 9 *If γ_1, γ_2, ..., γ_n is the sorted set of curves given by the sorting procedure, then γ_i is pseudo-convex with respect to γ_1, γ_2, ..., γ_{i-1} for every $2 \leq i \leq n$.*

PROOF.

Let be $2 \leq i \leq n$ and let γ_j be a curve with $j < i$. By construction there is a reference curve γ_k, $j \leq k \leq i$, separating γ_i and γ_j, thus having one of the three following situations:

(i) $\gamma_k = \gamma_i$ and $\gamma_j \in C_1(\gamma_k)$.

(ii) $\gamma_k = \gamma_j$ and $\gamma_i \in C_2(\gamma_k)$.

(iii) $\gamma_j \in C_1(\gamma_k)$ and $\gamma_i \in C_2(\gamma_k)$.

Taking into account proposition 8 one verifies in all three cases that γ_i is pseudo-convex with respect to γ_j.

As a consequence of the previous lemmas, one verifies the main theorem of the paper:

Theorem 10 *The arrangement* $\mathrm{A}(\Gamma)$ *of unbounded Jordan curves that intersect each other in at most two points can be computed incrementally in $O(n\lambda_3(n))$ time.*

6. Applications

This result can be applied in geometric location problems. In particular in covering problems with geometric figures as wedges, circular annulus or strips.

In these problems one wants to locate the best position for a geometric figure in the plane in order to cover as much points of a given set of points as possible. In the solution of these problems in the dual space, arrangements as the considered in this paper have to be managed. (Details can be found in [1]).

7. Open problems

An open problem is to determine the complexity of the internal zone of this kind of arrangements. Internal zone is the half-zone which is not the external zone. It is an open problem for $s \geq 1$ for specific families of curves. For arbitrary curves the known complexity is $O(\lambda_{s+2}(n))$ ([4], pág. 125).

Note that for $s = 1$ one have a known problem that is open from more than ten years (as mentioned M. Sharir in the Dagstuhl Workshop [2]):

Problem 11 *Given a circle and an arrangement of pseudo-lines determine the complexity of the half-zone of the circle contained in its interior.*

References

[1] A. Calatayud, *Problemas Geométricos de Localización*, Ph.D., Universidad Politécnica de Madrid, 2004.

[2] Dagstuhl Seminar on Computational Geometry, March, 2003.

[3] J. Pach, *New Trends in Discrete and Computational Geometry*, Springer-Verlag, 1993.

[4] Micha Sharir, Pankaj K. Agarwal, *Davenport-Schinzel Sequences and their Geometric Applications*, Cambridge University Press, 1995.

On relative isodiametric inequalities

Cerdán, A. [a], Miori, C. [a]and Segura Gomis, S. [a]

[a]*Departamento de Análisis Matemático, Universidad de Alicante, Campus de San Vicente del Raspeig, E-03080-Alicante, Spain*

Abstract

We consider subdivisions of convex bodies G in two subsets E and $G \setminus E$. We obtain several inequalities comparing the relative volume 1) with the minimum relative diameter and 2) with the maximum relative diameter. In the second case we obtain the best upper estimate only for subdivisions determined by straight lines in planar sets.

Key words: Relative geometric inequalities, relative isodiametric inequalities, area, diameter

1. Introduction

Relative geometric inequalities are inequalities in which we compare relative geometric measures, i.e., functionals that give geometric information of the subsets (E and $G\setminus E$) determined by the subdivision of an original set G.

The first relative geometric inequalities that appeared in the literature were the so called relative isoperimetric inequalities. These inequalities compare the relative area with the relative perimeter:

If G is an open convex set in the Euclidean plane $\mathbb{R}^2$ and E is a subset of G with non-empty interior and rectificable boundary such that both E and its complement $G\setminus E$ are connected, we define:

- the *relative boundary* of E as $\partial E \cap G$.
- the *relative perimeter* of E, $P(E,G)$, as the length of the relative boundary, and
- the *relative area* of E as:

$$A(E,G) = min\{A(E), A(G\setminus E)\}.$$

With the above assumptions we say that a relative isoperimetric inequality is an inequality of the type:

$$\frac{A(E,G)}{P(E,G)^{\alpha}} \leq C,$$

where C and α are positive numbers.

Email addresses: aacs@alu.ua.es (Cerdán, A.), cm4@alu.ua.es (Miori, C.), Salvador.Segura@ua.es (Segura Gomis, S.).

Many results have been obtained about relative isoperimetric inequalities (see for instance [2],[7]).

There are also results comparing the relative perimeter with other geometric magnitudes different from the relative area. For results comparing the relative perimeter with the relative diameter and the relative inradius see [4].

In this paper we want to study *relative isodiametric inequalities*, in which we compare the relative volume with the relative diameters of a subset of a convex body. First we need to define these notions:

Let $G \subset \mathbb{R}^n$ be a convex body and $E \subset G$ a subset of G such that E as well as $G \setminus E$ are connected and have non–empty interior. Let $D(.)$ be the diameter functional.

(i) the *relative volume* is the minimum between the volume of E and the volume of its complement,

$$V(E,G) = min\{V(E), V(G \setminus E)\},$$

(ii) the *minimum relative diameter* is the minimum between the diameter of E and the diameter of its complement,

$$d_m(E,G) = min\{D(E), D(G \setminus E)\},$$

and

(iii) the *maximum relative diameter* is the maximum between the diameter of E and the diameter of its complement,

$$d_M(E,G) = max\{D(E), D(G \setminus E)\}.$$

Relative isodiametric inequalities are those that give either an upper or a lower estimate of the ratios:

$$\frac{V(E,G)}{d_m(E,G)^n} \text{ or } \frac{V(E,G)}{d_M(E,G)^n}.$$

We compare the relative volume with the n-power of the relative diameters (n is the dimension of the ambient space) because as this ratio is invariant under dilatations, we obtain geometric information about the subdivision: the estimates do not depend on the size of the bodies but only on their shapes.

We are interested not only in obtaining relative isodiametric inequalities, but also in determining those sets (called maximizers or minimizers) for which the equality sign is attained.

2. Relative isodiametric inequalities concerning the relative area and the minimum relative diameter of a subset of a convex body

The aim of this section is to maximize and minimize the ratio between the relative area and the n-power of the minimum relative diameter of a subset E of G.

We begin minimizing the given ratio, and in this case we have to consider two different cases. First, we are going to study general subdivisions of G and later we are going to consider the special case in which the subdivision is obtained by a hyperplane cut.

Theorem 1 *Let G be an open convex body and E a subset of G such that E and $G \setminus E$ are connected. Then,*

$$\frac{V(E,G)}{d_m(E,G)^n} \geq 0.$$

PROOF. For any convex body G we can consider a sequence of hypersurfaces $\{S_i\}_{i=1}^{\infty}$ as close as we want to the boundary of G, such that both ends of the diameter of G belong to S_i $\forall i \in \mathbb{N}$, and the regions E_i bounded by S_i have volume decreasing to zero. Then, we conclude that:

$$\lim_{i\to\infty} \frac{V(E_i,G)}{d_m(E_i,G)^n} = \frac{0}{D(G)^n} = 0.$$

□

Theorem 2 *Let G be an open convex body and E a subset of G obtained by a hyperplane cut. Then,*

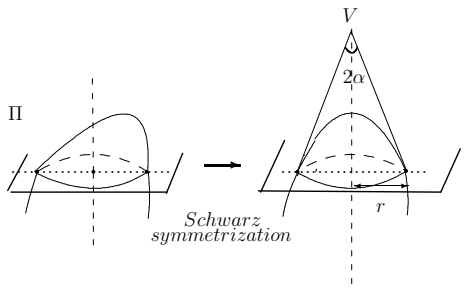
$$\frac{V(E,G)}{d_m(E,G)^n} \geq 0. \tag{1}$$

PROOF. We can distinguish two cases:
Case 1: G is strictly convex:

As G is a strictly convex body, we can choose a hyperplane Π so that if E is a subset of G obtained by the intersection with the half-space determined by Π, in all the points of $\partial E \setminus (G \cap \partial E)$ there exists a tangent hyperplane.

Let us consider a straight line t orthogonal to the hyperplane Π; we can apply the Schwarz symmetrization with respect to t to the subset E and we obtain a new set E' of revolution which has the same volume than E and such that the image of the relative boundary of E under Schwarz symmetrization is a (n-1)-dimensional ball with radius r. Moreover, this symmetrization does not increase the diameter, so $D(E) \geq D(E')$.

The body of revolution E' is contained in a n-dimensional cone C with vertex V, which obviously has greater volume than E'.



Finally, we consider a sequence of parallel hyperplanes to Π so that $V(E')$ decreases to zero and the half angle at the vertex V goes to $\pi/2$. Then,

$$\lim_{\alpha\to\pi/2} \frac{V(E,G)}{d_m(E,G)^n} \leq \lim_{\alpha\to\pi/2} \frac{V(E')}{(2r)^n} \leq$$

$$\lim_{\alpha\to\frac{\pi}{2}} \frac{\pi^{\frac{(n-1)}{2}} \operatorname{cotan}\,\alpha}{n2^n\Gamma(\frac{n-1}{2}+1)} = 0.$$

Consequently, the inequality (1) holds.

Case 2: G is not strictly convex:

If G is not strictly convex there exists a straight line segment t in the boundary of G. We consider a sequence of hyperplanes Π_i parallel to t so that

the volume of the subsets E_i determined by the intersections with Π_i decreases to zero. Then,

$$\lim_{i\to\infty} \frac{V(E_i,G)}{d_m(E_i,G)^n} \leq \frac{0}{(length(t))^n} = 0.$$

□

The following proposition provides an upper bound for the ratio between the relative area and the n-power of the minimum relative diameter.

Theorem 3 *Let G be an open convex body and E a subset of G such that E and $G \setminus E$ are connected. Then,*

$$\frac{V(E,G)}{d_m(E,G)^n} \leq \frac{\pi^{n/2}}{\Gamma(1+\frac{n}{2})2^n}$$

PROOF. Let $B(r)^n$ be a ball with radius r such that:

$$V(B(r)^n) \leq \frac{V(G)}{2} \Longleftrightarrow r \leq \left(\frac{V(G)}{2\omega_n}\right)^{1/n}$$

then, as a consequence of the isodiametric inequality (see [1]):

$$\frac{V(E,G)}{d_m(E,G)^n} \leq \frac{\pi^{n/2}}{\Gamma(1+\frac{n}{2})2^n}.$$

□

3. Relative isodiametric inequalities concerning the relative area and the maximum relative diameter of a subset of a convex body

The aim of this section is to maximize and minimize the ratio between the relative area and the n-power of the maximum relative diameter of a subset E of G.

Theorem 4 *Let G be an open bounded convex set in $\mathbb{R}^n$ and E a subset of G such that E and $G \setminus E$ are connected. Then,*

$$\frac{V(E,G)}{d_M(E,G)^n} \geq 0.$$

PROOF. Let G be an open convex body in the Euclidean space. We can suppose without lost of generality that $0 \in G$. Let us consider the sequence $\{E_i\}_{i=2}^{\infty}$ where each $E_i = \frac{1}{i}G$.

If we compute the ratio between the relative volume and the n-power of the maximum relative diameter, the limit of this ratio is 0 when $i \to \infty$. In fact, the relative volume decreases to 0 and $d_M(E_i,G) = D(G \setminus E_i)$ is the diameter of G for all i:

$$\lim_{i\to\infty} \frac{V(E_i,G)}{d_M(E_i,G)^n} = \frac{0}{D(G)^n} = 0.$$

□

Theorem 5 *Let G be an open convex body and E a subset of G obtained by a hyperplane cut. Then,*

$$\frac{V(E,G)}{d_M(E,G)^n} \geq 0.$$

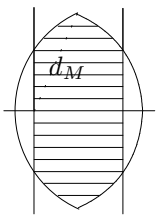
It is easy to prove this theorem using the same argument that in the proof of theorem 2 (when G is not strictly convex), considering $t = D(G)$.

The problem of maximizing the ratio between the relative area and the maximum relative diameter is attached to the so called "fencing problems". Such problems consider dividing a region into two parts of equal volume (area) by a fence. We are going to use some results about fencing problems for proving the following theorem (see [5], [6]).

Theorem 6 *Let G be a planar bounded convex set and E a subset of G obtained by a straight line cut, then:*

$$\frac{A(E,G)}{d_M(E,G)^2} \leq 1.2869...$$

The equality is attained for the optimal body described in the following figure (see [5]).



PROOF. Let l be the straight line dividing G into two regions E and $G \setminus E$, and suppose that $A(E) \leq A(G \setminus E)$. Let us consider two different cases:

1) $d_M(E,G) = D(G \setminus E)$ and 2) $d_M(E,G) = D(E)$.

1) If $d_M(E,G) = D(G \setminus E)$ and $A(E) < A(G \setminus E)$, we translate l till another line l' determining a new division of G into two other regions E' and $G \setminus E'$ in such a way that one of the two following situations hold:

1.1) $A(E') = A(G \setminus E')$ and $d_M(E',G) = D(G \setminus E')$.

Then $A(E',G) = A(E') > A(E) = A(E,G)$ and $d_M(E',G) < d_M(E,G)$ and E' determines a fencing problem. Hence,

$$\frac{A(E,G)}{d_M(E,G)^2} \leq \frac{A(E')}{d_M(E',G)^2} \leq 1.2869...,$$

(For the last inequality see [5] and [6])

1.2) $A(E') < A(G \setminus E')$ and $d_M(E',G) = D(E') = D(G \setminus E')$.

In this case we have $A(E,G) = A(E) \leq A(E') = A(E',G)$ and $d_M(E,G) \geq d_M(E',G)$, so:

$$\frac{A(E,G)}{d_M(E,G)^2} \leq \frac{A(E')}{D(E')^2}, \tag{2}$$

and also,

$$\frac{A(E,G)}{d_M(E,G)^2} \leq \frac{A(G \setminus E')}{D(G \setminus E')^2}. \tag{3}$$

Now we consider the intersection points P and Q of l' with ∂G.

Let E'' be either E' or $G \setminus E'$ where the supporting lines at P and Q make internal angles whose sum is smaller or equal than π. Let us consider the symmetric set of E'' with respect to the middle point O of the segment PQ. Let this set be E'''. $E'' \cup E'''$ is a centrally symmetric convex set where the area is $2A(E'')$. It is easy to see that:

$$D(E''') = D(E'') = d_M(E',G) =$$
$$= D(E') = D(G \setminus E').$$

Then, from inequalities (2) and (3),

$$\frac{A(E,G)}{d_M(E,G)^2} \leq \frac{A(E'')}{D(E'')^2} \leq 1.2869...,$$

(For the last inequality see [5]).

2) Suppose that $d_M(E,G) = D(E)$. Then,

$$\frac{A(E,G)}{d_M(E,G)^2} = \frac{A(E)}{D(E)^2},$$

and also,

$$\frac{A(E,G)}{d_M(E,G)^2} \leq \frac{A(G \setminus E)}{D(G \setminus E)^2}.$$

Let E' be either E or $G \setminus E$, where the supporting lines at P and Q realize internal angles whose sum is smaller or equal than π. By a similar argument to that used in the case 1.2 we conclude that

$$\frac{A(E,G)}{d_M(E,G)^2} \leq \frac{A(E'')}{D(E'')^2} \leq 1.2869...$$

□

References

[1] Bieberbach, A.: *Über* eine Extremaleigenschaft des Kreises, *Jber. Deutsch. Math.-Vereinig* **24** (1915), 247–250.

[2] Cianchi, A.: On relative isoperimetric inequalities in the plane, *Boll. Unione Mat. Italiana* **7** 3-B (1989), 289–325.

[3] Croft, H. T., Falconer, K. J. and Guy, R. K.: *Unsolved problems in Geometry*, Springer, Berlin, 1991.

[4] Cerdán, A., Schnell, U. and Segura Gomis, S.: On relative geometric inequalities, *to appear in Math. Ineq. and Appl.* 2003.

[5] Miori, C., Peri, C. and Segura Gomis, S.: On fencing problems, *Quaderno 23, Universitá Cattolica del Sacro Cuore di Brescia*, 2003.

[6] Miori, C., Peri, C. and Segura Gomis, S.: On fencing problems, *Preprint.*

[7] Peri, C.: On relative isoperimetric inequalities, *Conferenze del Seminario di Matematica dell'Università di Bari*, **279**, 2000.

Point set stratification and minimum weight structures

Manuel Abellanas [a,2], Mercè Claverol [*,b,1], Ferran Hurtado [c,1],
[a] *Dpt. Mat. Aplic., FI. UPM. , Boadilla del Monte, 28660 Madrid, Spain*
[b] *Dpt. Mat. Aplic. IV, UPC. EPSEVG. Vilanova i La Geltrú 08800, Spain*
[c] *Dpt. Mat. Aplic. II, UPC. FIB. FME., 08028 Barcelona, Spain*

Abstract

Three different concepts of depth in a point set are considered and compared: Convex depth, location depth and Delaunay depth. As a notion of weight is naturally associated to each depth definition, we also present results on minimum weight structures (like spanning trees, poligonizations and triangulations) with respect to the three variations.

Key words: Tukey depth, halfspace depth, convex depth, Delaunay depth, minimum weight, layers.

1. Introduction

In bivariate data analysis, different ways of partitioning data sets as well as peeling methods (mainly for outlier rejection) have been proposed by different authors according to several definitions of *depth*. Every notion of depth of a point with respect to a point set S gives rise to a partition of the set S into *layers* and also to a partition of the whole plane into *levels*. The layers are the subsets of points of S having the same depth. The level of a point q with respect to S is the depth of q in $S \cup \{q\}$. The *weight* of a segment is the absolute value of the difference of depth between its vertices. There are also several ways to associate weights to geometric structures defined by the points of S and some set of edges (segments with endpoints in S), as described in Section 4.

In [10](pg. 363) Okabe et al. mention the interest in comparing Delaunay depth with respect to other depths. In this paper we do a comparative study of properties of layers and levels associated to finite sets of points in the plane considering three different definitions of depth: *convex depth* [8], *Tukey depth* also know as *halfspace depth* or *location depth* [11] and *Delaunay depth* introduced by Green in [7]. A thorough study is presented in [5].

After introducing basic definitions in Section 2, we study and compare the complexity of layers and levels in Section 3, and we give in Section 4 properties of minimum weight structures (spanning trees, poligonizations and triangulations) related to the three depths considered.

2. Preliminaries

Let S be a set of n points in the plane, $CH(S)$ the convex hull of S and p any point of S. Any generic depth of p with respect to S is denoted by $d_S(p)$.

The *convex depth* of p, is defined recursively as follows: if $p \in CH(S)$, $d_S(p) = 1$, else $d_S(p) = d_{S \setminus CH(S)}(p) + 1$. For values of $j \leq \lfloor n/2 \rfloor$ we say that the *location depth* of p is $d_S(p) = j$ if and only if there is a line through p leaving exactly $j - 1$ points on one side, but no line through p separates a smaller subset. The *Delaunay depth* of p is defined to be $d + 1$ when the graph theoretical distance from p to $CH(S)$ in the Delaunay triangulation $DT(S)$ of S is d. In all three cases we call *depth of* S the depth of its deepest point.

* Corresponding author
Email addresses: mabellanas@fi.upm.es (Manuel Abellanas), merce@mat.upc.es (Mercè Claverol), Ferran.Hurtado@upc.es (Ferran Hurtado).
[1] Partially supported by *DURSY* 2001SGR00224 and *MCYT* BFM2003-0368
[2] Partially supported by *MCYT* TIC2003-08933-C02-01

	$d_S(p)$ (Depth)	$Lay_i(S)$ (Layer i)	$Lev_i(S)$ (Level i)	
Convex	if $p \in CH(S)$, $d_S(p) = 1$ else $d_S(p) = d_{S \setminus CH(S)}(p) + 1$	$Lay_i(S) = CH(S_i)$ $S_i = \{x \in S / d_S(x) = i\}$	Depth of a point relative to a set S $d(p, S) = d_{S \cup \{p\}}(p)$	$Lev_i(S)$= $\{x \in \mathbb{R}^2 / d(x, S) = i\}$
Location depth	$d_S(p) = j, \quad j \leq \lfloor \lvert S \rvert / 2 \rfloor \Leftrightarrow$ some line through p leaves exactly $j-1$ points on one side, none leaves less			
Delaunay	if $p \in CH(S)$, $d_S(p) = 1$ else $d_S(p) =$ distance from p to $CH(S)$ $+1$, in $DT(S)$	$Lay_i(S)$ =subgraph of $DT(S)$ induced by S_i $S_i = \{x \in S \lvert d_S(x) = i\}$		

Table 1: Definitions

The i-th *layer* of S, $Lay_i(S)$, is defined for convex depth as well as for location depth by $Lay_i(S) = CH(S_i)$, where $S_i = \{x \in S \mid d_S(x) = i\}$, (Figures 1 and 2). For the Delaunay depth, $Lay_i(S)$ is the subgraph of $DT(S)$ induced by S_i, (Figure 3).

Let p be any point in the plane. For the three depths considered, the depth of p relative to the set S is $d(p, S) = d_{S \cup \{p\}}(p)$ and the i-th *level* for the set S is defined by $Lev_i(S) = \{x \in \mathbb{R}^2 \lvert d(x, S) = i\}$. The concept of k-hull introduced by Cole, Sharir and Yap in [6] corresponds to $\bigcup_{j \geq k} Lev_j(S)$.

Table 1 shows all these definitions together.

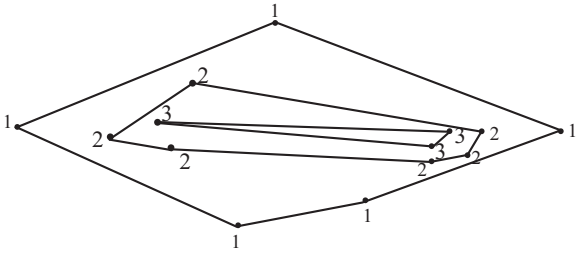


Fig. 1. Convex layers.

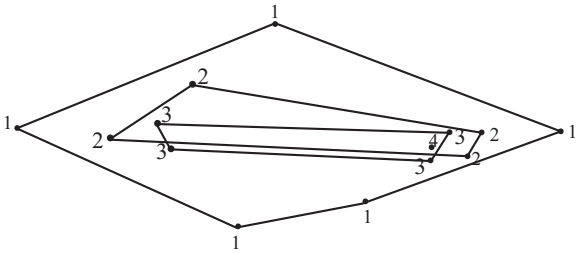


Fig. 2. Location layers.

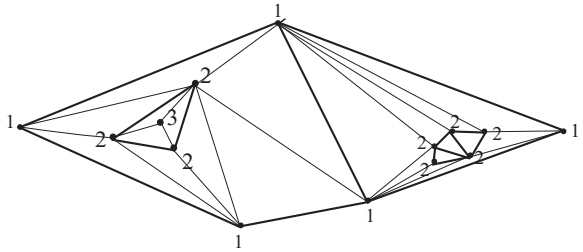


Fig. 3. Delaunay layers.

3. Point set stratification

Given a set S of n points in the plane the convex layers can be constructed with Chazelle's optimal $O(n \log n)$ algorithm [4]. Convex layers form a sequence of nested convex polygons defining a partition of the plane into regions, which coincide with the levels, (Figure 1). Therefore layers and levels have linear complexity in the convex depth case and can be constructed in optimal $O(n \log n)$ time.

As for location depth, a worst case optimal algorithm for computing all $Lev_i(S)$, (where $n/3 \leq i \leq n/2$) in $O(n^2)$ time is obtained by using topological sweep in the dual arrangement of lines (see [5], [9]). The boundaries of the levels, in this case, form a sequence of nested convex polygons. Points of $Lay_i(S)$ are in convex position and belong to the boundary of $Lev_i(S)$, but this boundary can also have other vertices not in S, (Figure 4). Some layers can be empty and different layers can cross each other. While the complexity of levels may reach $O(n^2)$, the size of the layers is $O(n)$. The layers in the location depth case can be computed using the mentioned $O(n^2)$ sweep algorithm yet, to our knowledge, it is an open problem to construct them in less time or to prove a quadratic lower bound for the problem.

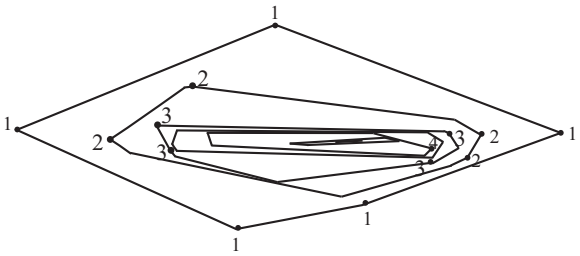


Fig. 4. Location levels.

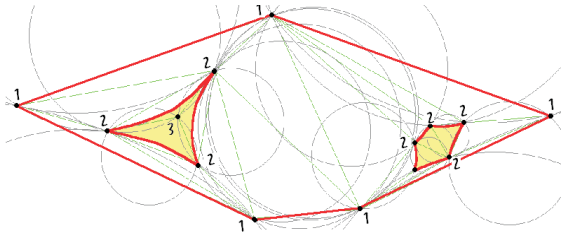


Fig. 5. Delaunay levels.

For Delaunay depth, all the layers $Lay_i(S)$, $i \leq n/3$, can easily be found by visiting $DT(S)$ in linear time once constructed, which requires $O(n \log n)$ time (Figure 3). Notice that one layer can have more than one connected component. The maximum number of connected components is strictly decreasing on the number of layers, (see [5]). This maximum number varies between $\lfloor (n+2)/3 \rfloor$ and $\lfloor n/2 \rfloor$, attained with suitable constructions having $\lfloor n/3 \rfloor$ and 2 layers, respectively.

Delaunay layers are not necessarily polygons, however they form a structure based in nested cycles of points of the same weight.

The weight of a point relative to a set S depends on the Delaunay circles (circumcircles of Delaunay triangles) that contain the point, therefore the arrangement of Delaunay circles contains all the information about Delaunay levels, (Figure 5). As it has size $O(n^2)$ and can be constructed in $O(n^2 \log n)$ time, it is possible to obtain the Delaunay levels within this time. Nevertheless, in the following theorem we prove that in order to obtain all the $Lev_i(S)$ it is not necessary to construct the whole arrangement of circles.

Lemma 1 *Let S be a set of points in the plane. If the Delaunay depth of a point p with respect to S is j, there is a cycle of $Lay_{j-1}(S)$ containing p in its interior.*

As a consequence, the number of levels for Delaunay depth is equal to the number of layers or to the number of layers plus one.

Observation 2 *Let C be a circle having exactly two points u and v of S on its boundary and containing no points of S in its interior. Then any circle crossing the two arcs determined by u and v in the boundary of C contains some interior point from S.*

Theorem 3 *Let S be a set of points in the plane being f its Delaunay depth . The Delaunay levels of S are nested sets. The boundaries between $Lev_j(S)$ and $Lev_{j+1}(S)$, for $2 \leq j \leq f$, are curves composed by arcs of the Delaunay circles determined by two points u, v of $Lay_j(S)$ and one point w of $Lay_{j-1}(S)$.*

Theorem 3 proves that the overall size of the Delaunay levels is $O(n)$ and justifies the steps of the following algorithm.

Algorithm 1 DELAUNAY LEVELS OF S.

1. *Compute $DT(S)$.*
2. *Compute the Delaunay depths for all points in S.*
3. *Compute the boundaries of the levels as follows: $Lev_1(S)$ is the convex hull of S; for every $j \geq 2$, the Delaunay circles C_j, defined by two points u, v of $Lay_j(S)$ and one point w of $Lay_{j-1}(S)$, determine the boundary between $Lev_j(S)$ and $Lev_{j+1}(S)$ which consists of the inner boundary of the union of Delaunay circles C_j (Figure 6).*

The running time of the algorithm is $O(n \log^2 n)$: $DT(S)$ can be computed in $O(n \log n)$ time, Step 2 only takes $O(n)$ time, and every boundary in Step 3 can be computed in $O(n_i \log^2 n_i)$ time where n_i is the number of Delaunay circles C_j considered in the corresponding layer (see [12] pg. 97). Taking into account that the total number of Delaunay circles is $O(n)$, Step 3 takes $O(n \log^2 n)$ time. Note that the expected time for Step 3 is $O(n \log n)$ [12], and therefore, the expected time for the entire algorithm is $O(n \log n)$.

4. Minimum weight structures

A notion of weight related to the concept of depth was associated in [2] to geometric structures consisting of segments (edges) with endpoints in S. The weight of a geometric structure admits two natural definitions.

Definition 4 *The total weight,* t-weight*, of a geometric structure, is the sum of the weights of its edges.*

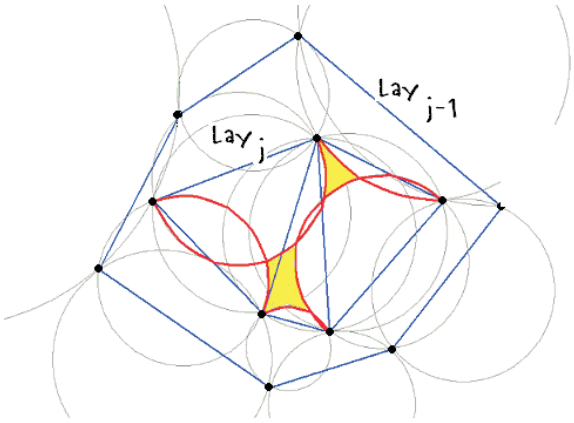


Fig. 6. The shaded region is $Lev_{j+1}(S)$.

Definition 5 *The worse weight,* w-weight*, of a geometric structure, is the weight of its heaviest edge.*

In this paper we focus on minimum t-weight structures.

4.1. *Minimum t-weight spanning trees*

An algorithm for constructing a minimum t-weight spanning tree ($MWST$), can be easily obtained by taking for each weight a spanning tree of the set of points of that weight and connecting whith an edge every pair of trees whose weight differs by one.

In [5] the following combinatorial result on the number of $MWST$ for a set of points is proved for convex depth. The result extends to Delaunay weight but does not extend to the case of location weight, since the vertices of $Lay_k(S)$ may be points not in S.

Proposition 6 *Let S be a set of n points and let C_i be the cardinality of layer i. For convex depth and Delaunay depth, the number of $MWST$ in S is*

$$f(C_1, \cdots, C_k) = \frac{C_1^{C_1} \cdot C_2^{C_2} \cdots C_{k-1}^{C_{k-1}} \cdot C_k^{C_k}}{C_1 \cdot C_k}.$$

The number of $MWST$ is strictly decreasing on the number of layers. The number of $MWST$ varies between n^{n-2}, attainable in the case of one layer, and 3^{n-2}, attainable when the number of layers is $k = (n+2)/3$. For a fixed number of layers, the maximum number of $MWST$ is attained when the last layer has one point and all the others have 3 points except possibly one (this layer cannot be the first except for the case of two layers).

4.2. *Minimum t-weight polygonizations*

Minimum weight polygonizations in the convex case are studied in [2], where next proposition is proved. This result also applies to each of the weights we are considering, the proof is similar in all the cases.

Proposition 7 *Let S be a set of n points with depth f. Every polygonization of S has weight greater or equal to $2f-2$. The value $2f-2$ is attained if and only if the sequence of depths of the boundary vertices is (circularly) unimodal.*

In [2] and [3] certain polygons are obtained with good computational properties. In particular, after the removal of any number of consecutive layers, the remaining set can be polygonized in constant time. These polygonizations, called *onion polygonizations*, achieve the minimum weight and furthermore, thanks to the convexity of the layers, are always constructible for any set of points. In the case of the Delaunay weight, we do not always have polygonizations of weight $2f-2$ (see [5]). This does not depend on the number of connected components of the Delaunay layers. Examples can be found of polygonizations with weight $2f-2$ when the number of components is as big as possible (Figure 7).

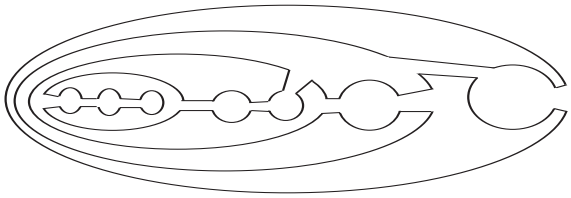


Fig. 7. A polygonization scheme with Delaunay weight $2f-2$ for $f=5$. The layers of the point set have 11 connected components.

Proposition 8 *The minimum Delaunay weight of any polygonization of a set of n points is never greater than n. There are examples of sets in which the weight of the minimum weight polygonization is $\Omega(n)$.*

4.3. *Minimum t-weight triangulations*

For the convex depth case we have obtained the following algorithm, which runs in $O(n \log n)$ time and generates triangulations that contain the convex layers.

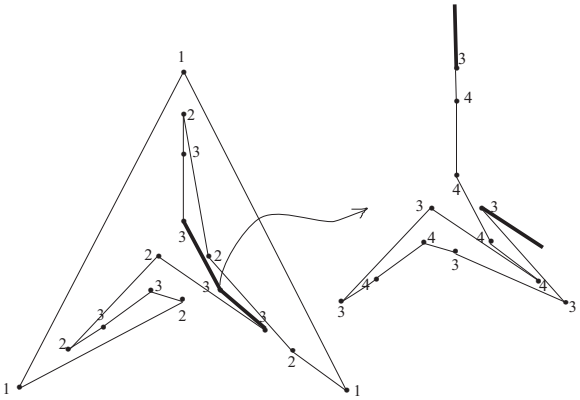


Fig. 8. Every polygonization of this point set has weight $\Omega(n)$.

Algorithm 2 *[5]* SMALL WEIGHT TRIANGULATION T OF A SET S OF n POINTS.

1. *Include in T all the edges of the convex layers.*
2. *For every convex layer find a minimal subpolygon P that contains the next inner layer; add to T the edges of P.*
3. *Complete T in any way by triangulating the remaining holes.*

It is worth noticing that there are other triangulations with the same weight that the triangulations obtained as output of the above algorithm, as shown in Figure 9.

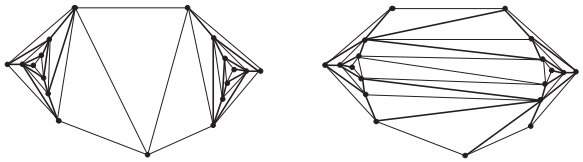


Fig. 9. Two triangulations with the same weight; the right one contains the convex layers. Solid edges have weight 1, light edges have weight 0.

We conjecture that the triangulations obtained using Algorithm 2 are always minimum weight triangulations. We have proved that this is the case for sets with two layers and also when all layers are triangles (except possibly the innermost one), yet a general proof remains elusive to us.

The study of the minimum t-weight triangulations for location depth and Delaunay depth is ongoing work. One would expect the Delaunay triangulation to have minimum Delaunay t-weight, but this is not always the case (see figure 10); however the Delaunay triangulation has minimum Delaunay w-weigt.

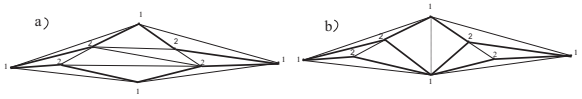


Fig. 10. a) Triangulation of minimum t-weight 8. b) Delaunay triangulation of the same point set as to the left, with t-weight 10, which is not minimum.

References

[1] G. Aloupis, C. Cortés, F. Gómez, M. Soss, G. Toussaint. Lower bounds for computing statistical depth. *Computational Statistics & Data Analysis* **40** (2002) 223–229.

[2] M. Abellanas, J. García, G.Hernández, F. Hurtado, O. Serra, J. Urrutia. Updating Polygonizations, *Computer Graphics Forum* **vol. 12, n.3** (1993) 143–152.

[3] M. Abellanas, J. García, G.Hernández, F. Hurtado, O. Serra, J. Urrutia. Onion Polygonizations, *Information Processing Letters* **vol. 57** (1996) 165–173.

[4] B. Chazelle. On the Convex Layers of a Planar Set, *IEEE Transactions on Information Theory* **vol. IT-31, n.4** (1985) 509–517.

[5] M. Claverol, Geometric Problems on Computational Morphology, *Ph. D Thesis in preparation*.

[6] R. Cole, M. Sharir, C.K. Yap. On K-hulls and related problems, *SIAM Journal of Computing* **vol. 16** (1987) 61–77.

[7] P.J. Green. Peeling Bivariate Data in: *Interpreting multivariate data*. John Wiley y Sons Ltd., (1981) 163–198.

[8] P. J. Hubert. Robust statistics: a review. *Ann. Math. Statistics* **43 n.3** (1972) 1041–1067.

[9] H. K. Miller, S. Ramaswami, P. Rousseeuw, T. Sellares, D. Souvaine, I. Streinu, A. Struyf. Efficient computation of location depth contours by methods of computational geometry, *Statistics and Computing* (2003).

[10] A. Okabe, B. Boots, K. Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, John Wiley and Sons Ltd. Chichester (1992). 362–363.

[11] J.W. Tukey, Mathematics and the picturing of data, *Proceedings of the International Congress of Mathematicians* **2** (1975) 523–531.

[12] P. Agarwal, M. Sharir, Arrangements and its applications, in *Handbook of Comput. Geom.*, J.R. Sack and J. Urrutia eds., North-Holland 2000.

Smoothed Number of Extreme Points under Uniform Noise

Valentina Damerow [a,b], Christian Sohler [b,1],
[a] *PaSCo Graduate School and*
[b] *HEINZ NIXDORF INSTITUTE, Computer Science Department, University of Paderborn, D-33102 Paderborn, Germany*

Abstract

We analyze the maximal expected number of extreme points of a point set P in $\mathbb{R}^d$ that is slightly perturbed by random noise. We assume that each point in P is uniformly distributed in an axis-aligned hypercube of side length 2ϵ centered in the unit hypercube (the center of the hypercube can be regarded as the point position without noise). Our model is motivated by the fact that in many applications the input data is inherently noisy, e.g. when the data comes from physical measurement or imprecise arithmetic is used. For this input distribution we derive an upper bound of $\mathcal{O}((n \cdot \log n/\epsilon)^{1-1/(d+1)})$ on the number of extreme points of P.

Key words: Randomization, Smoothed Analysis

1. Introduction

The convex hull of a point set in the d-dimensional Euclidean space is one of the fundamental combinatorial structures in Computational Geometry. Many of its properties have been studied extensively in the last decades. In this paper we are interested in the number of vertices of the convex hull of a random point set, sometimes referred to as the *number of extreme points* of the point set. It is known since nearly 30 years that the number of extreme points of a point set drawn uniformly at random from the (unit) hypercube is $\mathcal{O}(\log^{d-1} n)$, cf. [1]. The number of extreme points has also been studied for many other input distributions, e.g. for Gaussian normal distribution. In this paper we consider the expected number of extreme points when each input point is chosen from a (possibly) different small subcube of the unit hypercube. We can think of this input distribution as resulting from some point set P (defined by the centers of the subcubes), where each point is afflicted with some small random noise. Our model is motivated by the fact that in many applications the input data is inherently noisy, e.g. when the data comes from physical measurement or imprecise arithmetic is used.

1.1. *Related Work*

Several authors have treated the structure of the convex hull of n random points. In 1963/64 Rényi and Sulanke, [8] and [9], were the first to present the mean number of extreme points in the planar case for different stochastic models. They showed for n points uniformly chosen from a convex polygone with r vertices a bound of $r \cdot \log n$. This work was continued by Efron [4], Raynaud [6] and [7], and Carnal [2] and extended to higher dimensions. For further information we refer to the excellent book [10] by Santaló.

In 1978 Bentley, Kung, Schkolnick and Thompson [1] showed that the expected number of extreme points of n i.i.d. random points in d space is $\mathcal{O}(\ln^{d-1} n)$ for fixed d for some general probability distributions. Har-Peled gave in [5] a different proof of this result. Both results are based on the computation of the expected number of maximal points (cf. Section 3).

The concept of smoothed analysis was introduced in 2001 by Spielman and Teng [11]. In 2003

Email addresses: vio@upb.de (Valentina Damerow), csohler@upb.de (Christian Sohler).

[1] Partially supported by DFG project 872-8/2 and IST program of the EU under contract IST-1999-14186 (ALCOM-FT).

this concept was applied to the number of changes to the combinatorial describtion of the smallest enclosing bounding box of a moving point set [3] where different probability distributions for the random noise were considered. This work already covers the two dimensional case of this paper's problem. By considering moving points the two dimensional problem was reduced to a one dimensional probelm. The general extension to higher dimensions holds some non trivial difficulties even for the case when uniformle distributed noise is considered.

2. Problem Statement and Uniform Case

Let $P := \{p_1, \ldots, p_n\}$ denote a point set in the unit hypercube $[0,1]^d$ and let $\mathcal{V}(P)$ be the number of *extreme points* (i.e., the number of vertices of the convex hull) of P. Furthermore, let $r_1, \ldots, r_n$ be i.i.d. random vectors chosen uniformly at random from $[-\epsilon, \epsilon]^d$ and let $\widetilde{p_1} := p_1 + r_1, \ldots, \widetilde{p_n} := p_n + r_n$ be the perturbed points and $\widetilde{P}$ the set of perturbed points. Then we define the *smoothed number* of extreme points to be $\mathcal{V}(\widetilde{P}) := \max_P \mathbf{E}[\mathcal{V}(\widetilde{P})]$.

Our bounds are based on the following observation: A point $p \in P$ is not extreme, if each of the 2^d orthants centered at p contains at least one point. In this case, we say that p is not *maximal*. It follows immediatly that the number of maximal points is an upper bound on the number of extreme points. Therefore, we will from now on count the number of maximal points.

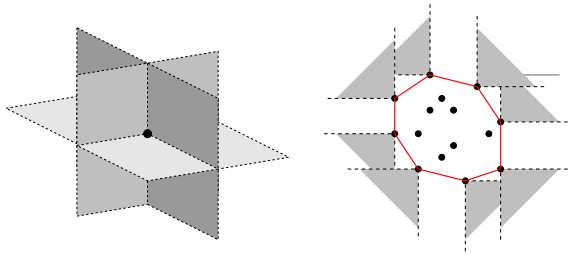


Fig. 1. 'A point in three dimensional space has eight orthants.' *and* 'Every extreme point is also a maximal point.'

As a warm-up we illustrate our approach on the well-understood case of n points chosen uniformly at random from the d-dimensional unit hypercube. We show how to obtain in this case an upper bound of $\mathcal{O}(\log^{d-1} n)$ on the number of maximal points and hence on the number of extreme points.

Theorem 1 *Let $P = \{p_1, \ldots p_n\}$ be a set of n points chosen uniformly at random from the d-dimensional unit hypercube. Then the expected number of extreme points of P is $\mathcal{O}(\log^{d-1} n)$ for fixed dimension d.*

Proof : To prove the theorem we show that

$$\mathbf{Pr}[p_i \text{ is maximal}] = \mathcal{O}(\log^{d-1} n/n) \ . \qquad (1)$$

By linearity of expectation it follows immediately that the number of extreme points is $\mathcal{O}(\log^{d-1} n)$. To prove (1) we consider the probability that a fixed orthant $\emptyset(p_i)$ centered at p_i is empty. Using a standard union bound we get

$$\mathbf{Pr}[p_i \text{ is maximal}] \leq 2^d \cdot \mathbf{Pr}[\emptyset(p_i) \text{ is empty}] \ .$$

Wlog. we now fix orthant $\emptyset(p_i) := \prod_{j=1}^{d} [-\infty, p_i^{(j)}]$. We can write the probability that $\emptyset(p_i)$ is empty as an integral in the following way: consider p_i having the coordinates $(x^{(1)}, \ldots, x^{(d)})$. The probability for any other point $p_k \in P \setminus \{p_i\}$ to be not in $\emptyset(p_i)$ is then equal to $1 - x^{(1)} \cdot x^{(2)} \cdots x^{(d)}$. Since there are $n-1$ other points in P the probability that $\emptyset(p_i)$ is empty is exactly

$$\int_0^1 \cdots \int_0^1 (1 - x^{(1)} \cdots x^{(d)})^{n-1} \mathrm{d}x^{(1)} \cdots \mathrm{d}x^{(d)} \ . \qquad (2)$$

We solve this integral by repeatetd substitution and demonstrate this on the 2 dimensional integral. We start with the integral $\int_0^1 \int_0^1 (1-xy)^{n-1} \mathrm{d}x\mathrm{d}y$ and substitute in a first step $1 - xy =: z = z(x)$ which gives us $\mathrm{d}z = -y \cdot \mathrm{d}x$ and $z(0) = 1$ and $z(1) = 1 - y$:

$$\int_0^1 \int_0^1 (1-xy)^{n-1} \mathrm{d}x\mathrm{d}y = \int_0^1 \int_{1-y}^1 \frac{z^{n-1}}{y} \mathrm{d}z\mathrm{d}y =$$
$$\int_0^1 \left[\frac{1}{n} \cdot \frac{z^n}{y}\right]_{1-y}^1 \mathrm{d}y = \frac{1}{n} \int_0^1 \frac{1}{y} \cdot (1 - \underbrace{(1-y)}_{=:\, z}^n) \mathrm{d}y \ .$$

Now we substitute $1 - y =: z = z(y)$ and we get $\mathrm{d}z = -1 \cdot \mathrm{d}y$ and $z(0) = 1$ and $z(1) = 0$:

$$\frac{1}{n} \int_0^1 \frac{1}{1-z} \cdot (1 - z^n) \mathrm{d}z = \frac{1}{n} \int_0^1 \sum_{i=0}^{n-1} z^i \mathrm{d}z =$$
$$\frac{1}{n} \left[\sum_{i=0}^{n-1} \frac{z^{i+1}}{i+1}\right]_0^1 = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{i} = \frac{\log n + \mathcal{O}(1)}{n} \ .$$

Since there are 4 quadrants and n points it follows that the expected number of maximal points in the planar case is $\mathcal{O}(\log n)$.

The d-dimensional integral (2) boils down to the sum

$$\frac{1}{n}\sum_{i_1=1}^{n}\frac{1}{i_1}\sum_{i_2=1}^{i_1}\frac{1}{i_2}\cdots\sum_{i_{d-1}=1}^{i_{d-2}}\frac{1}{i_{d-1}}=\mathcal{O}\left(\frac{\log^{d-1}n}{n}\right),$$

which proves the theorem. □

3. Smoothed Number of Extreme Points

We now want to apply the same approach to obtain an upper bound on the smoothed number of extreme points. We consider again a perturbed point $\widetilde{p_i} = p_i + r_i$, where $p_i = (p_i^{(1)}, \ldots, p_i^{(d)}) \in [0,1]^d$ and r_i a random vector chosen uniformly from $[-\epsilon, \epsilon]^d$. It follows that $\widetilde{p_i}$ lies in the hypercube $\prod_{j=1}^{d}[p_i^{(j)} - \epsilon, p_i^{(j)} + \epsilon] =: \mathrm{phc}(p_i)$ which we want to call the perturbation hypercube of point p_i.

Now we recall that $\emptyset(\widetilde{p_i}) = \prod_{j=1}^{d}[-\infty, \widetilde{p_i}^{(j)}]$. For any other perturbed point $\widetilde{p_k} = p_k + r_k$, $i \neq k$, the probability that $\widetilde{p_k}$ does not lie in $\emptyset(\widetilde{p_i})$ is

$$\mathbf{Pr}[\widetilde{p_k} \notin \emptyset(\widetilde{p_i})] = \int_{\mathcal{I}(\widetilde{p_i},p_k)}\left(\frac{1}{2\epsilon}\right)^d \mathrm{d}y$$

where $\mathcal{I}(\widetilde{p_i}, p_k)$ is the set of all valid positions for $\widetilde{p_k}$ not lying in $\emptyset(\widetilde{p_i})$, i.e. $\mathcal{I}(\widetilde{p_i}, p_k) = \mathrm{phc}(p_k) - \emptyset(\widetilde{p_i})$. Note that $1/(2\epsilon)^d$ is the probability density of $\widetilde{p_k}$. We get

$$\mathbf{Pr}[\emptyset(\widetilde{p_i}) \text{ is empty}] =$$

$$\int_{\mathrm{phc}(p_i)}\left(\frac{1}{2\epsilon}\right)^d \cdot \left(\prod_{k\neq i}\int_{\mathcal{I}(x,p_k)}\left(\frac{1}{2\epsilon}\right)^d \mathrm{d}y\right)\mathrm{d}x \ .$$

The main idea is now to subdivide the unit hypercube into $m = 1/\delta^d$ smaller axis-aligned hypercubes of sidelength δ. Then we subdivide P into sets $C_1, \ldots, C_m$ where C_ℓ is the subset of P that is located (before the perturbation) in the ℓ-th small hypercube (we assume some ordering among the small hypercubes). Now we can calculate the expected number $\mathcal{D}(C_\ell)$ of maximal points for the sets C_ℓ and use

$$\mathcal{V}(\widetilde{P}) \ \leq \ \sum_{\ell=1}^{m}\mathcal{V}(\widetilde{C_\ell}) \ \leq \ \sum_{\ell=1}^{m}\mathcal{D}(\widetilde{C_\ell}) \qquad (3)$$

to obtain an upper bound on the expected number of extreme points in $\widetilde{P}$. The advantage of this approach is that for small enough δ the points in a single small hypercube behave almost as in the uniform random case.

We now want to compute the expected number of extreme points for the sets C_ℓ. We assume wlog. that C_ℓ is the hypercube $[0,\delta]^d$. Let $\bar{\delta} = (\delta, \ldots, \delta)$ and $\bar{0} = (0, \ldots, 0)$ etc. We now want to find an upper bound on the probability that $\emptyset(\widetilde{p_i})$ is empty. This probability is maximized, if $p_i = \bar{0}$ and $p_k = \bar{\delta}$ for every $p_k \in C_\ell$, $i \neq k$. Hence, we get

$$\mathbf{Pr}[\emptyset(\widetilde{p_i}) \text{ is empty}] \leq$$

$$\int_{\mathrm{phc}(\bar{0})}\left(\frac{1}{2\epsilon}\right)^d \cdot \left(\int_{\mathcal{I}(x,\bar{\delta})}\left(\frac{1}{2\epsilon}\right)^d \mathrm{d}y\right)^{n-1}\mathrm{d}x \ .$$

Using $\mathcal{I}(x,\bar{\delta}) = [\delta-\epsilon, \delta+\epsilon]^d - \prod_{j=1}^{d}[-\infty, x^{(j)}] = \prod_{j=1}^{d}[\max\{\delta-\epsilon, x^{(j)}\}, \delta+\epsilon]$ we can write the integral as

$$\int_{[-\epsilon,\epsilon]^d}\left(\frac{1}{2\epsilon}\right)^d \cdot \left(\sum_{\mathrm{I}\subseteq[d]}(-1)^{|\mathrm{I}|+1}\cdot \int_{\max\{(\delta-\epsilon)_\mathrm{I}, x_\mathrm{I}\}}^{(\delta+\epsilon)_\mathrm{I}}\left(\frac{1}{2\epsilon}\right)^{|\mathrm{I}|}\mathrm{d}y_\mathrm{I}\right)^{n-1}\mathrm{d}x \qquad (4)$$

where the subscript I for a variable denotes the $|\mathrm{I}|$ dimensional projection of the variable to the coordinates in I, i.e. $x_\mathrm{I} = (x^{(j_1)}, \ldots, x^{(j_t)})$ for $\mathrm{I} = \{j_1, \ldots, j_t\}$, and the maximum is taken coordinate wise. Let

$$\mathrm{F}(x,j) := \sum_{\mathrm{I}\subseteq[d-j]}(-1)^{|\mathrm{I}|+1}\int_{(\delta-\epsilon)_\mathrm{I}}^{(\delta+\epsilon)_\mathrm{I}}\left(\frac{1}{2\epsilon}\right)^{|\mathrm{I}|}\mathrm{d}y_\mathrm{I} + \sum_{\mathrm{I}\subseteq[j]}(-1)^{|\mathrm{I}|+1}\int_{x_\mathrm{I}}^{(\delta+\epsilon)_\mathrm{I}}\left(\frac{1}{2\epsilon}\right)^{|\mathrm{I}|}\mathrm{d}y_\mathrm{I} \ .$$

We can rewrite the integral (4) now in the form

$$\sum_{j=0}^{d}\binom{d}{j}\cdot\underbrace{\int_{-\epsilon}^{\delta-\epsilon}\cdots\int_{-\epsilon}^{\delta-\epsilon}}_{d-j}\underbrace{\int_{\delta-\epsilon}^{\epsilon}\cdots\int_{\delta-\epsilon}^{\epsilon}}_{j}\left(\frac{1}{2\epsilon}\right)^d\cdot\mathrm{F}(x,j)^{n-1}\mathrm{d}x \ . \qquad (5)$$

Next we want to calculate this integral. Again we will use repeated substitution. We obtain the

following result, which we prove only for the one dimensional case (the complete proof is deferred to the full version of this paper).

Lemma 2 *Let $C_\ell \subseteq P$ be a set of c_ℓ points in the hypercube $[0,\delta]^d$ before the perturbation takes place. The smoothed number of extreme points in $\widetilde{C}_\ell$ is at most*

$$\mathcal{V}(\widetilde{C}_\ell) \;\leq\; \mathcal{D}(\widetilde{C}_\ell) \;\leq\; c_\ell \cdot 2^d \cdot \left(d \cdot \frac{\delta}{2\epsilon} + \frac{\log^{d-1} c_\ell}{c_\ell} \right) \;.$$

Proof : *(only the 1-dimensional case)*

Let us consider the one dimensional integral. From (5) we have

$$\int_{-\epsilon}^{\delta-\epsilon} \frac{1}{2\epsilon} \cdot \left(\int_{\delta-\epsilon}^{\delta+\epsilon} \frac{1}{2\epsilon} \mathrm{d}y \right)^{n-1} \mathrm{d}x + \int_{\delta-\epsilon}^{\epsilon} \frac{1}{2\epsilon} \cdot \left(\int_{x}^{\delta+\epsilon} \frac{1}{2\epsilon} \mathrm{d}y \right)^{n-1} \mathrm{d}x \;=$$

$$\int_{-\epsilon}^{\delta-\epsilon} \frac{1}{2\epsilon} \mathrm{d}x \;+\; \int_{\delta-\epsilon}^{\epsilon} \frac{1}{2\epsilon} \cdot \left(\frac{1}{2\epsilon} \cdot \underbrace{(\delta+\epsilon-x)}_{=:\, z} \right)^{n-1} \mathrm{d}x \;.$$

The first integral solves to $\delta/(2\epsilon)$. In the second integral we substitute $\delta+\epsilon-x =: z = z(x)$ and we get $\mathrm{d}z = -1 \cdot \mathrm{d}x$ and $z(\delta-\epsilon) = 2\epsilon$ and $z(\epsilon) = \delta$. Thus it is

$$\int_{\delta-\epsilon}^{\epsilon} \frac{1}{2\epsilon} \cdot \left(\frac{1}{2\epsilon} \cdot (\delta+\epsilon-x) \right)^{n-1} \mathrm{d}x \;=$$

$$\int_{\delta}^{2\epsilon} \left(\frac{1}{2\epsilon} \right)^{n} \cdot z^{n-1} \, \mathrm{d}z \;=\; \frac{1}{n} \cdot \left(1 - \left(\frac{\delta}{2\epsilon} \right)^{n} \right) \;.$$

It follows that in the one dimensional case the probability for a point to be maximal is at most

$$2 \cdot \left(\frac{\delta}{2\epsilon} + \frac{1}{n} \right) \;.$$

□

We can now conclude from (3) and Lemma 2 that

$$\mathcal{V}(\widetilde{P}) \;\leq\; \sum_{\ell=1}^{1/\delta^d} c_\ell \cdot 2^d \cdot \left(d \cdot \frac{\delta}{2\epsilon} + \frac{\log^{d-1} c_\ell}{c_\ell} \right) \;.$$

Our main theorem follows choosing

$$\delta = \mathcal{O}\left(\left(\frac{\epsilon \cdot \log^{d-1} n}{d \cdot n} \right)^{1/d} \right) \;.$$

Theorem 3 *The smoothed number of extreme points of a set $\widetilde{P}$ of n perturbed points in d dimensional space with start points from the unit hypercube and under uniform noise from $[-\epsilon,\epsilon]^d$ is*

$$\mathcal{V}(\widetilde{P}) = \mathcal{O}\left(\left(\frac{n \cdot \log n}{\epsilon} \right)^{1-\frac{1}{d+1}} \right) \;.$$

References

[1] Bentley, J. L., Kung, H. T., Schkolnick, M., and Thompson, C. D. On the average number of maxima in a set of vectors. *J. Assoc. Comput. Mach. 25* (1978), 536–543.

[2] Carnal, H. Die konvexe Hülle von n rotationssymmetrisch verteilten Punkten. *Z. Wahrscheinlichkeitsth. 15* (1970), 168–176.

[3] Damerow, V., Meyer auf der Heide, F., Räcke, H., Scheideler, C., and Sohler, C. Smoothed motion complexity. In *Proceedings of the 11th European Symposium on Algorithms (ESA)* (2003), pp. 161–171.

[4] Effron, B. The convex hull of a random set of points. *Biometrika 52* (1965), 331–343.

[5] Har-Peled, S. On the extected complexity of random convex hulls. Tech. Rep. 330, School Math. Sci., Tel-Aviv Univ., Tel-Aviv, Israel, 1998.

[6] Raynaud, H. Sur le comportement asymptotique de l'enveloppe convexe d'un nuage de points tirés au hasard dans R^n. *C. R. Acad. Sci. Paris 261* (1965), 627–629.

[7] Raynaud, H. Sur l'enveloppe convexe des nuages de points aléatoires dans R^n. *J. Appl. Prob. 7* (1970), 35–48.

[8] Rényi, A., and Sulanke, R. Über die konvexe Hülle von n zufällig gewählten Punkten I. *Z. Wahrscheinlichkeitsth. 2* (1963), 75–84.

[9] Rényi, A., and Sulanke, R. Über die konvexe Hülle von n zufällig gewählten Punkten II. *Z. Wahrscheinlichkeitsth. 3* (1964), 138–147.

[10] Santaló, L. A. *Integral Geometry and Geometric Probability.* Addison-Wesley Publishing Company, 1976.

[11] Spielman, D., and Teng, S. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. In *Proceedings of the 33rd ACM Symposium on Theory of Computing (STOC)* (2001), pp. 296–305.

Optimizing a 2D Function satisfying Unimodality Properties

Erik D. Demaine* Stefan Langerman†

1 Introduction

A general formulation of discrete optimization is to maximize a given function $f : D \to \mathbb{R}$ over a discrete (finite) domain D. In general, of course, this problem may require $|D|$ probes to f. One approach to making optimization more tractable is to be satisfied with finding a *local* maximum, i.e., a point at which f attains a value larger than all "neighboring" points, for some definition of neighborhoods. In particular, for the standard 1D domain $D = \{1, 2, \ldots, n\}$, Fibonacci search [Kie53] finds a local maximum using $\log_\phi n + O(1)$ probes, where $\phi = (1 + \sqrt{5})/2$ is the golden ratio. Surprisingly, even for a square 2D domain $D = \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\}$, the problem complexity grows exponentially: Mityagin [Mit03] proved that $\Theta(n)$ probes to such an f are sufficient and sometimes necessary. Thus weakening the problem to finding local maxima does not cause the exponential speedup from 1D in higher dimensions.

Another approach to making optimization more tractable is to add assumptions about the function f. For example, if we assume that f is *unimodal* (denoted "$\odot$ unimodal"), i.e., it has exactly one local maximum, then finding local maxima and finding global maxima are equivalent. One could hope that having this structural information about the function would also help in finding that maximum. Unfortunately, a careful reading of the construction in [Mit03] of 2D functions f requiring $\Theta(n)$ probes are in fact unimodal.

We study the related condition that the 2D function f is unimodal in every row ($\leftrightarrow$ unimodal) and/or in every column ($\updownarrow$ unimodal). (These properties are satisfied by e.g. convex functions.) While seemingly weaker than unimodality, these properties are incomparable to unimodality, and in fact result in exponential speedup for finding local maxima.

Table 1 summarizes all of our results. Our upper bounds all follow from a combination of linear search and/or Fibonacci search in each dimension. Matching local bounds for global optimization follow in some cases from independence of the columns. Some bounds are tight only up to logarithmic factors, leaving intriguing open questions. In the full paper, we provide the omitted proof and prove the comforting fact that a natural random probing algorithm makes $\Omega(\lg m \lg n)$ expected probes even for a convex function, as in our dual Fibonacci search.

2 Lower Bound for $\updownarrow$ Unimodal

Theorem 1 *If $n \geq m^\varepsilon$ where $0 < \varepsilon \leq 1$, there is an adversary that (a) generates $\updownarrow$-unimodal functions each with a unique local optimum, and (b) forces any local or global optimization algorithm to make $(\varepsilon^2/4) \lg m \lg n - O(\lg m \lg \lg n)$ probes.*

*MIT Computer Science and Artificial Intelligence Laboratory, 200 Technology Square, Cambridge, MA 02139, USA, edemaine@mit.edu

†Chargé de Recherche FNRS, Département d'informatique, Université Libre de Bruxelles, ULB CP212, Bruxelles, Belgium, Stefan.Langerman@ulb.ac.be

Assumption	Local optimization		Global optimization	
None	$\leq min(\lg \frac{max}{min} + 4) + O(\lg max)$	[Mit03]	$\leq m \cdot n$	[lin.×lin.]
	$\geq min\{min, max/2\}$	[Mit03]	$\geq m \cdot n$	[indep.]
$\odot$ unimodal	$\leq min(\lg \frac{max}{min} + 4) + O(\lg max)$	[Mit03]	same as local	
	$\geq min\{min, max/2\}$	[Mit03]		
$\updownarrow$ unimodal	$\leq \log_\phi m \lg n + O(\lg m)$	[Fib.×Fib.]	$\leq n \log_\phi m + O(n)$	[Fib.×lin.]
	$\geq (\varepsilon^2/4) \lg m \lg n - O(\lg m \lg \lg n)$ if $n \geq m^\varepsilon$ where $0 < \varepsilon \leq 1$	[Thm. 1]	$\geq n \log_\phi m + O(n)$	[indep.]
$\updownarrow$, $\leftrightarrow$ unimodal	$\leq \lg m \lg n / \lg \phi + O(\lg min)$	[Fib.×Fib.]	$\leq min \log_\phi max + O(min)$	[Fib.×lin.]
	$\geq \lg m + \lg n$	[info. theo.]	$\Omega(min)$	[omitted]
$\odot$, $\updownarrow$ unimodal	$\leq \log_\phi m \lg n + O(\lg m)$	[Fib.×Fib.]	same as local	
	$\geq (\varepsilon^2/4) \lg m \lg n - O(\lg m \lg \lg n)$ if $n \geq m^\varepsilon$ where $0 < \varepsilon \leq 1$	[Thm. 1]		
$\odot$, $\updownarrow$, $\leftrightarrow$ unimodal	$\leq \lg m \lg n / \lg \phi + O(\lg min)$	[Fib.×Fib.]	same as local	
	$\geq \lg m + \lg n$	[info. theo.]		

Table 1: Worst-case bounds on the number of probes required to maximize a function $f : \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\} \to \mathbb{R}$. In the bounds, $max = \max\{m, n\}$ and $min = \min\{m, n\}$.

The adversary gives the algorithm extra information, which can only help. Whenever the algorithm probes the value at a particular point (x, y), the adversary reveals not only that value, but also the slope of that value in that column, i.e., whether the mode in that column x is above or below that point (x, y). Furthermore, if the mode of column x is above the probe point (x, y), then the adversary reveals all values in the column x below the point (x, y); symmetrically, if the mode is below the probe point, the adversary reveals all values above the point in its column. If the algorithm discovers the mode of column x, the adversary reveals all values in the column x. Thus we maintain the invariant that every column that is not totally revealed has some revealed values in the topmost few rows, some revealed values in the bottommost few rows, and the algorithm knows that the mode of the column is somewhere in between.

If the unrevealed region ever becomes disconnected, the adversary reveals all values in all connected components except the largest connected component. Thus we maintain the invariant that the unrevealed region is connected. We also maintain the invariant that the algorithm cannot discover the unique local optimum until every value has been revealed. Together these two invariants make the goal of the algorithm to disconnect the unrevealed region; otherwise, the algorithm must make at least one probe per column, for a total of at least n probes.

The main task of the adversary is to decide whether a probe point is above or below the mode of that column, and then to choose the revealed values below or above the probe point. The adversary bases its decision on matching the "nearest" previous decision, according to a particular distance function. Define the *(biased) distance* between two points (x_1, y_1) and (x_2, y_2) to be

$$|x_1 - x_2| + |y_1 - y_2|/m^{1-\varepsilon/2}.$$

Also define the (biased) distance between a point (x, y) and the top horizontal wall to be $y/m^{1-\varepsilon/2}$, and similarly define the biased distance to the bottom wall to be $(m + 1 - y)/m^{1-\varepsilon/2}$.

Suppose that the algorithm probes the point (x, y). If point (x, y) is closer to a horizontal wall than every revealed point, then the adversary reveals all values in column x between (x, y) and the nearest wall, specifying that the mode is in the other direction. Otherwise, the adversary specifies (x, y) to be above or below the mode in its column x according to whether the revealed point (x^*, y^*) nearest to (x, y) is above or below the mode in its column x^*. Then the adversary reveals

all unrevealed values starting from (x, y) in the opposite direction to the mode in column x. (In the special case described below that the algorithm discovers the mode among these revealed values, the specification that the mode is above or below (x, y) is false; in this case the adversary reveals all values in column x.)

The adversary chooses the revealed values as follows. Suppose that the algorithm probes (x, y) and say that the adversary decides that probe point (x, y) is above the mode in its column x. If the to-be-revealed points keep the unrevealed region connected, then the adversary repeatedly reveals that the bottommost unrevealed value in column x is one more than the largest previously revealed value, until reaching point (x, y). In this way the revealed values increase in an integer sequence from the bottommost unrevealed value to (x, y). Equivalently, the adversary reveals every unrevealed point (x, y') below (x, y) in column x to have value $m - d$ more than the largest previously revealed value, where $d = y - y'$ is the Manhattan distance between the unrevealed point (x, y') and the probe point (x, y).

On the other hand, if the to-be-revealed points disconnect the unrevealed region, then we isolate a point $(x, \hat{y})$ in column x that is adjacent to the largest resulting connected component, and assign that point $(x, \hat{y})$ to be the mode of column x. (As mentioned above, this assignment contradicts the recent decision of the adversary that the mode is above (x, y); this situation is the only one in which such a contradiction arises.) Then the adversary reveals every unrevealed point (x, y') in column x to have value $m + n - d$ more than the largest previously revealed value, where $d = |y' - \hat{y}|$ is the Manhattan distance between unrevealed point (x, y') and the assigned mode $(x, \hat{y})$ of column x. Simultaneously, we reveal every point (x', y') in every connected component except the largest to have value $m + n - d$ more than the largest previously revealed value, where d is the Manhattan distance between point (x', y') and the assigned mode $(x, \hat{y})$ of column x. Thus point $(x, \hat{y})$ indeed becomes the mode of column x, with $d = 0$.

Lemma 2 *The only point to become a local maximum according to the adversary is the mode of the final column to become completely revealed.*

Proof: The key property is that the values revealed by the adversary are strictly larger as we proceed from one probe to the next, because we always add a positive number to the largest previously revealed value. Thus, within a column, every point except the mode has an adjacent point with larger revealed value, and therefore only the mode could be a local maximum. But in the final phase of a column, the mode is chosen so that it is adjacent to the largest connected component, and all values to be revealed in that component are strictly larger. Thus even the mode of the column cannot be a local maximum, unless the largest connected component is in fact empty, i.e., the last column has been completely revealed. □

Lemma 3 *Biased distance satisfies the triangle inequality.*

Lemma 4 *The algorithm must make* $\min\{n, (\varepsilon/2)\lg m\}$ *probes before the unrevealed region first disconnects into multiple connected components.*

Proof: To disconnect the unrevealed region horizontally, the algorithm must make at least n probes. We claim that disconnecting the unrevealed region vertically requires at least $(\varepsilon/2)\lg m$ probes. Consider the minimum (biased) distance D between a revealed point above the mode in its column (or the top wall) and a revealed point below the mode in its column (or the bottom wall). Initially D is $m/m^{1-\varepsilon/2} = m^{\varepsilon/2}$. If the unrevealed region disconnects, D must have become at most 1 because $\varepsilon \leq 1$.

Whenever the algorithm probes a point (x, y) that does not disconnect the unrevealed region, we claim that D can decrease by at most a factor of 2. Suppose that (x, y) is closest to a previously revealed point above the mode in its column (or the top wall). Then every newly revealed point (x, y') is also closest to a previously revealed point above the mode in its column. By the triangle inequality (Lemma 3), the sum of the distances from every such point (x, y') to the nearest point above the mode in its column (or the top wall) and to the nearest point below the mode in its column (or the bottom wall) is at least D. The former distance is smaller, so the latter distance is at least $D/2$. Therefore the new value of D after this probe is at least $D/2$.

In conclusion, for D to reduce from $m^{\varepsilon/2}$ to at most 1, the algorithm must make at least $\lg m^{\varepsilon/2} = (\varepsilon/2) \lg m$ probes. □

Lemma 5 *The nearest point or horizontal wall to a point (x, y) is in a column x' such that $|x-x'| \leq m^{\varepsilon/2}$.*

Proof: The distance between (x, y) and either horizontal wall (in the same column x) is always at most $m/m^{1-\varepsilon/2} = m^{\varepsilon/2}$. The distance between (x, y) and any point (x', y') is at least $|x - x'|$, so if $|x - x'| > m^{\varepsilon/2}$, the distance is strictly larger than the distance between (x, y) and either horizontal wall. □

Finally we conclude the proof of Theorem 1. Consider an algorithm that makes fewer than $\lg n \lg m$ probes. As mentioned above, the algorithm must disconnect the unrevealed region or else it is doomed to make at least n probes. Lemma 4 says that the algorithm must make at least $\min\{n, (\varepsilon/2) \lg m\}$ probes for the first disconnection. Consider the final probe that caused the disconnection. By the pigeon-hole principle, the $(\lg n \lg m)m^{\varepsilon/2}$ consecutive columns including and to the right of this final probe must have a gap of at least $m^{\varepsilon/2}$ consecutive empty columns, because there are at most $\lg n \lg m$ probes total. We remove columns starting from the final probe up to but not including this gap of $m^{\varepsilon/2}$ consecutive empty columns. Similarly, we remove at most $(\lg n \lg m)m^{\varepsilon/2}$ columns to the left of the final probe up to but not including a gap of $m^{\varepsilon/2}$ consecutive empty columns. Thus we obtain two subproblems (one left and one right) that by Lemma 5 act completely independently from each other and from the probes causing the disconnection, as far as probes made so far. We recursively consider the subproblem corresponding to the larger connected component that remains. This recursive subproblem is a rectangle with m rows and at least $\lfloor n/2 \rfloor - (\lg n \lg m)m^{\varepsilon/2}$ columns. The recursive subproblem may have already been probed, but we can consider such probes as happening after this subproblem, because Thus the recursion applies until $n'/2 < (\lg n \lg m)m^{\varepsilon/2}$.

Therefore we obtain the lower bound of $\min\{n', (\varepsilon/2) \lg m\}$ probes, where $n' \geq 2(\lg n \lg m)m^{\varepsilon/2}$, recursively $\lg(n/(2(\lg n \lg m)m^{\varepsilon/2}))$ times. In total we obtain a lower bound of

$$\begin{aligned} & (\varepsilon/2)(\lg m)(\lg n - 1 - \lg\lg n - \lg\lg m - (\varepsilon/2) \lg m) \\ \geq\ & (\varepsilon/2)(\lg m)((\varepsilon/2) \lg n - O(\lg\lg n)) \\ \geq\ & (\varepsilon^2/4) \lg m \lg n - O(\lg m \lg\lg n). \end{aligned}$$

References

[Kie53] J. Kiefer. Sequential minimax search for a maximum. *Proc. Amer. Math. Soc.*, 4:502–506, 1953.

[Mit03] Anton Mityagin. On the complexity of finding a local maximum of functions on discrete planar subsets. In *Proc. 20th Sympos. Theoret. Aspects Comput. Sci.*, volume 2607 of *Lecture Notes Comput. Sci.*, pages 203–211, February 2003.

Competitive Search Ratio of Graphs and Polygons

Rudolf Fleischer *,[a,1], Tom Kamphans [b,2], Rolf Klein [b,2], Elmar Langetepe [b,2] and Gerhard Trippen [a,1]

[a] *Hong Kong University of Science and Technology, Clear Water Bay, Kowloon Hong Kong*
[b] *University of Bonn, Institute of Computer Science I, D-53117 Bonn, Germany*

Abstract

We consider the problem of searching for a goal in an unknown environment, which may be a graph or a polygonal environment. The *search ratio* is the worst-case ratio before the goal is found while moving along some search path, over the shortest path from the start point to the goal, minimized over all search paths. We investigate the problem of finding good approximations to the optimal search path, with or without a priori knowledge of the environment. In the latter case, we are dealing with an online problem. We must compute a good search path while exploring the unknown environment. We present a unified framework that allows us to derive competitive search path algorithms from existing competitive exploration algorithms, if it is possible to modify them in a certain natural way. Our transformation increases the competitive ratio at most by a factor of eight. This expresses the relationship between competitive online exploration and online searching more precisely than before. We apply our framework to searching in trees, (planar) graphs, and in (rectilinear) polygonal environments with or without holes.

Key words: Online motion planning, competitive ratio, searching, exploration, search ratio

1. Introduction

Exploration and searching are fundamental tasks in online motion planning that have attracted a lot of interest during the last decade, see for example the survey by Berman [4]. *Exploration* means that an agent should detect *all* objects inside an unknown environment whereas in *searching* one particular *goal* object has to be detected.

The *competitive ratio* is an established performance measure for online algorithms, see for example [3,7]. In exploration and search problems, the *cost* of an algorithm is typically the length of the path traveled by the agent.

Intuitively, there is a close relationship between online searching and exploration. Since the goal could be hidden anywhere, every search strategy must eventually explore the entire scene. On the other hand, there is a fundamental difference between searching and exploration. In order to achieve a small search ratio, a search strategy should explore the environment in a breadth first search manner, first exploring all locations close to the starting point, then iteratively increasing the search radius. In contrast, an exploration strategy may decide to first follow some long path. For example, for unknown undirected graphs depth first search (DFS) is a 2-competitive online exploration algorithm, but there is no competitive online search algorithm, see [9]. This indicates that online exploration may be an easier problem than online searching.

Koutsoupias et al. [10] first studied how to compute search path approximations offline. A *search path* π explores the entire enviroment. If the goal

* Corresponding author
Email addresses: rudolf@cs.ust.hk (Rudolf Fleischer), kamphans@informatik.uni-bonn.de (Tom Kamphans), rolf.klein@uni-bonn.de (Rolf Klein), langetep@informatik.uni-bonn.de (Elmar Langetepe), trippen@cs.ust.hk (Gerhard Trippen).

[1] The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. HKUST6010/01E) and by a grant from the Germany/Hong Kong Joint Research Scheme sponsored by the Research Grants Council of Hong Kong and the German Academic Exchange Service (Project No. G-HK024/02).
[2] This research was supported by the same DAAD/RGC 2003/2004 grant.

becomes visible for the first time while moving along π, we can move towards it. The quality of a search path is determined by a worst-case goal point v that maximizes the ratio between the length of the path to v via π and the shortest path to v, among all goal points v. Koutsoupias et al. called this worst-case ratio the *search ratio* of π. An *optimal search path* has the minimal search ratio among all search paths. They only studied graphs with unit length edges where the goal can only be located at one of the vertices of the graph, and they only studied the offline case, i. e., with full a priori knowledge of the graph. They showed that computing the optimal search ratio offline is an NP-complete problem, and they gave a polynomial time 8-approximation algorithm based on the doubling heuristic, which is also the main ingredient of our approach. The doubling heuristic is a standard technique in online motion planning, see for example [3,2].

Our goal is to approximate the optimal search path in an *unknown* environment by a constant-competitive *online* search algorithm. We propose a general framework how to derive such a competitive online algorithm when there exists a competitive online exploration algorithm for the environment that can be adapted in a certain natural way. Our framework can also be applied to offline exploration approximation algorithms, in which case it yields offline approximation algorithms for the optimal search ratio. The competitive ratio of the search algorithm is only at most four or eight times the approximation factor of the exploration algorithm, depending on whether goals can only be hidden at some finite set of points, like the vertices of a graph, for example, or whether the goal can be hidden anywhere in the environment.

In this extended abstract, all proofs are omitted.

2. Definitions

We want to find a good search path in some given *environment* E. In a graph environmen, edge lengths do not necessarily represent Euclidean distances in some embedding of the graph, even if the graph is planar. In particular, we do not assume that the triangle inequality holds.

The *goal set* $G \subseteq E$ is the set of locations in the environment where the goal might be hidden. For example, if E is a graph $G = (V, E)$, then the goal may either be located anywhere on some edge (*geometric search*), i. e. $G = V \cup E$, or it may only be located at some vertex (*vertex search*), i. e., $G = V$. To *explore* E means to move around in E and eventually see all potential goal positions G. To *search* E means to follow some exploration path in E, the *search path*, until the goal is detected. A path π is a search path if every point in G can be seen from at least one point on π. We assume that search paths always return to their start point. We make the usual assumption that goals must be at least in distance 1 from the start point s. If goals could be arbitrarily close to s, no algorithm could be competitive.

For $d \geq 1$, let $E(d)$ denote the part of E in distance at most d from s. A *depth-d restricted exploration* explores all potential goal positions in $G(d)$. The search path may move outside $E(d)$ as long as it stays within E. Depth-d restricted search is defined similarly.

Agents can either be *blind*, i. e., they can only sense their very close neighborhood, or they can have *vision*, i. e., they can see objects far away if the line of view is not blocked by an obstacle. In a graph environment, an agent with vision can usually see the endpoints of all adjacent edges at a vertex (even if the edges have a curved embedding in the plane). A blind agent standing at a vertex can only sense the outgoing edges of the vertex; it cannot see the length or the endpoint of an edge, and it cannot see the incoming edges [5].

In polygon searching we usually assume the agents to have vision, whereas in graph searching we usually assume the agents to be blind. Note that a blind agent must eventually visit all points in the goal set, wheras an agent with vision can afford to skip some potential goal positions as long as it can see them from somewhere else (maybe far away). For example, when searching a polygon it is sufficient to visit all visibility cuts, i. e., cuts emanating from the edges of reflex vertices.

We assume that agents have perfect memory. They always know a map of the already explored part of E, and they can always recognize when they visit some point for the second time, i. e., they have perfect localization.

Let π be a path in the environment E. For a point $p \in \pi$ let $\pi(p)$ denote the part of π between s and p and $\mathrm{sp}(p)$ the shortest path from s to p in E. We denote the length of a path segment $\pi(p)$ by $|\pi(p)|$. We write $|A|$ for the length of the tour computed by an algorithm A. For a point $p \in E$ let $\mathrm{rise}_\pi(p)$

denote the point $q \in \pi$ from which p is seen for the first time when moving along π starting at s.

The qualitiy of a search path is measured by its *search ratio* $\mathrm{sr}(\pi)$, defined as $\mathrm{sr}(\pi) := \max_{p \in G} \frac{|\pi(q)|+|qp|}{|\,\mathrm{sp}(p)|}$, where $q = \mathrm{rise}_\pi(p)$. Note that $q = p$ for blind agents. An *optimal search path*, π_{opt}, is a search path with a minimum search ratio $\mathrm{sr}_{\mathrm{opt}}$, i. e., $\mathrm{sr}_{\mathrm{opt}} = \mathrm{sr}(\pi_{\mathrm{opt}})$.

Since the optimal search path seems hard to compute [10], we are interested in finding good approximations of the optimal search path, in offline and online scenarios. We say a search path π is *C-competitive* (with respect to the optimal search path π_{opt}) if $\mathrm{sr}(\pi) \le C \cdot \mathrm{sr}(\pi_{\mathrm{opt}})$.

3. A General Approximation Framework

In this section we will show how to transform an exploration algorithm, offline or online, for a certain environment into a search algorithm, without losing too much on the approximation factor.

Let E be the given environment and π_{opt} an optimal search path. We assume that, for any point p, we can reach s from p on a path of length at most $\mathrm{sp}(p)$. Note that this is not the case for directed graphs, for example, but it is true for undirected graphs and polygonal environments.

For $d \ge 1$, let $Expl(d)$ be a family of depth-d restricted exploration algorithms for E, either online or offline. Let OPT and $\mathrm{OPT}(d)$ denote the corresponding optimal offline depth-d restricted exploration algorithms.

Definition 1 *The family $Expl(d)$ is* DREP (depth restricted exploration property) *if there are constants $\beta > 0$ and $C_\beta \ge 1$ such that, for any $d \ge 1$, $Expl(d)$ is C_β-competitive against the optimal algorithm $OPT(\beta d)$, i. e., $|Expl(d)| \le C_\beta \cdot |OPT(\beta d)|$.* □

In the normal competitive framework, $\beta = 1$. Usually, we cannot just take an exploration algorithm $Expl$ for E and restrict it to points in distance at most d from s. This way, we might miss useful shortcuts outside of $E(d)$. Even worse, it may not be possible to determine in an online setting which parts of the environment belong to $E(d)$, making it difficult to explore the right part of E.

To obtain a search algorithm for E we use the well-known *doubling strategy*. For $i = 1, 2, 3, \cdots$, we successively run the exploration algorithm $Expl(2^i)$, each time starting at s.

Theorem 2 *The doubling strategy based on a DREP exploration strategy is a $4\beta C_\beta$-competitive (plus an additive constant) search algorithm for blind agents, and a $8\beta C_\beta$-competitive search algorithm for agents with vision.* □

In the next two sections we will apply our framework to various types of environments and agents. The difficult part is always to find good DREP exploration algorithms.

4. Searching Graphs and Polygons

When searching graphs, we assume agents are blind. In the vertex search problem, we assume w.l.o.g. that graphs do not have parallel edges. Otherwise, there can be no constant-competitive vertex search algorithm.

Theorem 3 *For blind agents, there is no constant-competitive online vertex search algorithm for non-planar graphs with unit length edges, directed planar graphs with unit length edges, and undirected planar graphs with arbitrary edge lengths. Further, there is no constant-competitive online geometric search algorithm for directed graphs with unit length edges.* □

Note that non-competitiveness for planar graphs implies non-competitiveness for general graphs, non-competitiveness for unit length edges implies non-competitiveness for arbitrary length edges, and non-competitiveness for undirected graphs implies non-competitiveness for directed graphs.

On trees, DFS is a 1-competitive online exploration algorithm for vertex and geometric search that is DREP; it is still 1-competitive when restricted to search depth d, for any $d \ge 1$. Thus, the doubling strategy gives a polynomial time 4-competitive search algorithm for trees, online and offline. On the other hand, it is an open problem whether the computation of an optimal vertex or geometric search path in trees with unit length edges is NP-complete [10].

We will now give competitive search algorithms for planar graphs with unit length edges and for general graphs with arbitrary length edges. Both algorithms are based on an online algorithm for online tethered graph exploration. In the *tethered*

exploration problem the agent is fixed to the start point by a restricted length rope. An optimal solution to this problem was given by Duncan et al. [6]. As they pointed out, the algorithm can also be used for depth restricted exploration. We note that it can also be adapted to work on graphs with arbitrary length edges.

Lemma 4 *In planar graphs with unit length edges, Expl(d) is a DREP online vertex exploration algorithm with* $\beta = 1 + \alpha$ *and* $C_\beta = 10 + \frac{16}{\alpha}$.

Theorem 5 *The doubling strategy based on Expl(d) is a* $(104 + 40\alpha + \frac{64}{\alpha})$*-competitive online vertex search algorithm for blind agents in planar graphs with unit length edges.* □

For $\alpha = \sqrt{\frac{16}{10}}$, the competitive factor is minimal.

Lemma 6 *In general graphs with arbitrary length edges, Expl(d) is a DREP online geometric exploration algorithm with* $\beta = 1+\alpha$ *and* $C_\beta = 4+\frac{8}{\alpha}$. □

Theorem 7 *The doubling strategy based on Expl(d) is a* $(48 + 16\alpha + \frac{32}{\alpha})$*-competitive online geometric search algorithm for blind agents in general graphs with arbitrary length edges.* □

When searching polygons, we assume that agents have vision. The only known algorithm, *PE*, for the online exploration of a simple polygon by Hoffmann et al. [8] achieves a competitive ratio of 26.5. This algorithm can be adapted for depth restricted exploration.

Lemma 8 *In a simple polygon, PE(d) is a DREP online exploration algorithm with* $\beta = 1$ *and* $C_\beta = 26.5$. □

Theorem 9 *The doubling strategy based on PE(d) is a* 212*-competitive online search algorithm for an agent with vision in a simple polygon. There is also a polynomial time* 8*-competitive offline search algorithm.* □

The problem of efficiently computing an optimal search path in a polygon is still open.

Theorem 10 *For an agent with vision in a simple rectilinear polygon there is a* $8\sqrt{2}$*-competitive online search algorithm. There is also a polynomial time* 8*-competitive offline search algorithm.* □

Based on a construction by Albers et al. [1] we can show the following theorem.

Theorem 11 *For an agent with vision in a polygon with rectangular holes there is no constant-competitive online search algorithm.* □

The offline exploration problem is NP-hard, and there is an exponential time 8-approximation algorithm. Thus, we get an approximation factor of 8 for the optimal search ratio. The results of Koutsoupias et al. [10] imply that the offline problem of computing an optimal search path in a known polygon with (rectangular) holes is NP-complete.

References

[1] S. Albers, K. Kursawe, and S. Schuierer. Exploring unknown environments with obstacles. In *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*, pages 842–843, 1999.

[2] S. Alpern and S. Gal. *The Theory of Search Games and Rendezvous*. Kluwer Academic Publications, 2002.

[3] R. Baeza-Yates, J. Culberson, and G. Rawlins. Searching in the plane. *Inform. Comput.*, 106:234–252, 1993.

[4] P. Berman. On-line searching and navigation. In A. Fiat and G. Woeginger, editors, *Competitive Analysis of Algorithms*. Springer-Verlag, 1998.

[5] X. Deng and C. H. Papadimitriou. Exploring an unknown graph. *Journal of Graph Theory*, 32:265–297, 1999.

[6] C. A. Duncan, S. G. Kobourov, and V. S. A. Kumar. Optimal constrained graph exploration. In *Proc. 12th ACM-SIAM Symp. Discr. Algo.*, pages 307–314, 2001.

[7] A. Fiat and G. Woeginger, editors. *On-line Algorithms: The State of the Art*, volume 1442 of *Lecture Notes Comput. Sci.* Springer-Verlag, 1998.

[8] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel. The polygon exploration problem. *SIAM J. Comput.*, 31:577–600, 2001.

[9] C. Icking, T. Kamphans, R. Klein, and E. Langetepe. On the competitive complexity of navigation tasks. In H. B. Hendrik I. Christensen, Gregroy D. Hager and R. Klein, editors, *Sensor Based Intelligent Robots*, volume 2238 of *LNCS*, pages 245–258, Berlin, 2002. Springer.

[10] E. Koutsoupias, C. H. Papadimitriou, and M. Yannakakis. Searching a fixed graph. In *Proc. 23th Internat. Colloq. Automata Lang. Program.*, volume 1099 of *LNCS*, pages 280–289. Springer, 1996.

Finding Planar Regions in a Terrain

Stefan Funke [1], Theocharis Malamatos, Rahul Ray
Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany

Abstract

We consider the problem of computing large connected regions in a triangulated terrain of size n for which the normals of the triangles deviate by at most some small fixed angle. In previous work an exact near-quadratic algorithm was presented, but only a heuristic implementation with no guarantee was practicable. We present a new approximation algorithm for the problem which runs in $O(n/\epsilon^2)$ time and—apart from giving a guarantee on the quality of the produced solution—has been implemented and shows good performance on real data sets representing fracture surfaces with around half a million triangles.

1. Introduction

A terrain is a surface in $\mathbb{R}^3$ defined by a function $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. If f is piecewise linear and the surface consists of a collection of triangles, the terrain is called a *triangulated irregular network (TIN)*. Given a TIN $\mathcal{T}$, the goal is to find large, nearly planar regions in $\mathcal{T}$. We define 'near planarity' as follows: for a real parameter $\delta > 0$, we say that a subset of triangles T in $\mathcal{T}$ is *δ-planar* if (i) the adjacency graph of the triangles in T is connected, and (ii) there is a vector $\overrightarrow{r}$ such that for each $t \in T$, $\angle(n_t, r) \leq \delta$, where n_t denotes the normal of triangle t. We call $\overrightarrow{r}$ the *reference normal*. (Throughout the paper $\angle(v, u)$ denotes the angle between two vectors $\overrightarrow{v}$ and $\overrightarrow{u}$.)

Researchers in the material sciences examine surface topographies of materials as they provide useful information about the generation process and the internal structure of the material. Surfaces generated by fracture, wear, corrosion and machining are of interest. Among many other criteria, they want to examine feature-related parameters like facets in brittle fracture surfaces. This application motivated our work.

Email addresses: funke@mpi-sb.mpg.de (Stefan Funke), tmalamat@mpi-sb.mpg.de (Theocharis Malamatos), rahul@mpi-sb.mpg.de (Rahul Ray).

[1] Part of this research was conducted while the author was visiting the University of Illinois at Urbana-Champaign, USA.

Related work. Assume that each triangle in $\mathcal{T}$ is assigned a weight and that any set of triangles has weight equal to the sum of the weights of its triangles. Smid, Ray, Wendt and Lange [2] considered the problem of finding a δ-planar subset of $\mathcal{T}$ of maximum weight. They presented an $O(n^2 \log n (\log\log n)^3)$ algorithm, where n is the number of triangles in $\mathcal{T}$. Since this is impractical they proposed a heuristic which computes nearly planar regions quite quickly but with no guarantee. Also they suggested that finding a δ-planar set with weight at least a constant fraction of the optimum may be equally difficult as solving the problem exactly, see [2].

Our results. In this paper we adopt the following notion of approximation. Given a real parameter $\epsilon > 0$, we say that a subset of triangles T of $\mathcal{T}$ is *ϵ-approximate* δ-planar if it is $\delta(1+\epsilon)$-planar and has weight at least as large as an optimal δ-planar set. Under this notion, we present a new approximation algorithm that runs in $O(n/\epsilon^2)$ time which is independent of δ. For n sufficiently large, the algorithm uses optimal $O(n)$ space. Recently Har-Peled and Mazumdar [1] have used a similar notion of approximation for the problem of computing the smallest k-enclosing disk.

We have implemented and empirically evaluated the algorithm on real test data from the application domain consisting of fracture surface terrains with more than 500,000 triangles. It provides very good quality results in reasonable time.

2. Preliminaries

Let $\mathcal{T}$ be a TIN. We associate with $\mathcal{T}$ an undirected weighted graph $G_{\mathcal{T}}(V, E)$ as follows. Each triangle t in $\mathcal{T}$ has an associated weight $w(t)$ and corresponds to a vertex v_t in V. An edge connects two vertices of V if and only if the corresponding triangles in $\mathcal{T}$ are adjacent. Note that $G_{\mathcal{T}}$ is the dual graph of $\mathcal{T}$, is planar and has degree three. Each vertex $v_t \in V$ is assigned the weight of its associated triangle $w(t)$ which can be, for example, equal to the area of the triangle t or one (when we want to maximize the area of the detected region or the number of triangles, respectively). The weight $w(V')$ of any subset V' of V is defined as the sum of the weights of the vertices in V'.

For a point $u \in \mathbb{R}^3$ we denote by $\overrightarrow{u}$ the vector $\overrightarrow{Ou}$, where O is the origin. Let $\mathbb{S}^2$ denote the unit sphere, i.e., the boundary of the three-dimensional ball of radius one centered at the origin. For each triangle $t \in \mathcal{T}$, we can associate a point $p_t \in \mathbb{S}^2$ which represents the normalized normal of triangle t. Specifically, $\overrightarrow{p_t} = \overrightarrow{n_t}/|\overrightarrow{n_t}|$. Our goal is to approximate the space $\mathbb{S}^2$ of all possible normals by a finite set of points $\mathbb{V} \subset \mathbb{S}^2$ such that for any $p \in \mathbb{S}^2$, there is a point $v \in \mathbb{V}$ nearby.

Definition 1 *A set of point $\mathbb{V} \subset \mathbb{S}^2$ is called a $\delta\epsilon$-*discretization *of $\mathbb{S}^2$ if $\forall p \in \mathbb{S}^2 : \exists v \in \mathbb{V}$ with $\angle(v, p) \leq \delta \cdot \epsilon$.*

Lemma 2 *There exists a $\delta\epsilon$-discretization of $\mathbb{S}^2$ of size $O(1/(\delta\epsilon)^2)$ which can be computed in the same time.*

The following construction yields a $\delta\epsilon$-discretization of $\mathbb{S}^2$ as needed in Lemma 2. (We omit its proof in this abstract.) Consider a cube L with sidelength two centered at the origin. Note that $\mathbb{S}^2 \subset L$. Place a 2-dimensional grid of size $k \times k$ with $k = \lceil\sqrt{2}/(\delta\epsilon)\rceil$ over each of the six faces of L. This generates k^2 equally sized square grid cells on each face of L, where each cell has sidelength at most $(\delta\epsilon\sqrt{2})$, and $6k^2 + 2$ grid points overall. See Figure 1. Let A be the set consisting of these grid points. Our $\delta\epsilon$-discretization $\mathbb{V}$ of $\mathbb{S}^2$ is defined as

$$\mathbb{V} = \left\{ \frac{\overrightarrow{z}}{|\overrightarrow{z}|} : z \in A \right\},$$

that is, for each gridpoint z we shoot a ray from the origin through z and we include the point where the ray leaves $\mathbb{S}^2$ into set $\mathbb{V}$.

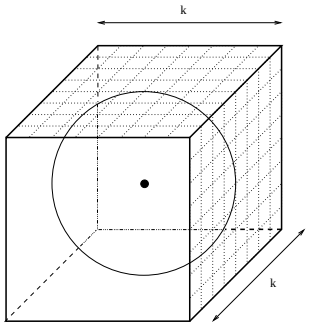


Fig. 1. Cube with sidelength two containing $\mathbb{S}^2$ and with a $k \times k$ grid on each of its faces.

3. Finding Large Planar Regions

3.1. *The Basic Algorithm*

We first describe a simple algorithm for the problem that computes an ϵ-approximate solution in $O(n/(\delta\epsilon)^2)$ time. The algorithm proceeds as follows:

1. Compute a $\delta\epsilon$-discretization $\mathbb{V}$ of $\mathbb{S}^2$.
2. For each $p \in \mathbb{V}$,
   (a) Compute the set V_p of vertices v_t with $\angle(p, n_t) \leq (1 + \epsilon) \cdot \delta$.
   (b) Consider the subgraph of $G_{\mathcal{T}}$ induced by set V_p and determine its heaviest connected component C_p.
3. Report the set of triangles T corresponding to the heaviest component C_p found in Step 2 and the associated reference normal $\overrightarrow{p}$.

We now discuss the running time and correctness of the algorithm. Computing the $\delta\epsilon$-discretization takes $O(1/(\delta\epsilon)^2)$ by Lemma 2. For each element $p \in \mathbb{V}$ we determine the subgraph induced by V_p and compute its connected components, which can be done in $O(n)$ time. So the total running time of Step 2 is $O(n/(\delta\epsilon)^2)$ which also dominates the overall running time.

For correctness, observe first that clearly the computed set T is $\delta(1 + \epsilon)$-planar. It remains to show that the weight of T is at least that of an optimal δ-planar set T^*. For set T^* there exists a vector $\overrightarrow{r^*}$ such that for all triangles $t \in T^*$, $\angle(r^*, n_t) \leq \delta$. Let p be a point in $\mathbb{V}$ for which $\angle(p, r^*) = \min_{u \in \mathbb{V}} \angle(u, r_{\text{opt}})$. By the definition of $\mathbb{V}$, the angle $\angle(p, r^*)$ must be at most $\delta\epsilon$. Then, for any triangle $t \in T^*$ the angle between $\overrightarrow{n_t}$ and $\overrightarrow{p}$ is at most $\delta + (\delta\epsilon) = \delta(1 + \epsilon)$. Therefore for all $t \in T^*$,

$v_t \in V_p$ and hence, the algorithm will find a connected component with at least the same weight.

Lemma 3 *Given a triangulated irregular network $\mathcal{T}$, and two real parameters $\delta > 0$ and $\epsilon > 0$ we can compute in $O(n/(\delta\epsilon)^2)$ time an ϵ-approximate δ-planar set of triangles in $\mathcal{T}$.*

The running time of the basic algorithm is optimal in terms of n. But one may ask whether the dependence on ϵ or δ can be improved. In the following we will refine the algorithm to achieve $O(n/\epsilon^2)$ running time, which is independent of δ.

3.2. *The Refined Algorithm*

The improvement in the running time of the algorithm comes from two factors. First we determine a set of reference normals $\mathbb{V}'$ of size at most $O(n/\epsilon^2)$ that contains all relevant reference normals, avoiding the inspection of $\Omega(1/(\delta\epsilon)^2)$ potential reference normals. Second by a bucketing scheme, we reduce significantly the number of times each triangle is considered. The refined algorithm proceeds as follows:

1. For each triangle $t \in \mathcal{T}$ with normal n_t, let p_t be a point in $\mathbb{V}$ for which $\angle(p_t, n_t) = \min_{u\in\mathbb{V}} \angle(u, n_t)$; store t in the bucket associated with p_t.
2. Determine a set $\mathbb{V}' \subset \mathbb{V}$ of potential reference normals as $\mathbb{V}' = \{p \in \mathbb{V} : \exists p_t$ with non-empty bucket and $\angle(p, p_t) \leq (1 + 2\epsilon) \cdot \delta\}$.
3. For each $v \in \mathbb{V}'$,
   (a) Collect the set of triangles N_v which are contained in buckets of reference normals $r' \in \mathbb{V}'$ with $\angle(r', v) \leq (1 + 2\epsilon) \cdot \delta$.
   (b) Prune N_v keeping only those triangles t with $\angle(n_t, v) \leq (1 + \epsilon) \cdot \delta$. Let N'_v be the pruned set.
   (c) Consider the subgraph of $G_{\mathcal{T}}$ induced by the vertices corresponding to triangles in N'_v and determine its heaviest connected component C_v.
4. Output the heaviest connected component C_v found in Step 3.

Before analysing the algorithm, we state a small lemma which informally says that in the $\delta\epsilon$-discretization, for any point p there is a small number of other points nearby.

Lemma 4 *Let p be a point in the $\delta\epsilon$-discretization $\mathbb{V}$. Then the number of points $p' \in \mathbb{V}$ with $\angle(p, p') < (1 + 2\epsilon) \cdot \delta$ is at most $O(1/\epsilon^2)$.*

The proof of this lemma follows easily from our construction of $\mathbb{V}$ and it is omitted. We note also that given a triangle normal n_t we can compute the point p_t with $\angle(n_t, p_t) = \min_{u\in\mathbb{V}} \angle(n_t, u)$ in constant time by first determining which face of the cube L is hit by the ray $\overrightarrow{n_t}$ and then locating the position of the intersection point within the grid on that face. We now state the main result of this section:

Theorem 5 *Given a triangulated irregular network $\mathcal{T}$, and two real parameters $\delta > 0$ and $\epsilon > 0$ we can compute in $O(n/\epsilon^2)$ time an ϵ-approximate δ-planar set of triangles in $\mathcal{T}$.*

PROOF. We claim that the output of the refined algorithm give us such a set. To show correctness, it suffices to prove that the algorithm computes the same solution as the basic algorithm. We leave this proof to the reader. We focus now on the running time. Step 1 of the algorithm takes $O(n)$ since for each t we can in constant time determine p_t and access the associated bucket using a hashing scheme. Step 2, where we form the set $\mathbb{V}'$, takes $O(n/\epsilon^2)$ since there are at most n non-empty buckets. For each of the non-empty buckets, we explore $O(1/\epsilon^2)$ points in the neighborhood by Lemma 4. Finally, for the overall running time of Step 3, observe that again by Lemma 4, each triangle can be collected by at most $O(1/\epsilon^2)$ reference normals and that the running time of one iteration of Step 3 is $O(|N_v|)$. Therefore Step 3 takes $O(n/\epsilon^2)$ time in total. □

4. Experimental Results

We have implemented the refined algorithm of Section 3.2 in C++ using the LEDA library [3]. Experiments were carried out on a 1.8 GHz Pentium 4 machine with 256 MB of RAM. We used several data sets representing fracture surfaces from the application domain in material sciences. Input data were given as 512×512 raster images with the intensity of each pixel corresponding to its height value. To obtain the TIN, we triangulated the point set by creating triangles $(i, j), (i+1, j), (i+1, j+1)$ and $(i, j), (i, j+1), (i+1, j+1)$. See Figure 2.

To speed-up our program (while preserving the same guarantee), we have come up with the following three heuristics:

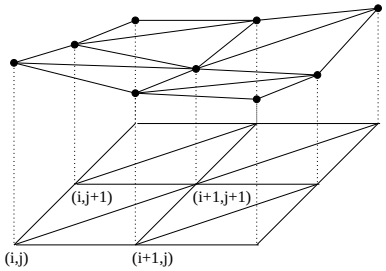


Fig. 2. Triangulation scheme for the array of height values.

(i) With each potential reference normal we associate the weight of all triangles in buckets at distance at most $(1 + 2\epsilon) \cdot \delta$ and examine the reference normals in decreasing order of weight. If the current best solution exceeds the associated weight of the next reference normal, we can stop examining further.

(ii) If the normal of a triangle t forms an angle larger than $2(1 + \epsilon)\delta$ with the normal of each of the three adjacent triangles, we can prune t out (at Step 1). It can only form a singleton solution.

(iii) In Step 3(b) where we compute the set of relevant triangles N'_v we only check triangles in buckets at angle distance more than δ, thus saving some expensive floating-point operations.

In our experiments, the three heuristics combined reduce the running time by nearly a factor of two. For our test instances consisting of roughly $500,000$ triangles and a choice of $\delta = 0.2$ (about 11.5 degrees) and $\epsilon = 0.2$, the running time varied between 70 and 110 sec. The best solution was in all cases found within the first 20 seconds due to the prioritization scheme. Thus most of the running time was spent on checking that no better solution exists. (A heuristic could just report the solution computed, for example, after 30 seconds.)

We also ran experiments to determine the variations of the running time as a function of n and ϵ. The dependence graph on ϵ for a fixed data set is shown in Figure 3. The upper curve denotes the total running time and the lower curve denotes the time when the reported solution was detected. In Figure 4 we examine the dependence on n (number of triangles). Note that again the reported solution was found after only few seconds.

For the heuristic implementation in [2], running times reported are in the range of 30–40 seconds but as it is mentioned the solution computed may be far from optimal.

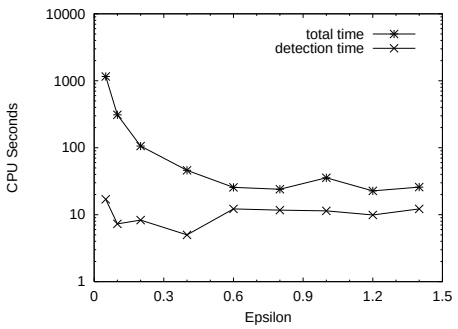


Fig. 3. CPU time versus ϵ for $n = 522$K, $\delta = 0.2$.

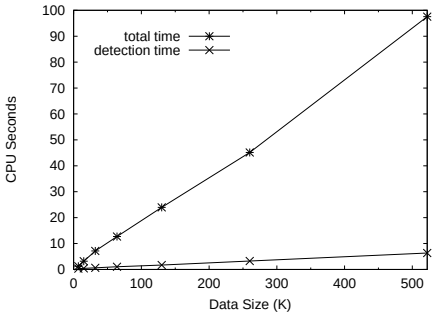


Fig. 4. CPU time versus n for $\delta = 0.2$, $\epsilon = 0.2$.

References

[1] S. Har-Peled and S. Mazumdar. Fast algorithms for computing the smallest k-enclosing disc. In *Proc. 11th Annu. European Sympos. Algorithms*, Lecture Notes Comput. Sci. Springer-Verlag, 2003.

[2] K. Lange, R. Ray, M. Smid, and U. Wendt. Computing large planar regions in terrains. In *Proceedings of the 8th International Workshop on Combinatorial Image Analysis (IWCIA)*, Electronics Notes in Computer Science, Volume 46, 2002.

[3] K. Mehlhorn and S. Näher. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge, U.K., 1999.

[4] U. Wendt, K. Lange, M. Smid, R. Ray, and K. Tönnies. Surface topography quantification by integral and feature-related parameters. *Materialwissenschaft und Werkstofftechnik*, 33(10):621–627, October 2002.

Maximizing the Area of Overlap of two Unions of Disks under Rigid Motion

Mark de Berg [a], Sergio Cabello [b,1], Panos Giannopoulos [*,b,1,2], Christian Knauer [c], René van Oostrum [b,1] and Remco C. Veltkamp [b]

[a] *Faculty of Mathematics and Computing Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, the Netherlands.*
[b] *Institute of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, the Netherlands.*
[c] *Institut für Informatik, Freie Universität Berlin, Takusstraße 9, D-14195 Berlin, Germany.*

Abstract

Let A and B be two sets of n resp. m ($m \geq n$) disjoint unit disks in the plane. We consider the problem of finding a rigid motion of A that maximizes the total area of its overlap with B. The function describing the area of overlap is quite complex, even for combinatorially equivalent translations, and hence, we turn our attention to approximation algorithms. We give deterministic $(1-\epsilon)$-approximation algorithms for the maximum area of overlap under translation and rigid motion that run in $O((nm/\epsilon^2)\log(m/\epsilon))$ and $O((n^2m^2/\epsilon^3)\log m))$ time respectively. For the later, if Δ is the diameter of set A, we get an $(1-\epsilon)$-approximation in $O(\frac{m^2n^{4/3}\Delta^{1/3}\log n\log m}{\epsilon^3})$ time which is improvement when $\Delta = o(n^2)$. Under the condition that the maximum is at least a constant fraction of the area of B, we give a probabilistic $(1-\epsilon)$-approximation algorithm for rigid motions that runs in $O((cn^2/\epsilon^5)\log^2(n/\epsilon))$ time and succeeds with probability at least $1-e^{-c}-n^{-6}$.

Key words: Geometric Optimization, Approximation Algorithms, Shape Matching, Area of Overlap, Unions of Disks, Rigid Motions

1. Introduction

Shape matching is a fundamental problem in computer vision: given two shapes A and B and a distance measure, one wants to determine a transformation of A - a translation, for instance, or a rigid motion - that minimizes its distance to B. Typical problems include: matching point sets with respect to the Hausdorff distance and matching polygons with respect to the Hausdorff or Fréchet distance between their boundaries; see Alt et al. [1] for a survey. The *area of overlap* (or the *area the symmetric difference*) of two polygons is less sensitive to noise and therefore more appropriate for certain applications. The function of the area of overlap of two translated simple polygons was first studied by Mount et al. [5]. They showed that this function is a continuous, $O((nm)^2)$ piecewise polynomial surface of at most degree two with a representation that can be computed in $O((nm)^2)$ time for two polygons with n and m vertices; efficient algorithms for determining the optimal translation for the case of *convex* polygons and constant-factor approximation of the minimum area of symmetric difference of two convex shapes under translation can be also found in the survey by Alt et al. [1].

Recently, de Berg et al. [3] examined the problem of matching unions of convex homothets or unions of fat objects and gave a nearly quadratic deterministic $(1-\epsilon)$-approximation algorithm for the maximum area of overlap under translations;

* Panos Giannopoulos
Email addresses: m.t.d.berg@TUE.nl (Mark de Berg), sergio@cs.uu.nl (Sergio Cabello), panos@cs.uu.nl (Panos Giannopoulos), Christian.Knauer@inf.fu-berlin.de (Christian Knauer), rene@cs.uu.nl (René van Oostrum), Remco.Veltkamp@cs.uu.nl (Remco C. Veltkamp).
[1] This research was partialy supported by the STW, project UIF 5055 and the Cornelis Lely Stichting
[2] The author was partially supported by a Marie Curie Fellowship of the EC programme "Combinatorics, Geometry, and Computation, FU Berlin, HPMT-CT-2001-00282.

we mention part of these results in this abstract as well. Their motivation comes from applications in matching shapes that are 'expressed' as unions of convex objects and weighted point set matching. Cheong et al. [2] gave a nearly linear time probabilistic approximation algorithm for the same problem.

Let $A = \{A_1, \ldots, A_n\}$ and $B = \{B_1, \ldots, B_m\}$, $(n \leq m)$ be two sets of disjoint unit disks in the plane. We consider B to be fixed, while A can be translated and rotated relative to B. Let $\mathcal{I}$ be the infinite set of all possible isometries in the plane; we call $\mathcal{I}$ the *configuration space*. We denote by R_θ a rotation about the origin by some angle $\theta \in [0, 2\pi)$ and by $T_{\vec{t}}$ a translation by some $\vec{t} \in \mathbb{R}^2$. Rotated only versions of A are denoted by $A(\theta) = \{A_i(\theta), \ldots, A_n(\theta)\}$; translated only versions of A are denoted by $A(\vec{t}) = \{A_i(\vec{t}), \ldots, A_n(\vec{t})\}$. Any isometry $I \in \mathcal{I}$ can be uniquely defined as $I = I_{\vec{t},\theta} = R_\theta \circ T_{\vec{t}}$ or $I = I_{\vec{t}',\theta'} = T_{\vec{t}'} \circ R_{\theta'}$, for some $\theta, \theta' \in [0, 2\pi)$ and $\vec{t}, \vec{t}' \in \mathbb{R}^2$; transformed versions of A are denoted by $A(\vec{t}, \theta) = \{A_i(\vec{t}, \theta), \ldots, A_n(\vec{t}, \theta)\}$ for some $I_{\vec{t},\theta} \in \mathcal{I}$. Let $r_i(\vec{t})$ be the distance of $A_i(\vec{t}, \theta)$'s center to the origin.

Let $V(\mathcal{C})$ be the area of compact set $\mathcal{C} \in \mathbb{R}^2$ and $\mathcal{V}_{ij}(\vec{t}, \theta) := V(A_i(\vec{t}, \theta) \cap B_j)$. The *area of overlap* of $A(\vec{t}, \theta)$ and B, as $\vec{t}, \theta$ vary, is a function $\mathcal{V} : \mathcal{I} \to \mathbb{R}$ with $\mathcal{V}(\vec{t}, \theta) = V(A(\vec{t}, \theta) \cap B) = \sum_{A_i \in A, B_j \in B} V(A_i(\vec{t}, \theta) \cap B_j)$. We investigate the following problem: *For two sets A, B, defined as above, compute the optimal isometry $I_{\vec{t}_{opt},\theta_{opt}}$ that maximizes $\mathcal{V}(\vec{t}, \theta)$.*

One can prove that, the maximum number of combinatorially distinct translations of A with respect to B can be as high as $\Theta(n^2 m)$ and the function describing the area of overlap is quite complex, even for combinatorially equivalent translations. If rotations are allowed as well the complexity of the configuration space can be as high as $\Theta(m^2 n^3)$.

Our major contributions are the following: (i) a lower bound on the area of overlap which is vital for almost all our algorithms, (ii) a nearly quadratic deterministic algorithm for translations, (iii) deterministic and nearly quadratic probabilistic algorithms for rigid motions. The algorithms are based on a simple two step framework in which an approximation of the best translation if followed by an approximation of the best rotation, whenever the latter applies. This way we first achieve an absolute error on $\mathcal{V}(\vec{t}_{opt}, \theta_{opt})$ which we then turn into a relative one using the lower bound theorem. In the deterministic one we employ a clever sampling of configuration space directed by some special properties of the function $\mathcal{V}_{ij}$. The probabilistic one is a combination of sampling of translation space, random sampling of both input sets and the technique by Cheong et al. [2].

2. The translational case

First, we present a lower bound on the maximum area of overlap under translation.

Theorem 1 *Let $A = \{A_1, \ldots, A_n\}$ and $B = \{B_1, \ldots, B_m\}$, be two sets of disjoint unit disks in the plane. Let $\vec{t}_{opt}$ be the translation that maximizes the area of overlap $\mathcal{V}(\vec{t})$ of $A(\vec{t})$ and B over all possible translations $\vec{t}$ of set A. If k_{opt} is the number of overlapping pairs $A_i(\vec{t}_{opt})$ and B_j, then $\mathcal{V}(\vec{t}_{opt})$ is $\Theta(k_{opt})$.*

We proceed with the $(1-\epsilon)$-approximation algorithm. The algorithm is based on sampling of transformation space by using a uniform grid. This is possible due to the following lemma that states that $\mathcal{V}(\vec{t})$ is a sum of functions with some Lipschitz behaviour.

Lemma 2 *Let k_{opt} be the number of overlapping pairs $A_i(\vec{t}_{opt})$ and B_j. For any given $\delta > 0$ and any $\vec{t} \in \mathbb{R}^2$ for which $|\vec{t}_{opt} - \vec{t}| = O(\delta)$, we have $\mathcal{V}(\vec{t}_{opt}) - \mathcal{V}(\vec{t}) = O(k_{opt}\delta)$.*

Figure 1 shows algorithm TRANSLATION(A, B, ϵ).

TRANSLATION(A, B, ϵ):

(i) Initialize an empty binary search tree $\mathcal{S}$ with entries of the form $(\vec{t}, \mathcal{V}(\vec{t}))$ where $\vec{t}$ is the key.
(ii) For each pair of disks $A_i \in A$ and $B_j \in B$ do:
  (a) Impose a uniform grid of spacing $\Theta(\epsilon)$ on $T_{ij} = B_j \ominus A_i$.
  (b) For each grid point $\vec{t}_g \in T_{ij}$ do:
    – If $\vec{t}_g$ is in $\mathcal{S}$, then $\mathcal{V}(\vec{t}_g) := \mathcal{V}(\vec{t}_g) + \mathcal{V}_{ij}(\vec{t}_g)$ otherwise, insert $\vec{t}_g$ in $\mathcal{S}$ with $\mathcal{V}(\vec{t}_g) := \mathcal{V}_{ij}(\vec{t}_g)$.
(iii) Report the grid point $\vec{t}_{apx}$ that maximizes $\mathcal{V}(\vec{t}_g)$, and $\mathcal{V}(\vec{t}_{apx})$.

Fig. 1. Algorithm TRANSLATION(A, B, ϵ).

Theorem 3 *Let $A = \{A_1, \ldots, A_n\}$ and $B = \{B_1, \ldots, B_m\}$ be two sets of disjoint unit disks in the plane. Let $\vec{t}_{opt}$ be the translation that maximizes $\mathcal{V}(\vec{t})$. Then, for any given $\epsilon > 0$,* TRANSLATION(A, B, ϵ) *computes a translation*

$\vec{t}_{apx}$, *for which* $\mathcal{V}(\vec{t}_{apx}) \geq (1-\epsilon)\mathcal{V}(\vec{t}_{opt})$, *in* $O((mn/\epsilon^2)\log(m/\epsilon))$ *time.*
All these results hold for sets of convex homothets where the ratio of the areas of any two objects is bounded and any object intersects only a constant number of other objects in its set [3].

3. The rotational case

We consider the following restricted scenario: set B is fixed, and set A can be rotated around the origin. This problem has a one-dimensional configuration space: the angle of rotation. Consider the function $\mathcal{V} : [0, 2\pi) \to \mathbb{R}$ defined by $\mathcal{V}(\theta) := V(A(\theta) \cap B)$. We start with a result that states that the term $\mathcal{V}_{ij}(\theta)$ has some Lipschitz behaviour.

Lemma 4 *Let* A_i, B_j *be any fixed pair of disks. For any given* $\delta > 0$ *and any* θ_1, θ_2 *for which* $|\theta_1 - \theta_2| \leq \delta/(2r_i)$, *we have* $|\mathcal{V}_{ij}(\theta_1) - \mathcal{V}_{ij}(\theta_2)| \leq 2\delta$.

For a pair A_i, B_j, we define the rotational interval $R_{ij} = \{\theta \in [0, 2\pi) : A_i(\theta) \cap B_j \neq \emptyset\}$. Note that R_{ij} is connected under the considered topology; we denote its length by $|R_{ij}|$. Instead of computing $V_{ij}(\theta)$ at each $\theta \in R_{ij}$, we would like to sample it at regular intervals whose length is at most $\delta/(2r_i)$. At first, it looks as if we would have to take an infinite number of sample points as $r_i \to \infty$. However, as the following lemma shows, the number of samples we need to consider is bounded.

Lemma 5 *For any* A_i, B_j *with* $r_i > 0$, *and any given given* $\delta > 0$, *we have* $|R_{ij}|/(\delta/(2r_i)) = O(1/\delta)$.
Algorithm Rotation(A, B, δ), see Figure 2, maximizes $\mathcal{V}(\theta)$ up to an absolute error. By computing a value $\tilde{\mathcal{V}}(\theta)$ that approximates $\mathcal{V}(\theta)$, we save a linear factor in the running time.

Lemma 6 *Let* θ_{opt} *be the rotation that maximizes* $\mathcal{V}(\theta)$ *and let* k_{opt} *be the number of overlapping pairs* $A_i(\theta_{opt})$ *and* B_j. *For any given* $\delta > 0$, *the rotation* θ_{apx} *reported by* Rotation(A, B, δ) *satisfies* $\mathcal{V}(\theta_{opt}) - \mathcal{V}(\theta_{apx}) = O(k_{opt}\delta)$, *and can be computed in* $O((mn/\delta)\log m)$ *time.*

4. A $(1-\epsilon)$-approximation algorithm for rigid motions

A simple algorihtm RigidMotion(A, B, ϵ) which gives a $(1-\epsilon)$-approximation of the optimum is described in Figure 3. It uses the fact that any isometry can be expressed as a translation followed by a rotation around the origin.

Rotation(A, B, δ):

(i) For each pair of disks $A_i \in A$ and $B_j \in B$, choose points $\Theta_{ij} := \{\theta_{ij}^1, \ldots, \theta_{ij}^{s_{ij}}\}$ with an uniform spacing of $\delta/(2r_i)$ and such that $R_{ij} \subset [\theta_{ij}^1, \theta_{ij}^{s_{ij}}]$. Make sure that the midpoint of R_{ij} is in Θ_{ij}.
(ii) Sort the values $\Theta := \bigcup_{i,j} \Theta_{ij} = \{\theta_{ij}^s\}$, keeping repetitions and solving ties arbitrarily, if applicable. Let $\theta_0, \theta_1, \ldots$ be the ordering of Θ.
(iii) Let $\tilde{\mathcal{V}} : \Theta \to \mathbb{R}$ with $\tilde{\mathcal{V}}(\theta_0) := \mathcal{V}(\theta_0)$.
(iv) For each $\theta_l = \theta_{ij}^s$ in increasing order of l do:
- If $\mathcal{V}_{ij}$ is decreasing at θ_{ij}^s, or θ_{ij}^s is the midpoint of R_{ij}, then $\tilde{\mathcal{V}}(\theta_l) := \tilde{\mathcal{V}}(\theta_{l-1}) - \mathcal{V}_{ij}(\theta_{ij}^{s-1}) + \mathcal{V}_{ij}(\theta_{ij}^{s+1})$
- If $\mathcal{V}_{ij}$ is increasing at θ_{ij}^s, then $\tilde{\mathcal{V}}(\theta_l) := \tilde{\mathcal{V}}(\theta_{l-1}) - \mathcal{V}_{ij}(\theta_{ij}^{s-1}) + \mathcal{V}_{ij}(\theta_{ij}^{s})$

(v) Report the $\theta_{apx} \in \Theta$ that maximizes $\tilde{\mathcal{V}}(\theta)$.

Fig. 2. Algorithm Rotation(A, B, δ).

RigidMotion(A, B, ϵ):

(i) For each pair of disks $A_i \in A$ and $B_j \in B$ do:
(a) Set the center of rotation, i.e the origin, to be B_j's center by translating B appropriately. Impose a uniform grid of spacing $\Theta(\epsilon)$ on $T_{ij} = B_j \ominus A_i$.
(b) For each grid point $\vec{t}_g \in T_{ij}$ do:
- run Rotation$(A(\vec{t}_g), B, c\epsilon)$, where c is an appropriate constant; let θ_{apx}^g be the rotation returned. Compute $\mathcal{V}(\vec{t}_g, \theta_{apx}^g)$.

(ii) Report the pair (t_{apx}, θ_{apx}) that maximizes $\mathcal{V}(\vec{t}_g, \theta_{apx}^g)$, and $\mathcal{V}(t_{apx}, \theta_{apx})$.

Fig. 3. Algorithm RigidMotion(A, B, ϵ).

Theorem 7 *Let* $A = \{A_1, \ldots, A_n\}$ *and* $B = \{B_1, \ldots, B_m\}$ *be two sets of disjoint unit disks in the plane. Let* $I_{\vec{t}_{opt}, \theta_{opt}}$ *be the isometry that maximizes* $\mathcal{V}(\vec{t}, \theta)$. *Then, for any given* $\epsilon > 0$, RigidMotion(A, B, ϵ) *computes an isometry* $I_{\vec{t}_{apx}, \theta_{apx}}$, *for which* $\mathcal{V}(\vec{t}_{apx}, \theta_{apx}) \geq (1-\epsilon)\mathcal{V}(\vec{t}_{opt}, \theta_{opt})$, *in* $O((n^2m^2/\epsilon^3)\log m)$ *time.*
We can modify the algorithm such that its running time depends on the diameter Δ of the set A. The main idea is to convert our algorithm into one that is sensitive to the number of pairs of disks in A and B that have approximately the same distance, and then use the combinatorial bounds by Gavrilov et al. [4, Theorem 4.1]. In many applications it

is reasonable to assume bounds of the type $\Delta = O(n)$ [4], and therefore the result below is relevant.

Theorem 8 *Let $A = \{A_1, \ldots, A_n\}$ and $B = \{B_1, \ldots, B_m\}$ be two sets of disjoint unit disks in the plane. Let Δ be the diameter of A, and let $I_{\vec{t}_{opt},\theta_{opt}}$ be the isometry that maximizes $\mathcal{V}(\vec{t},\theta)$. Then, for any given $\epsilon > 0$, an isometry $I_{\vec{t}_{apx},\theta_{apx}}$ such that $\mathcal{V}(\vec{t}_{apx},\theta_{apx}) \geq (1-\epsilon)\mathcal{V}(\vec{t}_{opt},\theta_{opt})$ can be found in $O(\frac{m^2 n^{4/3}\Delta^{1/3}\log n \log m}{\epsilon^3})$ time.*

5. A Monte Carlo Algorithm

In this section we present a nearly quadratic Monte Carlo randomized algorithm that computes a $(1-\epsilon)$-approximation of the maximum area of overlap with high probability. The algorithm works under the assumption that $\mathcal{V}(\vec{t}_{opt},\theta_{opt}) \geq \alpha V(B) = \alpha m\pi$, for some $0 < \alpha \leq 1$, which implies that $n = \Theta(m)$ and $k_{opt} = \Theta(n)$. As noted by Cheong et al. [2], this is a reasonable assumption to make in many shape matching applications.

The first step is a combination of random sampling of set A and the sampling of the space of translations. This is based on the observation that the deterministic algorithm of Section 4 will compute a $(1-\epsilon)$-approximation k_{opt} times. Intuitively, the larger the k_{opt}, the less the neccessary number of pairs that have to be tried out in this step. The second step is based on a direct application of a technique by Cheong et al. [2]. The technique allows us to maximize, up to an absolute error, the area of overlap under rotation in linear time, by computing a point of maximum depth in a one dimensional arrangement. We sumarize it below.

We choose a uniform random sample S of points in B, and for a rotation θ we compute the estimate $e_S(\theta) = \frac{|A(\theta)\cap S|}{S}$. For a point $s \in S$ we define $W(s) = \{\theta | s \in A(\theta)\}$. Let $\Theta_A(S)$ be the arrangement of all regions $W(s)$; it is a one-dimensional arragement of rotational intervals.

Lemma 9 *Let θ_{opt} be the rotation that maximizes $\mathcal{V}(\theta)$. For any given $\delta > 0$, let S be a uniform random sample of points in B with $|S| \geq c_1 \frac{\log n}{\delta^3}$ where c_1 is an appropriate constant. The vertex θ_{apx} of $\Theta_A(S)$ that maximizes $e_S(\theta)$ satisfies $\mathcal{V}(\theta_{opt}) - \mathcal{V}(\theta_{apx}) \leq \delta V(B)$ with probability at least $1 - 1/n^6$.*

Algorithm RandomRigidMotion$(A, B, \alpha, c, \epsilon)$ is given in Figure 4.

Theorem 10 *Let $A = \{A_1, \ldots, A_n\}$ and $B = \{B_1, \ldots, B_m\}$, be two sets of disjoint unit disks in the plane. Let $I_{\vec{t}_{opt},\theta_{opt}}$ be the isometry that maximizes $\mathcal{V}(\vec{t},\theta)$ with the assumption that $\mathcal{V}(\vec{t}_{opt}, thopt) \geq \alpha V(B)$, for some $0 < \alpha \leq 1$. Then, for any given $\epsilon > 0$, $c \geq \alpha/6$, RandomRigidMotion$(A, B, \alpha, c, \epsilon)$ computes an isometry $I_{\vec{t}_{apx},\theta_{apx}}$, for which $\mathcal{V}(\vec{t}_{apx},\theta_{apx}) \geq (1-\epsilon)\mathcal{V}(\vec{t}_{opt},\theta_{opt})$, in $O((cn^2/\epsilon^5)\log^2(n/\epsilon))$ time. The algorithm succeeds with probability at least $1 - \epsilon^{-c} - 1/n^6$.*

RandomRigidMotion$(A, B, \alpha, c, \epsilon)$:

(i) Choose a uniform random sample $R_A \subset A$, with $|R_A| = \Omega(c/\alpha)$, and a uniform random sample S of points in B, with $|S| = \Omega(\log|A|/\epsilon^3)$.
(ii) For each pair of disks $A_i \in R_A$ and $B_j \in B$ do:
  (a) Impose a uniform grid of spacing $\Theta(\epsilon)$ on $T_{ij} = B_j \ominus A_i$.
  (b) For each grid point $\vec{t}_g \in T_{ij}$ do:
    – Compute a vertex θ^g_{apx} of maximum depth in $\Theta_{A(\vec{t}_g)}(S)$, and $\mathcal{V}(\vec{t}_g, \theta^g_{apx})$.
(iii) Report the pair $(\vec{t}_{apx}, \theta_{apx})$ that maximizes $\mathcal{V}(\vec{t}_g, \theta^g_{apx})$, and $\mathcal{V}(\vec{t}_{apx}, \theta_{apx})$.

Fig. 4. Algorithm RandomRigidMotion$(A, B, \alpha, c, \epsilon)$.

References

[1] H. Alt and L. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In J.R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 121–153. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1999.

[2] O. Cheong, A. Efrat, and S. Har-Peled. On finding a guard that sees most and a shop that sells most. In *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms*, page to appear, 2004.

[3] M. de Berg, P. Giannopoulos, C. Knauer, R. van Oostrum, and R. C. Veltkamp. The area of overlap of two unions of convex objects under translations. Technical Report UU-CS-2003-025, Institute of Information and Computing Sciences, Utrecht University, The Netherlands, 2003.

[4] M. Gavrilov, P. Indyk, R. Motwani, and S. Venkatasubramanian. Combinatorial and experimental methods for approximate point pattern matching. *Algorithmica*, 38(2):59–90, 2003.

[5] D. M. Mount, R. Silverman, and A. Y. Wu. On the area of overlap of translated polygons. *Computer Vision and Image Understanding*, 64:53–61, 1996.

Minimum weight pseudo-triangulations

Joachim Gudmundsson [a,1] Christos Levcopoulos [b]

[a] *Department of Mathematics and Computing Science, TU Eindhoven, 5600 MB, Eindhoven, the Netherlands.*
[b] *Department of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden.*

Abstract

We consider the problem of computing a minimum weight pseudo-triangulation of a set S of n points in the plane. We first present an $O(n \log n)$-time algorithm that produces a pseudo-triangulation of weight $O(wt(M(S)) \cdot \log n)$ which is shown to be asymptotically worst-case optimal, i.e., there exists a point set S for which every pseudo-triangulation has weight $\Omega(\log n \cdot wt(M(S)))$, where $wt(M(S))$ is the weight of a minimum spanning tree of S. We also present a constant factor approximation algorithm running in cubic time. In the process we give an algorithm that produces a minimum weight pseudo-triangulation of a simple polygon.

Pseudo-triangulations are planar partitions that recently received considerable attention mainly due to their applications in visibility [7], ray-shooting [3], kinetic collision detection [4], rigidity [10], and guarding [9].

A pseudo-triangle is a planar polygon that has exactly three convex vertices, called corners, with internal angle less than π. A pseudo-triangulation of a set S of n points in the plane is a partition of the convex hull of S into pseudo-triangles whose vertex set is exactly S.

A related problem is the problem of triangulating a point set. Minimizing the total length has been one of the main optimality criteria for triangulations and other kinds of partition. The complexity of computing a minimum weight triangulation is one of the most longstanding open problems in computational geometry and it is included in Garey and Johnson's [1] list of problems from 1979 that neither are known to be NP-complete, nor known to be solvable in polynomial time. As a result approximation algorithms for the MWT-problem have been considered. In this paper we consider the problem of computing a pseudo-triangulation of minimum weight (MWPT) which was posed as an open problem by Rote et al. in [8]. An interesting observation that makes the pseudo-triangulation favorable compared to a standard triangulation is the fact that there exists point sets where any triangulation, and also any convex partition (without Steiner points), has weight $\Omega(n \cdot wt(M(S)))$, while there always exists a pseudo-triangulation of weight $O(\log n \cdot wt(M(S)))$, where $wt(M(S))$ is the weight of a minimum spanning tree of the point set. We also present an approximation algorithm that produces a pseudo-triangulation whose weight is within a factor 27 times the weight of the MWPT. In comparison, the best constant approximation factor for the MWT-problem, Levcopoulos and Krznaric [6], which is proved to be achievable by a polynomial-time algorithm [6] is so much larger that it has not been explicitly calculated.

An edge/segment with endpoints in two points u and v of S will be denoted by (u, v) and its length $|uv|$ is equal to the Euclidean distance between u and v. Given a graph T on S we denote by $wt(T)$ the sum of all the edge lengths of T. The minimum spanning tree of S and the convex hull of S, denoted $M(S)$ and $CH(S)$ respectively, will be used frequently throughout the paper. Both structures can be computed in $O(n \log n)$ time.

1. A fast pseudo-triangulation

As mentioned in the introduction there exist point sets S where any triangulation will have

Email addresses: h.j.gudmundsson@tue.nl (Joachim Gudmundsson), christos@cs.lth.se (Christos Levcopoulos).

[1] Supported by the Netherlands Organisation for Scientific Research (NWO).

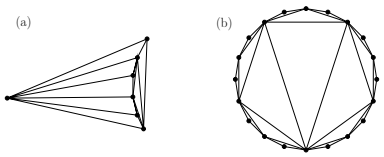


Fig. 1. (a) An example where any triangulation will have weight $\Omega(wt(M(S)) \cdot n)$. (b) An example where any pseudo-triangulation will have weight $\Omega(wt(M(S)) \cdot \log n)$.

weight $\Omega(n \cdot wt(M(S)))$, an example is given in Fig. 1a. A natural question is whether there exists similar worst-case bounds for pseudo-triangulations. In this section we show that one can always construct a pseudo-triangulation of weight $O(\log n \cdot wt(M(S)))$, and this is asymptotically tight, i.e., there exists a point set S for which every pseudo-triangulation has weight $\Omega(\log n \cdot wt(M(S)))$. We start with the lower bound.

Observation 1 *There exists a point set S in the plane such that any pseudo-triangulation has weight $\Omega(wt(M(S)) \cdot \log n)$.*

Next we present an algorithm that produces a pseudo-triangulation whose weight asymptotically meets the lower bound, that is:

Theorem 1 *Given a set S of n points in the plane one can in time $O(n \log n)$ produce a pseudo-triangulation of S of weight $O(wt(M(S)) \cdot \log n)$.*

The algorithm performs two main steps: first a partition of the convex hull of S into simple polygons $P_1, \ldots, P_m$ followed by a pseudo-triangulation of each polygon.

In this section we first show how a visibility polygon P can be pseudo-triangulated in time $O(n \log n)$ using edges of total weight $O(wt(P) \cdot \log n)$. We also show how to pseudo-triangulate a special polygon, called an hourglass polygon. Finally it is shown how one can construct a spanning graph of S that partitions the convex hull of S into empty polygons that either are visibility polygons, or hourglass polygons by using segments of small total weight. Combining these results gives us Theorem 1.

1.1. *Pseudo-triangulating a visibility polygon*

We will show that a weak visibility polygons whose visibility edge has two convex vertices easily can be pseudo-triangulated. This result will be used in the algorithm that pseudo-triangulates a visibility polygon. First a simple observation.

Observation 2 *The geodesic shortest path between any pair of points p and q in a weak visibility polygon P is a concave chain.*

Observation 3 *A weak visibility polygon P whose visibility edge (p_1, p_2) has two convex vertices can be pseudo-triangulated in time $O(n \log n)$ using edges of total weight $O(wt(P) \cdot \log n)$.*

Now we are ready to consider visibility polygons. Assume that we are given a visibility polygon P with respect to q with n vertices $p_1, \ldots, p_n$ ordered clockwise around the perimeter of P starting with q. Let $r_1, \ldots, r_m$ be the convex vertices of P.

Observation 2 implies that we can partition P into one pseudo-triangle and a set of weak-visibility polygons by adding the pseudo-triangle with corners at p_1, r_i and r_j, where $1 < i < j$. The two convex vertices r_i and r_j are chosen in such a way that the two angles $\angle p_2, p_1, r_i$ and $\angle p_n, p_1, r_j$ are less than π. Note also that p_2 and p_n are convex vertices since P is a visibility polygon. The pseudo-triangle will consist of the edges in the concave chain between r_i and r_j plus the edges (p_1, r_i) and (p_1, r_j). The resulting subpolygons outside the pseudo-triangle are weak visibility polygons whose visibility edges have convex vertices. According to Observation 3 each of these subpolygons can be pseudo-triangulated in $O(n \log n)$ time using edges of total weight $O(wt(P) \cdot \log n)$. Hence we have showed the following lemma.

Lemma 2 *The algorithm produces a pseudo-triangulation T of a visibility polygon P in $O(n \log n)$ time whose weight is $O(wt(P) \cdot \log n)$.*

We end this section by considering the pseudo-triangulation of an hourglass polygon. A polygon P is said to be an hourglass polygon if P consists of two concave chains connected by two edges.

We will later need the following straight-forward observation:

Observation 4 *An hourglass polygon P can be pseudo-triangulated in linear time by adding one edge e such that $wt(e) \leqslant 1/2 \cdot wt(P)$.*

1.2. *Partition a point set into simple polygons*

As input we are given a set S of n points in the plane, and as output we will produce a set of polygons that are either hourglass polygons or visibility polygons. The partition is done in two main steps. First construct the convex hull and the minimum spanning tree of S. This is done in $O(n \log n)$ time and it partitions $\mathrm{CH}(S)$ into simple (maybe de-

generate) polygons, denoted $P_1, \ldots, P_m$. Secondly, each polygon P_i is processed independently. The task at hand is to partition P_i into a set of hourglass polygons and "restricted" visibility polygons, which can be pseudo-triangulated as described in the previous section.

A *restricted* visibility polygon $rvp(P, q)$ of a polygon P with respect to a vertex q is a visibility polygon of P with respect to q such that every vertex of $P(q)$ also is a vertex of P.

Definition 3 *Every edge $e = (u, v)$ of a restricted visibility polygon $R(q)$ that short cuts exactly three edges of the maximal visibility polygon $P(q)$ is said to be a split edge.*

Now, let $v_1, \ldots, v_n$ be the vertices of P in clockwise order, starting at $q = v_1$. It remains to show how we can partition P into visibility polygons and hourglass polygons in $O(n \log n)$ time. The idea is to recursively partition P into restricted visibility polygons and hourglass polygons. Consider one level of the recursion. If P is not a restricted visibility polygon with respect to q, or an hourglass polygon then the following two steps are performed:

(i) Build a restricted visibility polygon $rvp(P, q)$ of P.

(ii) For each split edge e in $rvp(P, q)$ construct an hourglass polygon H such that $H \cap R(q) = e$.

A more precise description on how this can be performed in time $O(n \log n)$ can be found in the full version.

2. A MWPT of a simple polygon

Even though the above algorithm is asymptotically worst-case optimal with respect to the weight of the minimum spanning tree it can be very far from the optimal solution. In the rest of this paper we will focus on developing a constant factor approximation algorithm for the MWPT-problem. As a subroutine we will also develop an algorithm that finds an optimal pseudo-triangulation of a simple polygon.

Theorem 4 *Given a simple polygon P one can compute the minimum weight pseudo-triangulation of P in $O(n^3)$ time using $O(n^2)$ space.*

We will use a similar dynamic programming method as proposed by Gilbert [2] and Klincsek [5] for finding a minimum weight triangulation of a simple polygon. The basic observation used is that once some (pseudo-)triangle of the (pseudo-)triangulation has been fixed the problem splits into subproblems whose solutions can be found recursively, hence avoiding recomputation of common subproblems.

Let P be the simple polygon with n vertices $p_1, \ldots, p_n$ in clockwise order. Let $\delta(p_i, p_{i+j})$ be the shortest geodesic path between p_i and p_{i+j}. Define the *order* of a pair of points p_i, p_j to be the value $(i - j - 1) \mod n$, i.e., the number of vertices on the path from p_i to p_j along P in clockwise order. Sort the pairs with respect on their order, ties are broken arbitrarily. Note that every pair of points p_i and p_j will occur twice; once as (p_i, p_j) and once as (p_j, p_i). Now we process each pair in sorted order as follows.

Assume we are about to process (p_i, p_{i+j}) and that the path $\delta(p_i, p_{i+j})$ goes through the vertices $p_i = p_{i+a_0}, p_{i+a_1}, \ldots, p_{i+a_k} = p_{i+j}$. Note that the path partitions P into $k + 1$ subpolygons. Let $L[i, i + j]$ be the total edge length of an optimal pseudo-triangulation for the subpolygon (or subpolygons) containing the chain $p_i, p_{i+1}, \ldots, p_{i+j}$ of the perimeter of P. Compute $L[i, i + j]$ recursively as follows. If (p_i, p_{i+j}) is not a convex or concave chain then we set $L[i, i + j] = \infty$. In the case when the path is a concave or convex chain we obtain one polygon P' bounded by the path $\delta(p_i, p_{i+j})$ and the path between p_i and p_{i+j}, and k polygons $P_1, \ldots, P_k$ where each P_l is bounded by the edge $(p_{i+a_l}, p_{i+a_{l-1}})$ and the edges from $p_{i+a_{l-1}}$ to p_{i+a_l} along the perimeter of P. If the path is a concave or convex chain then we will have three cases.

- If $\delta(p_i, p_{i+j})$ contains more than one edge then we know that $L[*, *]$ already has been computed for every edge along $\delta(p_i, p_{i+j})$, hence we only have to add up the values of $L[*, *]$ which can be done in linear time, i.e., calculating $\sum_{\alpha=0}^{k-1} L[p_{i+a_\alpha}, p_{i+a_{\alpha+1}}]$.

- If $\delta(p_i, p_{i+j})$ contains exactly one edge (p_i, p_{i+j}) then an optimal pseudo-triangulation of P_1 can be obtained in linear time as follows. We will have two cases; either p_i and p_{i+j} are corners of the pseudo-triangle in P_1 containing (p_i, p_{i+j}) or not.

In the case when both p_i and p_{i+j} are convex vertices within P_1 then an optimal pseudo-triangulation of P_1 can be obtained in linear time as follows. Any optimal pseudo-triangulation of P_1 that contains the edge (p_i, p_{i+j}) must have p_i and p_{i+j} as corners thus we can try all possible vertices p_m, $i < m < i + j$ as the third corner.

Testing a pseudo-triangle with corners at p_i, p_{i+j} and p_m takes constant time since the $L[*,*]$-value of the paths between p_i and p_m, and p_m and p_{i+j} already has been computed.

Otherwise, if one or both of the points are not corners, it holds that there must be a pair of points p_x and p_y along the perimeter of P between p_i and p_{i+j} whose shortest geodesic path between them contains the edge (p_i, p_{i+j}). Hence, in this case the optimal solution has already been computed for P_1.

There are $O(n^2)$ pairs of points and each pair takes $O(n)$ time to process. The space bound follows from the fact that for every pair of points p_i and p_j we store $L[p_i, p_j]$. When all the $L[*,*]$ have been computed we can easily test every possible pseudo-triangle in constant time, thus Lemma 4 follows.

3. A better approximation

In this section we will give an approximation algorithm for the MWPT-problem. It is similar to the approximation algorithm presented in Section 1 in the sense that the two main steps are the same; first a partition of the convex hull of the point set into simple polygons followed by a pseudo-triangulation of each polygon. In the pseudo-triangulation step we will use the optimal algorithm presented in the previous section. As input we are given a set S of n points in the plane, and as output we will produce a pseudo-triangulation T of S.

Algorithm PseudoTriangulate(S)

(i) Construct the convex hull and the minimum spanning tree of S. This partitions $CH(S)$ into simple (maybe degenerate) polygons denoted $Q_1, \ldots, Q_k$.

(ii) Apply Theorem 4 to each of the k polygons. The pseudo-triangulation obtained together with the convex hull and the minimum spanning tree of S is reported.

The proof of the following theorem can be found in the full version of the paper.

Theorem 5 *Given a set of points S algorithm* PseudoTriangulate *computes a pseudo-triangulation T of S in time $O(n^3)$ using $O(n^2)$ space such that $wt(T) = 4(1+4\sqrt{2}) \cdot wt(T_{opt})$, where T_{opt} is a minimum weight pseudo-triangulation of S.*

4. Open problems and Acknowledgement

An obvious question is whether the minimum weight pseudo-triangulation problem is as hard as finding the minimum weight triangulation? The MWT-problem is one of the few open problems listed in Garey and Johnson's 1979 book on NP-completeness [1] that remain open today.

A second open problem concerning the weight of a pseudo-triangulation is if there exists a minimum pseudo-triangulation of low weight. It was shown by Streinu [10] that every point set allows a minimum planar pseudo-triangulation that has $2n - 3$ edges. Neither of the two algorithms presented in this paper produces minimum pseudo-triangulations, although the dynamic programming algorithm for simple polygons can be modified to compute a minimum weight minimum pseudo-triangulation.

The first author would like to thank Mark de Berg and Bettina Speckmann for valuable discussions during the work on this paper.

References

[1] M. Garey and D. Johnson. Computers and Intractability. W. H. Freeman and Company, 1979.

[2] P. D. Gilbert. New results in planar triangulations. Report R–850, Univ. Illinois Coord. Science Lab, 1979.

[3] M. T. Goodrich and R. Tamassia. Dynamic Ray Shooting and Shortest Paths in Planar Subdivisions via Balanced Geodesic Triangulations. Journal of Algorithms, 23(1):51–73, 1997.

[4] D. Kirkpatrick and B. Speckmann. Kinetic Maintenance of Context-Sensitive Hierarchical Representations for Disjoint Simple Polygons. Proc. 18th ACM Symp. on Computational Geometry, 2002.

[5] G. Klincsek. Minimal triangulations of polygonal domains. Annals of Discrete Math., 9:121-123, 1980.

[6] D. Krznaric and C. Levcopoulos. Quasi-Greedy Triangulations Approximating the Minimum Weight Triangulation. J. of Algorithms 27(2): 303-338. 1998.

[7] M. Pocchiola and G. Vegter. Pseudo-triangulations: Theory and applications. In Proc. 12th ACM Symposium on Computational Geometry, 1996.

[8] G. Rote, C. A. Wang, L. Wang, and Y. Xu. On constrained minimum pseudotriangulations. Proc. 9th Symposium on Computing an Combinatorics, 2003.

[9] B. Speckmann and C. D. Tóth. Allocating Vertex piguards in Simple Polygons via Pseudo-Triangulations. Proc. 14th ACM-SIAM Symp. on Discrete Alg., 2003.

[10] I. Streinu. A Combinatorial Approach to Planar Non-Colliding Robot Arm Motion Planning. Proc. 41st ACM Symposium on Found. of Comp. Sci., 2000.

Computing the convex hull of disks using only their chirotope

Luc Habert and Michel Pocchiola
Département d'Informatique, Ecole Normale Supérieure, 45 rue d'Ulm 75230 Paris, France.
Luc.Habert@ens.fr, Michel.Pocchiola@ens.fr

Abstract

We show that the convex hull of a collection of n pairwise disjoint disks in the plane is computable in $O(n \log n)$ time using only the chirotope of the collection of disks. The method relies mainly on the development of an (elementary) theory of convexity in the universal covering space of the punctured plane.

1. Introduction

1.1. *Result of the paper.*

Throughout the paper we consider a finite family o_i of $n \geq 2$ pairwise disjoint bounded closed 2-dimensional convex sets in the plane with regular boundaries (*disks* for short) and we assume that the disks are in general position in the sense that there is no line tangent to three disks. A bitangent is a closed line-segment tangent to two disks at its endpoints. The chirotope of the family of disks is defined as the map χ that associates with each ordered triplet (u, v, w) of bitangents tangent to a same disk o the value $+1$ if walking counterclockwise around the boundary of o we encounter the bitangents u, v, w in cyclic order $u, v, w, u, v, \ldots$; the value -1, otherwise. The main result of the paper is the following.

Theorem 1 *The convex hull of a collection of n pairwise disjoint disks in the plane is computable in $O(n \log n)$ time using only the chirotope of the collection of disks.* □

1.2. *Previous work.*

Several algorithms have been developed in the past to address the convex hull problem for disks in the plane : the set of hull-bitangents can be computed as the set of breakpoints of the upper envelope of the support functions of the o_i using a divide-and-conquer algorithm [6, chap. 6] as described in [5], running in $O(n \log n)$ time but making use of the chirotope of the family of directions of the set of bitangents of the set of disks augmented with a point at infinity. More sophisticated techniques – using even more involved predicates like slicing the disks – have been developped to design output sensitive convex hull algorithm [4]. (The related problem of computing the convex hull of a simple curved polygon is adressed in [1].)

For point obstacles the situation is different : Graham's scan [2] and Knuth's incremental algorithm [3] both compute the convex hull of a set of points using only its chirotope.

1.3. *Motivations.*

Applied motivations We have devised an algorithm that computes a pseudo-triangulation given the convex-hull. This algorithm also runs in $O(n \log n)$ and uses only the chirotope. Now, pseudotriangulations are useful data structures.

Implementation motivations The chirotope shows fewer degenerate cases than the more involved predicates used in previous algorithms, and our algorithms use simpler data-structures.

Theoretical motivations The convex hull depends only on the chirotope, not on the more involved predicates used in existing algorithms. Computing these objects by means of the most basic predicates possible is interesting in its own right. It can also lead to isolating a family of axioms satisfied by the chirotopes, in the style of Knuth's CC-systems.

2. Convexity in the universal covering space of the punctured plane.

2.1. *Definitions and notations*

Let $p : \tilde{\mathbb{P}} \longrightarrow \mathbb{P}$ be a universal covering of the punctured plane $\mathbb{P} = \mathbb{R}^2 \setminus \{(0,0)\}$. The reader might have in mind as a model of the map p the vertical projection upon plane $z = 0$ of the screw surface $\{(r\cos\theta, r\sin\theta, \theta) \in \mathbb{R}^3 \mid (\theta, r) \in \mathbb{R} \times \mathbb{R}^{+*}\}$. θ is called the *angle-coordinate* of point (θ, r) of $\tilde{\mathbb{P}}$ and r is called its distance-coordinate. As the reader might suspect we use the geometric structure on $\tilde{\mathbb{P}}$ inherited via the covering map from the euclidean structure on the punctured plane. Thus a line (half-line, line segment) of $\tilde{\mathbb{P}}$ is a connected component of the pre-image under the covering map of a line (half-line, line-segment) of $\mathbb{P}$. A line will be regarded as oriented towards growing angle-coordinate. Define the *angle* (denoted $\hat{L}$) of a line L, as the least upper bound of the angle coordinate of its points. A line L is uniquely identified by $\hat{L}$ and the distance to the origin of its projection onto $\mathbb{P}$, denoted $\delta(L)$.

Points A and B in $\tilde{\mathbb{P}}$ are said visible if there is a line-segment in $\tilde{\mathbb{P}}$ with endpoints A and B. A subset X of $\tilde{\mathbb{P}}$ is called convex if for any pair (A, B) of visible points in X, line-segment $[AB]$ is included in X. An intersection of convex sets is a convex set, which allows to define the convex hull of a set as the smallest (for the inclusion relation) convex set containing the set. An half-plane of $\tilde{\mathbb{P}}$ is the closure of a connected component of the complement of a line of $\tilde{\mathbb{P}}$. The projection of an half-plane under p is either the whole punctured plane $\mathbb{P}$ or an half-plane of $\mathbb{P}$.

A *s*imple convex of $\tilde{\mathbb{P}}$ is a connected component of the pre-image under p of a closed bounded convex subset of $\mathbb{P}$. A *l*ong convex of $\tilde{\mathbb{P}}$ is a closed convex that contains a point (θ, r) for any θ and some r depending on θ, and that is bounded in the r direction for any θ. The pre-image under p of a closed bounded convex subset of $\mathbb{R}^2$ containing the origin of the plane in its interior is an example of a long convex.

A positive supporting line of convex is a directed line containing boundary points such that the convex lies on the left side of the directed line.

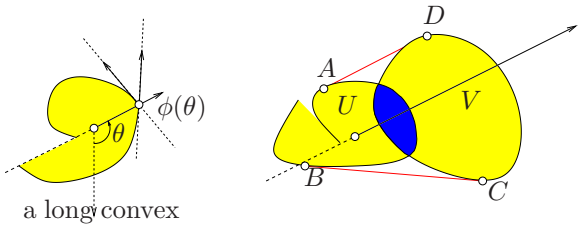


Fig. 1. A simple comvex and a long convex share 2 common exterior tangents. The drawaing is done in the sheet of the disk = the sheet of the point of the disk that realizes its distance to the origin.

2.2. *Some theorems*

Theorem 2 *The border of a long convex is a curve $\theta \to (\theta, \phi(\theta))$ that has a left-hand and a right-hand tangent for every θ, such that if they differ, the right-hand one has a larger angle than the left-hand one, and the angle of the left- or right-hand tangent increases with θ. The tangents to the border are positive supporting lines (and vice-versa).* □

Let U be a convex and let x be a point. A boundary point y of U is said visible from x if line segment $[x, y)$ exists and U are disjoint. The set of boundary points of U visible from y is a closed connected set.

Theorem 3 *Let U be a long convex, and V a simple convex, such that $V \cap U = \emptyset$ or V is not included in U and ∂U intersects ∂V in exactly two points that are not angular points of both ∂U and ∂V. Then U and V share exactly two positive common supporting lines, and the border of their convex hull is made of two half-unbounded connected sub-arcs $-\infty B$ and $A+\infty$ of ∂U, a connected sub-arc CD of ∂V and two bitangent segments BC and CA to U and V where BA (rep. DC) is the set of boundary points of U (resp. V) visible from V (resp. U).* □

3. Our algorithm.

3.1. *Notations.*

Wlog, assume $(0,0) \in o_1$. Every o_i $(i > 1)$ has infinitely many simple convex lift-up in $\tilde{\mathbb{P}}$, while o_1 has a single lift-up, $\tilde{o}_1$, a long convex. Choose one lift-up of o_2 and denote it $o_{2,1}$. If I and J are lift-up of o_i and o_j for some i and j, we define $v_{\epsilon I \epsilon' J}$ as the lift-up of $v_{\epsilon i \epsilon' j}$ whose endpoints belong in I and J, when such a lift-up actually exists (for instance this is always the case between $\tilde{o}_1$ and any

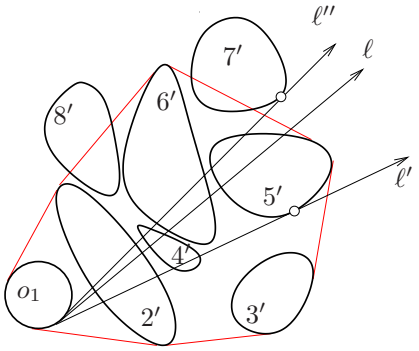


Fig. 2. In this example the disks $o_{i'}$ is labelled i', $B(\ell) = v_{1'2'}v_{2'3'}v_{3'5'}v_{5'6'}v_{6'2'}v_{2'1'}$, $D(\ell) = \{1', 2', 3', 4', 5', 6'\}$, and $Q(\ell) = [1, 2', 6', 5']$.

lift-up of any o_i with $i > 1$). Wlog, assume $v_{\tilde{o}_1 o_{2,1}}$ has angle 0. Denote $o_{i,k}$ the lift-up of o_i such that $v_{\tilde{o}_1 o_{i,k}}$ has angle in $[2(k-1)\pi, 2k\pi)$. Finally, denote $\ell_{i,k}$ the half-line, supporting $v_{\tilde{o}_1 o_{i,k}}$, with origin its tangency point upon $\tilde{o}_1$.

We perform a rotational sweep, with a half-line ℓ tangent to $\tilde{o}_1$ at its origin. The sweep starts at position $\ell = \ell_{1,1}$. During the sweep, we keep track of a subset of the objects that intersect ℓ, therefore, we will stop when ℓ reaches every $\ell_{i,k}$ when $i > 1$ (an "enter event"), and some $\ell_{i,k}$ when $i < -1$ (a "leave event"). We define a total order on the set of half-lines leaving $\tilde{o}_1$ by setting $\ell \prec \ell'$ if and only if $\hat{\ell} < \hat{\ell}'$. Computationally, if $i > 0$ and $j > 0$, then $\ell_{i,k} \prec \ell_{j,l}$ if and only if $k < l$ or $k = l$ and $\chi(v_{12}, v_{1i}, v_{1j})$ (or $i = 2$), so that we can sort the events (if, say $i < 0$ and $\chi(v_{1-i}, v_{12}, v_{1i})$, substitute $k+1$ to k). Then for a position ℓ of the sweep half-line, we define :

(i) the counterclockwise sequence $B(\ell) = v_1 v_2 \dots v_k$ of bitangents of the convex hull $C(\ell)$ of the set $D(\ell)$ of disks $o_{j,k}$ such that $\ell_{j,k} \preceq \ell$ – by convention $\tilde{o}_1$ is an element of $D(\ell)$ –, starting from $v_1 = v_{\tilde{o}_1 o_{2,1}}$; the arc with source v_i and sink v_{i+1} is denoted a_i and its supporting object o'_i (a_0 and a_k are half-unbounded loops around $\tilde{o}_1$). In fact, we sometimes regard B as the list of arcs a_i.

(ii) the arc $a(\ell)$ defined as the arc $a_{j'}$ where j' is the minimal element of the subset of indexes j ($1 \le j \le k$) such that ℓ pierces the supporting disks o'_i of a_i ($j \le i \le k$) in the order $o'_k, o'_{k-1}, \dots, o'_j$ ($o'_k = o_1$). The list $o'_k o'_{k-1} \dots o'_{j'}$ is denoted $Q(\ell)$.

See Figure 2 for an illustration. We write ℓ^- for any half-line of the open interval (ℓ', ℓ) where ℓ' is the previous event.

During the sweep, we maintain $B(\ell)$, $Q(\ell)$ and $a(\ell)$. Initially $\ell = \ell_{2,1}$, $D(\ell) = \{\tilde{o}_1, o_{2,1}\}$, $B(\ell) = [v_{\tilde{o}_1 o_{2,1}}; v_{o_{2,1}\tilde{o}_1}]$, $a(\ell)$ is the arc with source $v_{\tilde{o}_1 o_{2,1}}$ and sink $v_{o_{2,1}\tilde{o}_1}$, and $Q(\ell) = \tilde{o}_1 o_{2,1}$. We store Q in a binary search tree. To that end, we need to be able to decide given two objects $o_{i,k}$ and $o_{j,l}$ intersecting ℓ which one is to the right of the other along ℓ. This can be done with the procedure "if $\chi(v_{1j}, v_{1i}, v_{1-j})$ then $\chi(v_{ij}, v_{1i}, v_{i-j})$ else $\chi(v_{1j}, vji, v_{j-i})$".

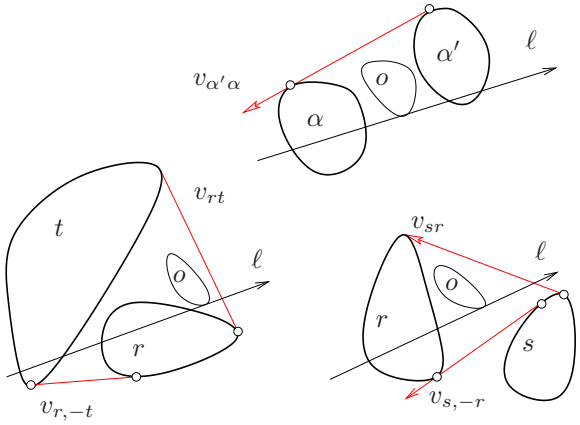


Fig. 3. o_i is included in the current convex hull.

3.2. *Handling enter-events.*

Assume we are to process the enter-event for object o. Let r be the rightmost disk of $Q(\ell^-)$, s its predecessor along $B(\ell^-)$, and t its successor (if any, i.e., if $r \neq \tilde{o}_1$).

Theorem 4 *l pierces either $a(\ell^-)$, or $v_{r(\ell^-)t(\ell^-)}$ or $v_{s(\ell^-)r(\ell^-)}$.*

Theorem 5 *Assume that o appears along ℓ at the left of disk r, between, say, disks α and α'. Then either o is included in $C(\ell^-)$, or o intersects $v_{\alpha'\alpha}$. The latter case occurs if and only if $p(o)$ intersects bitangent $v_{p(\alpha')p(\alpha)}$, and then the conditions of theorem 3 are satisfied.* □

Theorem 6 *Assume that o appears along ℓ at the right of disk r. Then either arc $a(\ell^-)$ contains a point visible from o or o is included in $C(\ell^-)$. The latter case stands if and only if either (non exclusive this time) $p(o)$ is included in the convex hull of $p(r)$ and $p(t)$ (in that case, ℓ pierces v_{sr} or $v_r t$), or $p(o)$ is included in the pseudotriangle bounded by the bitangents $v_{p(s)p(r)}$ and $v_{p(s),-p(r)}$, and $p(s)$ is not above $p(\ell)$ and $s + (2\pi, 0)$ has not been inserted yet*

(in that case l pierces through $v_{s(\ell)r(\ell)}$). See Figure 3 for an illustration. Also, when o is not included in $C(\ell^-)$, the conditions of theorem 3 are satisfied. □

Now, we explain how to update B and Q. First we locate o in Q.

Assume first that *o is to the left of r*. Let α and α' be its left-hand and right-hand neighbours in $Q(\ell^-)$. If $p(o)$ does not intersect $v_{p(\alpha)p(\alpha')}$ we just ignore o : $B(\ell) = B(\ell^-)$, $Q(\ell) = Q(\ell^-)$, and $a(\ell) = a(\ell^-)$. Otherwise we insert o into Q, and update B : we split B at v, and we regard the two resulting parts as stacks of arcs whose respective heads are the arcs contributed by α and α'. Then we pop from the left-hand stack until an arc β, say supported by disk o', is met such that $o' = \tilde{o}_1$ or $\chi(u, v, w)$ where u is the source of $p(\beta)$, w is its sink, and $v = v_{p(o)p(o')}$. Similarly we pop from the right-hand stack until an arc β', say supported by disk o'', is met such that $\chi(u, v', w)$ where u is the source of $p(\beta')$, w is its sink, and $v'' = v_{p(o'')p(o)}$. Then, we shorten β (respectively β') : its source (respectively sink) is replaced with $v_{oo'}$) (respectively $v_{o''o}$). Then, to build $B(\ell)$, we concatenate what is left of the two stacks, with the arc (denoted δ) of ∂o with source $v_{o''o}$ and sink $v_{oo'}$ in between. When an arc that follows $a(\ell^-)$ (included) in $B(\ell^-)$ is popped, its supporting disk is removed from Q. If $a(\ell^-)$ is removed from the right-hand stack, denoting u the object supporting the predecessor of δ along $B(\ell)$, we insert u into q if $u + (2\pi, 0)$ has not undergone an enter-event yet and $p(u)$ intersects $p(\ell)$ at the right of $p(o)$, so that u becomes $r(\ell)$ instead of o.

Assume now that *o is to the right of r*. We discard o when one of the two cases stated in theorem 6 holds. Otherwise, we proceed as in the previous case, with three exceptions : we split $B(\ell^-)$ through arc a instead of bitangent $v_{\alpha'\alpha}$ (that is, there is one copy of a at the head of both stacks); the copy of a at the top of the left-hand stack is popped if and only if $\chi(v_{ro}, v_{or}, v_{rt})$; an object is removed from Q if only if an arc it supports is popped from the left-hand stack.

The predicates used to decide whether o should be discarded can be implemented by means of χ, we omit the details.

3.3. *Handling leave events.*

In fact, only leave events for r need to be processed. The processing of those events simply consists in removing r from Q.

3.4. *Sketch of a proof of correction.*

From theorems 5, 6 and 3, if follows that : (1-) when o is included in $C(\ell^-)$, B is not updated, so that $B(\ell)$ is still the border of $C(\ell)$; (2-) when o is not included in $C(\ell^-)$, the border of $C(\ell)$ is made of two connected arcs of the border of $C(\ell^-)$, one connected arc of the border of o and two bitangents to o. What our algorithm does is a two-way walk along the border of $C(\ell^-)$, starting from a point known to be in the arc of $C(\ell^-)$ that is to be discarded, until discovery of the two bitangents.

3.5. *Extracting the planar convex hull.*

We could prove that after only two rounds (that is, at $\ell^-_{2,3}$), $p(B)$ contains the convex hull of $o_1, \ldots, o_n$ as a factor. But we do not know how to identify efficiently that factor based solely on the chirotope. Hence the need for a third round. Denote B the value of B obtained at the end of the third round.

Theorem 7 *Let γ be the first arc of B (for counter-clockwise orientation) supported by some $o_{i,2}$. Then γ indeed exists, the bitangent $b' = b + (2\pi, 0)$ appears in B (it is created during the third round), and the projection of the factor $\gamma M b'$ of B is the convex hull of the o_i.* □

References

[1] C. Bajaj and M. S. Kim. Convex hull of objects bounded by algebraic curves. *Algorithmica*, 6:533–553, 1991.

[2] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Inform. Process. Lett.*, 1:132–133, 1972.

[3] D. E. Knuth. *Axioms and Hulls*, volume 606 of *Lecture Notes Comput. Sci.* Springer-Verlag, Heidelberg, Germany, 1992.

[4] F. Nielsen and M. Yvinec. Output-sensitive convex hull algorithms of planar convex objects. *Internat. J. Comput. Geom. Appl.*, 8(1):39–66, 1998.

[5] D. Rappaport. A convex hull algorithm for discs, and applications. *Comput. Geom. Theory Appl.*, 1(3):171–181, 1992.

[6] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications.* Cambridge University Press, New York, 1995.

Optimal Spanners for Axis-Aligned Rectangles [1]

Tetsuo Asano [a] Mark de Berg [b] Otfried Cheong [b] Hazel Everett [c] Herman Haverkort [d]
Naoki Katoh [e] Alexander Wolff [f]

[a] *JAIST, Japan*
[b] *TU Eindhoven, the Netherlands*
[c] *LORIA, France*
[d] *Utrecht University, the Netherlands*
[e] *Kyoto University, Japan*
[f] *Universität Karlsruhe, Germany*

1. Introduction

Geometric networks arise frequently in our everyday life: road networks, telephone networks, and computer networks are all examples of geometric networks that we use daily. They also play a role in disciplines such as VLSI design and motion planning. Almost invariably, the purpose of the network is to provide a connection between the nodes in the network. Often it is desirable that the connection through the network between any pair of nodes be relatively short. From this viewpoint, one would ideally have a direct connection between any pair of nodes. This is usually infeasible due to the costs involved, so one has to compromise between the quality and the cost of the connections.

For two given nodes in a graph, the ratio of their distance in the graph and their 'direct' distance is called the *dilation* or *stretch factor* for that pair of nodes, and the dilation of a graph is the maximum dilation over all pairs of nodes. For geometric networks, this is more precisely defined as follows. Let S be a set of n points (in the plane, say), and let $\mathcal{G}$ be a graph with node set S. Now the dilation for a pair of points p, q is defined as the ratio of the length of the shortest path in $\mathcal{G}$ between p and q, and the length of the segment pq. (The length of a path is the sum of the lengths of its edges.) Again, the dilation of $\mathcal{G}$ is the maximum dilation over all pairs of points in S. A graph with dilation t is called a *t-spanner*. Ideal networks are t-spanners for small t with small cost.

Spanners were introduced by Peleg and Schäffer [6] in the context of distributed computing, and by Chew [1] in the context of computational geometry. They have attracted much attention since—see for instance the survey by Eppstein [2]. The cost of spanners can be measured according to various criteria. For example, it is sometimes defined as the number of edges (here the goal is to find a spanner with $O(n)$ edges), or as the total weight of the edges (here the goal is to find a spanner whose total weight is a constant times the weight of a minimum spanning tree). Additional properties, such as bounding the maximum degree or the diameter, have been considered as well.

We generalize the notion of spanners to geometric networks whose nodes are rectangles rather than points. Let S be a set of n non-intersecting, axis-parallel rectangles and let E be a set of axis-parallel segments connecting pairs of rectangles. For any two points p, q in the union of the rectangles, the dilation is now the ratio of the length of the shortest rectilinear path in the network between p and q and their L_1-distance. Here a path in the network is a path that stays within the union of the rectangles and the connecting segments. The dilation of the network is the maximum dilation over all pairs p, q. Again, our aim is to construct a network whose dilation is small. To illustrate the concept, imagine one is given a number of rect-

Email addresses: t-asano@jaist.ac.jp (Tetsuo Asano), m.t.d.berg@tue.nl (Mark de Berg), ocheong@win.tue.nl (Otfried Cheong), everett@loria.fr (Hazel Everett), herman@cs.uu.nl (Herman Haverkort), naoki@archi.kyoto-u.ac.jp (Naoki Katoh), awolff@ira.uka.de (Alexander Wolff).

[1] Part of this research was done during the First Utrecht-Carleton Workshop on Computational Geometry. H.H. acknowledges support by the Netherlands' Organization for Scientific Research (NWO).

angular buildings, which have to be connected by footbridges. It is quite frustrating if, to walk to a room opposite ones own room in an adjacent building, one has to walk all the way to the end of a long corridor, then along the footbridge, and then back again along the corridor in the other building. Hence, one would usually place the footbridge in the middle between buildings. Following this analogy, we will call the rectangles in the input *buildings* from now on, and the connecting segments *bridges*. We call the underlying graph of the network the *bridge graph*.

The generalization we study introduces one important additional difficulty in the construction of a spanner: for points one only has to decide *which* edges to choose in the spanner, but for buildings, one also has to decide *where* to place the bridge between a given pair of buildings. It is the latter problem we focus on in this paper: we assume the topology of the network (the bridge graph) is given, and our only task is to place the bridges so as to minimize the dilation.

Formally, our problem can be stated as follows: we are given a set S of axis-parallel disjoint rectangles (buildings) in the plane, a graph $\mathcal{G}$ with node set S, and for each arc e of $\mathcal{G}$ a *bridge region* Λ_e, an axis-aligned rectangle connecting the two buildings. Buildings may degenerate to segments or points. The bridge graph $\mathcal{G}$ must only have arcs between buildings that can be connected by a horizontal or vertical segment, and may not have multiple edges or loops. The bridge regions must be disjoint from each other and the buildings. Our goal is to find a set of horizontal or vertical bridges lying in the bridge regions that has minimum dilation.

Figure 1 shows a bridge graph (the bridge regions are shaded) and a set of possible bridges. Note that the bridge regions Λ_2 and Λ_3 simply allow any bridge between the two buildings, but bridge region Λ_1 has been chosen so as to avoid intersecting s_4 or the bridge between s_3 and s_4.

Our results are as follows.

- In general, the problem is NP-hard.
- If the bridge graph is a tree, then the problem can be solved by a linear program with $O(n^2)$ variables and constraints.
- If the bridge graph is a path, then the problem can be solved in $O(n^3 \log n)$ time.
- If the bridge graph is a path and the buildings are sorted vertically along this path, the problem can be solved in time $O(n^2)$. A $(1+\varepsilon)$-approximation can be computed in linear time.

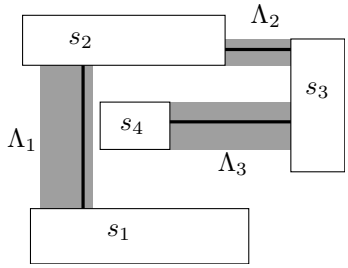


Fig. 1. A bridge graph and a bridge configuration

2. The bridge graph is arbitrary

The bridge-placement problem is NP-hard if the bridge graph is allowed to be arbitrary. We prove this by a reduction from PARTITION. The input to PARTITION is a set B of n positive integers, and the task is to decide whether B can be partitioned into two subsets of equal sum. PARTITION is NP-hard [3, Problem SP12].

Theorem 1 *It is NP-hard to decide whether the bridges in a given bridge graph on n rectangular buildings can be placed such that the dilation is at most* 2.

3. The bridge graph is a tree

In this section we will show that the bridge-placement problem can be solved by a linear program if the bridge graph is a tree. We start by introducing some terminology and notation, and by proving some basic lemmas. As before, we denote the bridge graph by $\mathcal{G}$. Any set of bridges realizing $\mathcal{G}$ will be called a *configuration*.

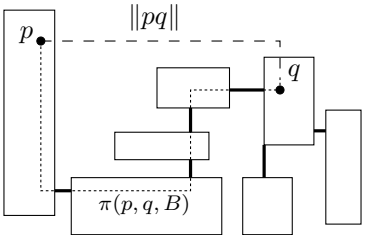


Fig. 2.

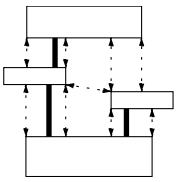


Fig. 3.

Given a configuration B and two points p and q in the union of all buildings, we use $\pi(p, q, B)$ to denote the family of rectilinear shortest paths from p to q within the configuration (that is, paths whose links lie inside buildings or on bridges). The paths of this family are essentially the same, they differ only in how they connect two points inside the same building, and so we will simply speak about *the unique path* $\pi(p, q, B)$. The *dilation* of the path $\pi = \pi(p, q, B)$ is $\mathrm{dil}(\pi) := |\pi|/\|pq\|$, where $|\pi|$ is the total length of π and $\|pq\|$ is the L_1-distance of p and q. Figure 2 shows a configuration and an example path.

The *dilation* $\mathrm{dil}(B)$ *of a configuration* B is defined as the maximum dilation of any path with respect to B. Our aim is to find a configuration of minimum dilation. We first characterize pairs of points that are responsible for the dilation of a given configuration.

Lemma 2 *Let σ be the dilation of a configuration B whose underlying graph is a tree. Then there are points p and q with $\mathrm{dil}(\pi(p, q, B)) = \sigma$ such that the closed bounding box of p and q does not contain any point of a building other than p and q, and at least one of the points p and q is a building corner.*

A point pair (p, q) as in the lemma—its bounding box contains no other point of any building and at least one of p and q is a building corner—will be called a *visible pair*—see Figure 3 for examples. We denote the set of all visible pairs by $\mathcal{V}$.

Given a bridge graph $\mathcal{G}$, our goal is to minimize

$$\max_{(p,q)\in\mathcal{V}} \mathrm{dil}(\pi(p, q, B))$$

over all configurations B realizing $\mathcal{G}$. We show that this problem can be reformulated as a linear program.

Theorem 3 *If the bridge graph $\mathcal{G}$ is a tree, then a placement of the bridges that minimizes the dilation can be computed by solving a linear program with $O(n^2)$ variables and constraints, where n is the number of bridges in the bridge graph.*

4. The bridge graph is a path

In the previous section we have given a linear program for the bridge-placement problem for the case where the bridge graph is a tree. Linear programs can be solved in practice, and for integer coefficients, interior-point methods can solve them in time polynomial in the bit-complexity of the input [4]. It is not known, however, if they can be solved in polynomial time on the real RAM, the standard model of computational geometry. In this section, we give polynomial time algorithms for the case where the bridge graph is a path.

Since the bridge graph $\mathcal{G}$ is a path, we can number the buildings and bridges so that bridge b_i connects buildings s_{i-1} and s_i, for $1 \le i \le n$ (so there are $n + 1$ buildings and n bridges). Before we continue, we need to introduce some more terminology. We consider a path $\pi = \pi(p, q, B)$ to be oriented from p to q. After traversing a bridge b, the path can continue straight on to traverse the next bridge b' if b and b' are collinear. In all other cases, it has to turn.

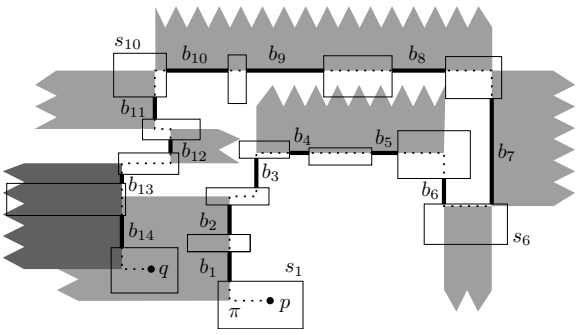


Fig. 4. U-turns and their outer sides

Given a path π, a *link* ℓ of π is a maximal straight segment of the path. A link can contain more than one bridge if they are collinear. For example, in Figure 4 there is a link containing b_1 and b_2, and another link containing b_8, b_9, and b_{10}.

The path π turns at both ends of a link (except for the first and last link). The link is a *right U-turn* if π turns right before and after the link. A *left U-turn* is defined symmetrically. In Figure 4, the links containing bridges (b_1, b_2), (b_4, b_5), and b_{12} are right U-turns, while the links containing b_7, (b_8, b_9, b_{10}), b_{11}, and (b_{13}, b_{14}) are left U-turns. Note that there can be U-turns that do not contain

any bridges, as the link of π inside building s_6 in Figure 4.

The *inner side* and *outer side* of a U-turn are rectangular regions infinite on one side, and bounded by the line supporting the link and the two lines orthogonal to it through the first and last points of the link. The outer side lies locally to the left of a right U-turn, or to the right of a left U-turn, the inner side lies locally to the right of a right U-turn or to the left of a left U-turn. In Figure 4, the outer sides of all U-turns are shaded.

U-turns are the links of a path that determine its dilation, as the following lemma shows.

Lemma 4 *Let B and B' be configurations, (p, q) a visible pair, and $\pi := \pi(p, q, B)$ and $\pi' := \pi(p, q, B')$ the paths between p and q with respect to the two configurations. If $\mathrm{dil}(\pi') < \mathrm{dil}(\pi)$ then there exists a U-turn ℓ containing $b_i \dots b_j$ of π such that the corresponding bridges $b'_i, \dots, b'_j$ of B' lie strictly on the inner side of ℓ.*

We will give an algorithm that takes as input the set of buildings $s_0, \dots, s_n$ and a real number $\sigma > 1$, and computes a configuration B with $\mathrm{dil}(B) \leq \sigma$, or determines that no such configuration exists.

The algorithm computes n sets $I_1, I_2, \dots, I_n$, where I_i is a set of possible bridges between s_{i-1} and s_i. The sets are defined recursively as follows. Assume that $I_1, \dots, I_{i-1}$ have already been defined. For each visible pair (p, q) with $p \in \bigcup_{j=0}^{i-1} s_j$ and $q \in s_i$ we define $I(p, q)$ as the set of bridges b_i connecting s_{i-1} and s_i such that the following holds: there is a set of bridges $b_1 \in I_1$, $b_2 \in I_2$, $\dots$, $b_{i-1} \in I_{i-1}$ such that $\mathrm{dil}(\pi(p, q, (b_1, \dots, b_i))) \leq \sigma$. Finally, I_i is the intersection of all $I(p, q)$.

Note that for each visible pair (p, q) we can choose the bridges in $I_1, \dots, I_{i-1}$ independently. This makes it possible to compute I_i efficiently, as we will see below. On the other hand, it implies that not every sequence of bridges chosen from the sets will be a configuration with dilation at most σ—our main lemma will be to show that such a sequence does indeed exist.

Once we know $I_1, \dots, I_n$, we can recursively compute a configuration with dilation at most σ: Choose an arbitrary bridge $b_n \in I_n$. If bridges $b_{n-1}, b_{n-2}, \dots, b_{i+1}$ have been computed, choose a bridge $b_i \in I_i$ whose distance from b_{i+1} is minimal. Since I_i is an "interval of bridges", this implies that either b_i and b_{i+1} are collinear, or b_i is one of the extreme bridges in I_i. We now prove that this approach is correct.

Lemma 5 *Let $I_1, \dots, I_n$ be given as defined above. A configuration B with dilation $\mathrm{dil}(B) \leq \sigma$ exists if and only if $I_n \neq \emptyset$. If it exists, it can be computed in $O(n)$ time from the intervals.*

Lemma 6 *The intervals $I_1, \dots, I_n$ defined above can be computed in $O(n^2)$ time and $O(n)$ space.*

Lemmas 6 and 5 imply the following theorem.

Theorem 7 *Given a bridge graph $\mathcal{G}$ on a set of $n+1$ buildings that is a path and a real number $\sigma > 1$, we can in time $O(n^2)$ compute a configuration B realizing $\mathcal{G}$ with $\mathrm{dil}(B) \leq \sigma$ or determine that no such configuration exists.*

It seems hard to improve this result when there are $\Theta(n^2)$ visible pairs that could determine the dilation. In fact, we do not even know how to decide in $o(n^2)$ time whether a *given* configuration has dilation $\leq \sigma$. If the number k of visible pairs of the given set of buildings is $o(n^2/\log n)$, the running time can be improved to $O(k \log n)$.

We solve the original optimization problem using Megiddo's parametric search [5].

Theorem 8 *Given a bridge graph on a set of $n+1$ buildings that is a path, we can compute a configuration with the optimal dilation in time $O(n^3 \log n)$, or in time $O(nk \log^2 n)$, where k is the number of visible pairs.*

References

[1] L. P. Chew. There are planar graphs almost as good as the complete graph. *J. Comput. Syst. Sci.*, 39:205–219, 1989.

[2] David Eppstein. Spanning trees and spanners. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 425–461. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[3] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.

[4] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[5] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30(4):852–865, 1983.

[6] D. Peleg and A. Schäffer. Graph spanners. *J. Graph Theory*, 13:99–116, 1989.

Pointed Encompassing Trees

(Extended Abstract)

Michael Hoffmann [a], Bettina Speckmann [b], and Csaba D. Tóth [c]

[a] *Institute of Theoretical Computer Science, ETH Zürich,* hoffmann@inf.ethz.ch.
[b] *Department of Mathematics and Computer Science, TU Eindhoven,* speckman@win.tue.nl.
[c] *Department of Computer Science, University of California at Santa Barbara,* toth@cs.ucsb.edu.

Abstract

It is shown that for any set of disjoint line segments in the plane there exists a *pointed binary encompassing tree*, that is, a spanning tree on the segment endpoints that contains all input segments, has maximal degree three, and such that every vertex is incident to an angle greater than π. As a consequence, it follows that every set of disjoint line segments has a bounded degree pseudo-triangulation.

1. Introduction

Disjoint line segments in the plane are the fundamentals of computational geometry. They form the atomic structure of most planar geometric data structures and geographic information systems. Planar objects are typically represented by a polygonal approximation which, in turn, is composed of (interior) disjoint line segments. Not surprisingly, researchers studied many of their combinatorial properties, such as visibility, compact representation, and ray shooting.

Geometric graphs. We follow one particularly well-studied trail: that of constrained geometric graphs. A *geometric graph* is a graph together with a planar embedding such that the edges are straight line segments. We consider *crossing-free* geometric graphs, that is, we do not allow two edges to cross. Observe that crossing-free is not equivalent to *planar*, since planar graphs may have embeddings in the plane with crossing edges. Given a set of disjoint segments in the plane (that is, a crossing-free geometric matching), we say that a graph is *encompassing* if it is a connected crossing-free geometric graph that contains all input segments as edges (without Steiner points).

It is known that there does not always exists a Hamiltonian circuit (nor path) through a set of disjoint segments. In fact, it is NP-complete to decide if a Hamiltonian circuit exists for a given set of segments, if the segments are allowed to intersect at their endpoints [16]. Rappaport et al. [17] gave a polynomial time algorithm for a set of convexly independent segments. Among n disjoint segments in the plane there are always $\Theta(\log n)$ for which an encompassing path exists [7], this number amounts to $\Theta(\sqrt{n})$ if all segments are axis-parallel [21].

The maximal degree of an encompassing tree on the segment endpoints that is *constrained* to contain all input segments is, therefore, at least three. After a preliminary upper bound of seven by Bose and Toussaint [6], Bose et al. [5] proved that an encompassing tree with maximal degree three always exists. Later Hoffmann and Tóth [8] showed that there is also a *Hamiltonian* encompassing graph with maximum degree three.

Pointedness. Pseudo-triangulations are decompositions of the plane invented by Pocchiola and Vegter [14]. A pseudo-triangulation is a partition of the convex hull of input points or polygonal objects into pseudo-triangles, that is, simple polygons with exactly three vertices whose interior angle is less than or equal to π. They obtained considerable attention recently, as they have found numerous important applications in visibility [13,14], rigidity [20], kinetic colli-

sion detection [1,11], and guarding [19]. Pseudo-triangulations, just like triangulations, are also crossing-free geometric graphs. A characteristic property of *minimal* pseudo-triangulations is that for every vertex p all the incident edges are on one side of a line through p (in other words, every vertex is incident to an angle greater than π). Using Streinu's terminology [20], this property is called *pointedness*.

By a result of Streinu [20], there is an encompassing pointed pseudo-triangulation for any set of disjoint line segments: We obtain a pseudo-triangulation by adding edges greedily while pointedness is maintained. Rote et al. [18] recently studied the size of minimum pseudo-triangulations *constrained* to contain a set of *non-crossing* segments. The pointed (or equivalently, minimum) pseudo-triangulation of n *disjoint* segments always has $4n - 3$ edges and $2n - 2$ faces.

2. Results

On one hand, both the algorithm of Hoffmann and Tóth [8] and that of Bose et al. [5] are doomed to violate pointedness due to their proof techniques. In fact, the algorithm in [8] can output a binary encompassing graph for which no spanning subgraph is pointed. On the other hand, the pointed encompassing tree obtained by the straightforward method of Streinu [20] has no guarantee on the degree of the resulting geometric graph. Here, we show how to construct an encompassing tree that respects pointedness *and* has maximal vertex degree at most three:

Theorem 1 *For any set of disjoint line segments in the plane there exists a pointed binary encompassing tree.*

An application. It is known that the triangulation of a planar point set can have arbitrarily high degree. This is also true for triangulations constrained to contain disjoint line segments. Kettner et al. [10] proved that for any set of points in the plane there is a pseudo-triangulation with maximal degree at most five. Bounded vertex degree is a useful property in most applications, as local operations or kinetic data structures require a constant amount of updates. Recently, Aichholzer et al. [2] showed that a bounded degree pseudo-triangulation constrained to contain a Hamiltonian circuit (a simple polygon) also exists, with a degree bound of ten. We can extend these results to pseudo-triangulations constrained to contain disjoint line segments (that is, a perfect matching).

Theorem 2 *Every set of disjoint line segments has a pointed pseudo-triangulation with maximum vertex degree at most ten.*

The best lower bound we could generate is a set of disjoint segments such that in any pointed pseudo-triangulation there is a vertex of degree at least six.

3. Proof technique

We define a class of weakly simple polygons that we call *pearl polygons*. Every vertex of a pearl polygon has either degree two or degree four. Moreover, for every degree four vertex we mark one incident edge such that deleting all marked edges from the pearl polygon results in a spanning tree. The convex hull of the segments belongs to the class of pearl polygons. We start out from the convex hull, and modify it locally using geodesic curves while maintaining a pearl polygon until certain conditions are satisfied. The proof is completed by applying induction in each face of a suitable convex subdivision of the interior of the pearl polygon.

The general scheme of the induction and the use of geodesic curves are similar to the proof techniques applied in [8]. However, the details are quite different and there are too many to list them within the scope of this abstract. Hence, we have to refer the reader to the full paper at this point.

4. Bounded degree pseudo-triangulations for disjoint segments

A more careful analysis reveals that the following form of Theorem 1 also holds:

Theorem 3 *For any set S of disjoint line segments in the plane there exists a pointed binary encompassing tree such that the maximal degree is at most three, and if a convex hull vertex has degree three then at least one of the incident edges is part of the convex hull.*

We combine this theorem with an algorithm of Aichholzer et al. [2], according to which a simple polygon can be pseudo-triangulated such that the degree of every convex vertex is at most four and the degree of every reflex vertex is at most five, that is, every convex (reflex) vertex has at most two (three) new incident edges in addition to the two incident polygon edges. This result also holds for weakly simple polygons that may have reflex interior angles of 2π.

Let us consider the union of the binary pointed encompassing tree T claimed by Theorem 3 and the convex hull $\mathrm{conv}(S)$ of the segments in S. The tree T partitions the polygonal domain $\mathrm{conv}(S)$ into *weakly* simple polygons. We obtain a bounded degree pseudo-triangulation by pseudo-triangulation each polygon using the algorithm of Aichholzer et al. [2]. Every interior vertex p_{int} has degree three, and one of the incident angular domains is greater than π. So the degree of p_{int} increases by at most $3 + 2 + 2$ to at most 10. Every convex hull vertex p_{hull} has degree at most 4 in $T \cup \mathrm{conv}(S)$ according to Theorem 3, and the one reflex angular domains lies at the exterior of $\mathrm{conv}(S)$ which does not need to be pseudo-triangulated. Therefore the degree of p_{hull} increases by at most $2 + 2 + 2$ to at most 10. This proves Theorem 2.

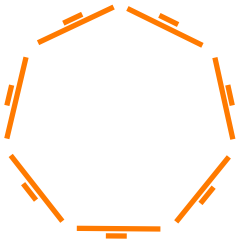


Fig. 1. Lower bound construction.

In the remainder, we describe a set of 14 disjoint segments whose every pseudo-triangulation has a vertex of degree at least 6. The segment endpoints form a regular 28-gon $P = (p_1, p_2, \ldots, p_{28})$. Place seven disjoint segments along the sides $p_{4k+2}p_{4k+3}$, and along parallel the diagonals $p_{4k+1}p_{4k+4}$ for $k = 1, 2, \ldots, 7$, see Fig. 1. The pseudo-triangulation of a convex polygon is also a triangulation. Any triangulation of the inner 14-gon either has a vertex of degree five, or it has three consecutive vertices, each of degree four. Taking the seven outer segments into account adds an additional convex hull edge to each vertex. Moreover, out of any three consecutive vertices of the inner 14-gon one gets another edge, since the outer quadrilaterals have to be (pseudo-)triangulated by any diagonal. Therefore, there is a vertex of degree at least six in P.

References

[1] P. K. Agarwal, J. Basch, L. J. Guibas, J. Hershberger, and L. Zhang, Deformable free space tilings for kinetic collision detection, in: *Proc. 4th WAFR'00*, 2001, Boston, 83–96.

[2] O. Aichholzer, M. Hoffmann, B. Speckmann, and Cs. D. Tóth, Degree bounds for constrained pseudo-triangulations, in: *Proc. 15th Canadian Conf. on Comput. Geom.*, pp. 155–158.

[3] B. Beauquier, S. Pérennes, and D. Tóth, All-to-all routing and coloring in weighted trees of rings, in *Proc. 11th ACM Sympos. Parallel Alg. and Architectures (Saint-Malo, 1999)*, 185–190.

[4] S. Bespamyatnikh, Computing homotopic shortest paths in the plane, in *Proc. 14th ACM-SIAM Symp. Discrete Algorithms (Baltimore, MD, 2003)*, 609–617.

[5] P. Bose, M. E. Houle, and G.T. Toussaint, Every set of disjoint line segments admits a binary tree, *Discrete Comput Geom.* **26** (2001), 387–410.

[6] P. Bose and G. T. Toussaint, Growing a tree from its branches, *J. Algorithms* **19** (1995), 86–103.

[7] M. Hoffmann and Cs. D. Tóth, Alternating paths through disjoint line segments, *Inform. Proc. Letts.* **87** (2003), 287–294.

[8] M. Hoffmann and Cs. D. Tóth, Segment endpoint visibility graphs are Hamiltonian, *Comput. Geom. Theory Appl.* **26** (2003), 47–68.

[9] A. Kaneko, On the maximum degree of bipartite embeddings of trees in the plane, in *Discrete and Computational Geometry*, LNCS, Vol. 1763, Springer, 2000, Berlin, pp. 166–171

[10] L. Kettner, D. Kirkpatrick, A. Mantler, J. Snoeyink, B. Speckmann, and F. Takeuchi, Tight degree bounds for pseudo-triangulations of points, *Comp. Geom. Theory Appl.* **25** (2003), 1–12.

[11] D. Kirkpatrick and B. Speckmann, Kinetic maintenance of context-sensitive hierarchical representations for disjoint polygons, in *Proc. 18th ACM Sympos. Comput. Geom.*, pp. 179–188.

[12] J. Pach and E. Rivera-Campo, On circumscribing polygons for line segments, *Comput. Geom. Theory Appl.* **10** (1998), 121–124.

[13] M. Pocchiola and G. Vegter, Minimal tangent visibility graphs, *Comput. Geom. Theory Appl.* **6** (1996), 303–314.

[14] M. Pocchiola and G. Vegter, Topologically sweeping visibility complexes via pseudo-triangulations, *Discrete Comput. Geom.* **16** (1996), 419–453.

[15] P. Raghavan and E. Upfal. Efficient routing in all-optical networks, in *Proc. 26th ACM Sympos. Theory of Computing (Montréal, QC, 1994)*, pp. 134–143.

[16] D. Rappaport, Computing simple circuits from a set of line segments is NP-complete, *SIAM J. Comput.* **18** (1989), 1128–1139.

[17] D. Rappaport, H. Imai and G. T. Toussaint, Computing simple circuits from a set of line segments, *Discrete Comput. Geom.* **5** (1990), 289–304.

[18] G. Rote, C. A. Wang, L. Wang, and Y. Xu, On constrained minimum pseudotriangulations, in *Proc. 9th Intern. Comp. Comb. Conf.*, LNCS, Vol. 2697, Springer, Berlin, 2003, pp. 445–454.

[19] B. Speckmann and Cs. D. Tóth, Allocating vertex π-guards in simple polygons via pseudo-triangulations, in *Proc. Sympos. on Discrete Alg. (Baltimore, MD, 2003)*, pp. 109–118.

[20] I. Streinu, A combinatorial approach to planar non-colliding robot arm motion planning, in *Proc. 41st Sympos. Found. of Comp. Sci. (Redondo Beach, CA, 2000)*, pp. 443–453.

[21] Cs. D. Tóth, Alternating paths along orthogonal segments, in *Proc. 8th Internat. Workshop Algs. Data Structures (Ottawa, ON, 2003)*, LNCS, Vol. 2748, Springer, Berlin, 2003, pp. 389–400.

[22] M. Urabe and M. Watanabe, On a counterexample to a conjecture of Mirzaian, *Comput. Geom.* **2** (1992), 51–53.

Lower Bounds for the Polygon Exploration Problem

Extended Abstract

Roland Hagius [a] Christian Icking [a] Elmar Langetepe [b]

[a] *FernUniversität Hagen, Praktische Informatik VI, 58084 Hagen, Germany*
[b] *Universität Bonn, Institut für Informatik I, 53117 Bonn, Germany*

Abstract

We improve the best known lower bound for the polygon exploration problem from 1.2071 to 1.2825.

1. Introduction

Exploring an unknown environment is a basic problem for autonomous mobile systems. Here, we suppose that a simple polygon in the plane represents the unknown environment and a point-shaped robot with a vision system, starting at some boundary point s, has the task of going around inside the polygon until the whole environment has been visible at least once before returning to s. Seeing all points inside the polygon is clearly equivalent to seeing its whole boundary, as long as the polygon is simple (does not contain holes, as we assume).

The *shortest watchman tour* starting and ending in s is the shortest tour that can see the whole polygon, see figure watchman. This is the perfect solution in a *known* environment, and it can be computed using the algorithms of Chin and Ntafos or Tan and Hirata [1,3,7,8].

But in an *unknown* environment the shortest watchman tour is also not known, so any tour that explores the polygon *online* is inevitably longer than the shortest watchman tour; there are exceptions for special cases [2]. So it is an interesting question to ask for a strategy that produces exploration tours that are not so much longer than the perfect solution in a known polygon, and Hoffmann et al. [4] have given such a competitive exploration strategy. This strategy guarantees a tour that is at most 26.5 times as long as the shortest watchman tour.

Although this upper bound is most probably not tight, it is not really obvious how to prove a shorter factor by enhancing this rather complicated proof or by giving a better strategy. And the worst case that is known for this strategy has a factor of just 5.

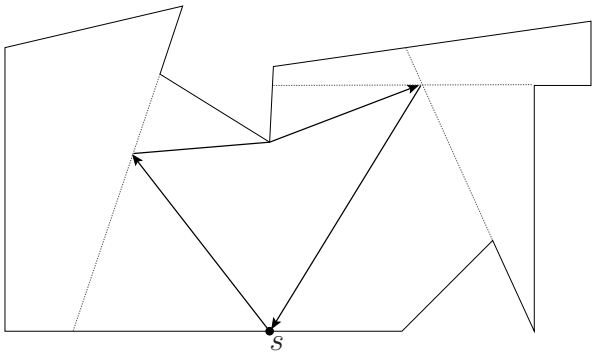


Fig. 1. A shortest watchman tour.

2. First ideas for lower bounds

In this paper we propose to look at this problem "from the other side": what is a lower bound for this problem, i. e., can we show that no strategy can guarantee a competitive factor less than this lower bound? A proof for such a bound can be given by a concrete polygon for which we have to show that an arbitrary strategy will necessarily make a certain detour as compared to the shortest tour. A trivial lower bound is $(\sqrt{2}+1)/2 \approx 1.2071$, see the left picture in Fig. 2: we use an isosceles, rectangular triangle, the start point s is at the right angle and at the other two corners there are two very small

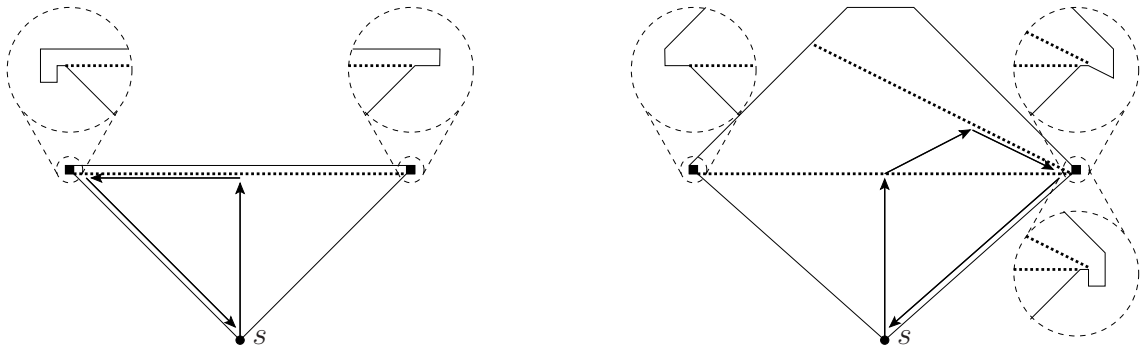


Fig. 2. Deriving the lower bounds 1.2071 (left) and 1.2145 (right).

pockets. One of the pockets is formed such that the corner must be visited while the other is not. The line segment connecting the two corners is called a *threshold*. To trick any strategy to make a detour we can wait until this threshold is reached: if this occurs left of the midpoint then the right corner must be visited, and vice versa. Thus, only when the robot eventually visits the threshold, it can see which corner is still to visit. Now it is easy to see that we have a lower bound of $(\sqrt{2}+1)/2$ here.

In fact we can act as the malicious adversary of a strategy: depending on the decisions of the strategy we decide how the polygon looks like at those parts that have not been seen yet. For example, the previous idea can be refined by introducing a second threshold, see the right picture in Fig. 2. First, we let the strategy reach the line between the two corners as before. Here, we reveal one of the corners, but in contrast to the first example for the second corner two possibilities remain: either we have to visit the corner itself or it suffices to reach the second threshold which is a diagonal line through the corner. After carefully choosing the lengths and the angles (the triangle is still isoceles, but the angle at s is 96.99°) we obtain a bound of 1.2145, which constitutes a small progress.

3. Using more thresholds

But this idea of several thresholds can be driven further. In the following, we sketch our approach with three thresholds. The basic figure is a triangle, but from one of its corners there are edges to an inner point, this will be the starting point s, see Fig. 3. So besides vertex s there are four corners in the scene with a hidden pocket. The pockets may be formed such that the corner must be visited or not, this is the information which is unknown at the start.

Any tour, also the shortest watchman tour, has to visit at least all three thresholds. It is also clear that a reasonable strategy will visit the thresholds in the same order as they appear along the triangle, otherwise an even bigger detour will be generated.

Now on each threshold we place a *critical point*: if the robot visits the threshold to the right of the critical point (as seen from s) then we as the malicious adversary decide that the left corner of the threshold must also be visited. If the visit occurs to the left of the critical point then the threshold is done except for the last one where we decide that in this case the right corner has to be visited.

We as the adversary are free to decide about the exact shape of the triangle, the placement of s, and also the placement of the critical points. This is a challenging optimization exercise with many degrees of freedom and several interwoven levels of optimization, which probably can only be solved numerically.

Due to the lack of space we can only briefly summarize our result obtained with the help of Cabri Geometry [5,6], see Fig. 4. We use an equilateral triangle (but it is not clear if this is the best) and a starting point s as shown in the figure. For each threshold there is one uncertainty, so we have eight cases altogether. For each case we take the length of the shortest watchman tour and compare it to the tour passing through the critical points and the corners that have to be visited in that case. The minimum of these eight ratios which we have determined to be at least 1.2825 is a lower bound for the exploration problem.

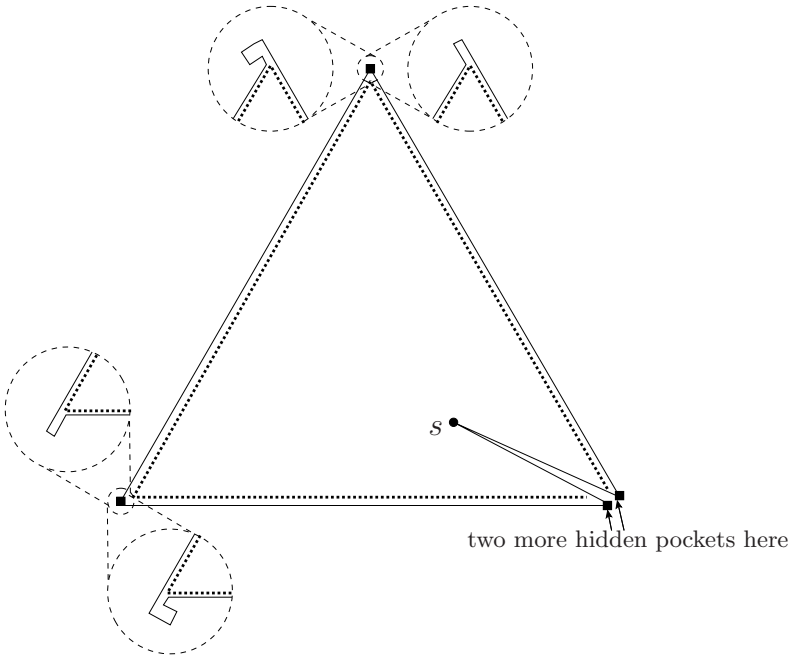


Fig. 3. A triangle-like polygon with three thresholds and four hidden pockets.

4. Conclusions

The new lower bound of 1.2825 for the polygon exploration problem represents a certain progress, but we think that with our technique one can go some further steps in that direction. Since the optimization problem presented here has so many degrees of freedom and consists of several levels that mutually depend on each other, we can not be sure, yet, to have found the global optimum for the three thresholds. Furthermore, it looks promising, but tedious, to introduce four or even more thresholds, but since the number of cases will be about two to the power of the number of thresholds, there is some considerable work to be done. Finally, it is interesting to note that there is still a great gap between the lower bound obtained in this way and the best known upper bound of 26.5.

References

[1] W.-P. Chin and S. Ntafos. Shortest watchman routes in simple polygons. *Discrete Comput. Geom.*, 6(1):9–31, 1991.

[2] X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment I: The rectilinear case. *J. ACM*, 45(2):215–245, 1998.

[3] M. Hammar and B. J. Nilsson. Concerning the time bounds of existing shortest watchman route algorithms. In *Proc. 11th International Symposium on Fundamentals of Computation Theory*, volume 1279 of *Lecture Notes Comput. Sci.*, pages 210–221. Springer-Verlag, 1997.

[4] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel. The polygon exploration problem. *SIAM J. Comput.*, 31:577–600, 2001.

[5] J.-M. Laborde. Some issues raised by the development of implemented dynamic geometry as with Cabri-géomètre. In *Abstracts 15th European Workshop Comput. Geom.*, pages 7–19. INRIA Sophia-Antipolis, 1999.

[6] J.-M. Laborde and F. Bellemain. *Cabri Geometry II.* Texas Instruments, Dallas, 1993.

[7] X. Tan and T. Hirata. Constructing shortest watchman routes by divide-and-conquer. In *Proc. 4th Annu. Internat. Sympos. Algorithms Comput.*, volume 762 of *Lecture Notes Comput. Sci.*, pages 68–77. Springer-Verlag, 1993.

[8] X. Tan, T. Hirata, and Y. Inagaki. Corrigendum to "an incremental algorithm for constructing shortest watchman routes". *Internat. J. Comput. Geom. Appl.*, 9(3):319–323, 1999.

Fig. 4. Eight cases for three thresholds.

Geometric data structures for multihierarchical XML tagging of manuscripts

Jerzy W. Jaromczyk [a,1], Neil Moore [a,1],
[a] *Deparment of Computer Science, University of Kentucky, Lexington, KY, USA*

Abstract

This paper shows an application of computational geometry methods to the preparation of image-based digital library editions. We present a formalism for describing non-hierarachical markup of manuscripts in terms of one- and two-dimensional geometry. This formalism allows us to use geometric data structures and algorithms to process marked-up documents. With this approach we can overcome many well-known and inherently difficult problems with non-hierarchical markup. We present algorithms based on segment trees and range-query structures for performing a number of queries on markup structure. This application of computational geometry data structures to a new domain also provides insights into novel types of geometric operations and queries. Some of these techniques are currently being used by researchers in the Research Computing for Humanities project, which aims to produce electronic editions of selected manuscripts from the British Library.

1. Introduction

From a computational geometry point of view, an old manuscript, a *folio* (a sheet of writing material) and the script (a text on the manuscript page) have interesting features in one, two and three dimensions and all of them are important. Spatial deformations of the folio can be studied for restoration purposes in three dimensions. Illuminations, damages, restorations, and paleographic features of individual letters can be studied as two dimensional objects. The script, a sequence of lines of text, can be viewed as one dimensional as it follows visible or invisible rulings that guide the layout of the text [B94]. In fact, connecting the image view of a manuscript with its transcript is typically the first task. In this paper we will discuss this linear aspect of folia and we will discuss geometric structures that support its tagging.

Extensive tagging or markup is an often excruciating task in the preparation of electronic editions of manuscripts. The tagging process results in a structured description of the document's contents, its features and attributes in a form that can be used to view and effectively query the edition. The XML (eXtensible Markup Language) is a prevailing format used for describing literary and artistic work for Digital Libraries. XML files describe well-hierarchical structures (e.g., book, volume, section, page, line, word and character) of the document and for that reason they can be viewed as rooted trees. Although it seems natural that most documents adhere to such hierarchies, it is often not true in practice. Even worse, this happens in the context of cultural heritage that urgently requires preservation: old and severely damaged manuscripts. An image of a damaged folio from Alfred the Great's Old English translation of Boethius's *Consolation of Philosophy* is demonstrated in Figure 1.

Words can span more than one line, damages or restorations can overlap words and parts of them. A rather simple case is illustrated in Figure 2 where in the convoluted tagging one word spans two lines; as such, it is not well-formed XML.

It has been long recognized that, in spite of its popularity, XML suffers from an inability to encode elements that are not in hierarchical relationships [RMD93]. There are numerous approaches to address this problem called the *concurrent hierarchies* problem; see, for example, [B95]. Mostly these approaches are concerned with tagging tran-

Email addresses: jurek@cs.uky.edu (Jerzy W. Jaromczyk), neil@cs.uky.edu (Neil Moore).

[1] This research was supported in part by NSF ITR grant 0219924

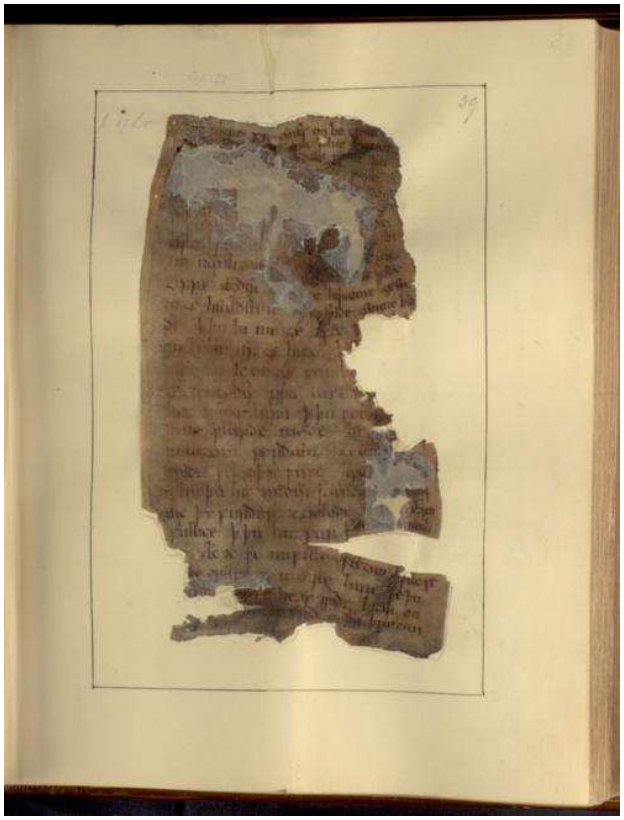


Fig. 1. An image of a damaged manuscript page

```
<line no="2" id="oa6038v02"><res src="Kr">sta</res><uncn
src="mcr">&eth;</uncn>ol<tsp></tsp>f&aelig;st gereaht
&thorn;ur<dmg agent="tear">h &thorn;</dmg>a <dmg
agent="tear">s</dmg><word>tro- </line>

<line no="3" id="oa6038v03">ngan</word> meaht, <abb
type="ampersand">7</abb>  ge<tsp></tsp>ende<tsp></tsp>
<dmg agent="hole">byrd</dmg>, swa swa </line>
```

Fig. 2. Markup that is not hierarchical

scripts (text-based documents). In our task a manuscript or an image of it is the primary source for preparing an electronic edition and we need to handle the concurrent hierarchies in geometric context of the image. This image-based approach is the most distinguishing aspect of our work.

A geometric view of the problem allows us to engage many data structures, in particular multi-dimensional ones. In this paper, we will focus on two of them: segment trees and a grid based data structure developed by Overmars for range queries (we will use it for line segments rather than points, though). We will analyze how these structures support a variety of queries that are essential in the context of editing and studying manuscripts.

There are several contributions of this paper. The first is in applying computational geometry to a new area. We will present a formalism for concurrent hierarchies that allows us to connect geometric structures with XML documents. Specifically, we will discuss a number of algorithms for querying the structure of a document, together with their asymptotic complexities.

2. Definitions

In this section we describe a formalism for representing multihierarchical document markup. This formalism allows us to connect the structure of markup with a geometric representation; this allows us to use geometric data structures and algorithms to process tagged documents.

2.1. *Markup elements*

We represent a markup element as a tuple (N, A, α, ω) where N is the element name, A (a map from strings to strings) the attributes, and α and ω are integers with $1 \leq \alpha \leq \omega$. The interval of a markup element e, written $\mathrm{I}(e)$, is the closed interval $[\alpha(e), \omega(e)] \subset \mathbb{N}$.

DEFINITION **2.1** *Let e_1 and e_2 be two markup elements. We define the relations:*

- $e_1 \prec e_2$ *if* $\omega(e_1) < \alpha(e_2)$
- $e_1 \sqsubset e_2$ *if* $\alpha(e_1) \geq \alpha(e_2)$ *and* $\omega(e_1) \leq \omega(e_2)$ *and* $\mathrm{I}(e_1) \neq \mathrm{I}(e_2)$. *In this case we say that e_1 is a descendant of e_2, or equivalently that e_2 is an ancestor of e_1.*

We state without proof the following theorems:

THEOREM **2.1** *The relations $\prec$ and $\sqsubset$ each form a strict partial order.*

THEOREM **2.2** *Let e_1 and e_2 be two markup elements. Exactly one of the following holds: $e_1 \prec e_2$; $e_2 \prec e_1$; $e_1 \sqsubset e_2$; $e_2 \sqsubset e_1$; e_1 overlaps e_2; or $\mathrm{I}(e_1) = \mathrm{I}(e_2)$.*

3. Hierarchies

DEFINITION **3.1** *Let E be a finite set of markup elements. E is hierarchical if:*

- *There is an element $r \in E$, called the root, such that, for each element $e \in E$, $e \sqsubset r$ or $e = r$*
- *No two elements of E overlap, and no distinct elements of E share the same interval.*

LEMMA **3.1** *Let E be a hierarchical set of markup elements, and $e \in E$. Then either e is the root and has no parents in E, or e is not the root and has exactly one parent in E.*

THEOREM **3.1** *Let E be a hierarchical set of markup elements. Let G be a graph on E such that the edge (e_1, e_2) is in G if e_1 is the parent of e_2. G is a tree.*

This justifies our use of the term "hierarchy". Because children of the same parent cannot be descendants of one another, by Theorem 2.2 they can be ordered by $\prec$. The tree is therefore an ordered tree.

4. Data structures and queries

In this section we describe two geometric data structures and apply them to a number of common queries on documents.

A segment tree [BW80, PS85] is a dynamic data structure to represent a set of segments. For insertions and deletions it is assumed that the endpoints belong to the set of n points known in advance.

The underlying structure is a balanced binary tree with leaves representing atomic segments. Each node corresponds to the union of the atomic segments rooted in this nodes. Intervals that belong to the collection represented in the segment tree are associated with nodes of the tree and satisfy the following property: a node v stores s if the union of its atomic segments is contained in s but the union of the atomic segments associated with the parent of v do not. Thanks to this property each segment is represented in at most $\mathcal{O}(\log n)$ nodes.

Insertions and deletions can be performed in $\mathcal{O}(\log n)$ time. Also, in the same time one can count the number of segments in the collection that includes a given query point.

While segment trees are well-suited for some types of queries, there are other queries which segment trees do not perform as efficiently. We use a range- query structure to support these queries.

We treat a segment $S = [\alpha, \omega]$ in $U = [1, M]$ as a point $p(S) = (\alpha, \omega)$ in U^2. We call this representation of (a collection of) segments a *segment grid*. Many properties of S then correspond to range properties of $p(S)$ in the segment grid. For example, $S \subset T$ if and only if $p(S)$ lies to the lower right of $p(T)$.

We shall make use of general range queries of the form: find all points lying in the (closed) rectangle bounded by (a, b) and (c, d). There exist a number of data structures supporting such queries in a two-dimensional grid. A number of these structures are described in [O88]; two are of particular interest for our purposes.

THEOREM **4.1** *(Overmars) We can represent n points in U^2 (and thus n segments in U) using $\mathcal{O}(n \log n)$ space in such a way that range queries take $\mathcal{O}(k + \log\log|U|)$ time, where k is the number of results returned by the query.*

This data structure makes use of perfect hashing, and is therefore slow to build. There is an alternative data structure with slightly worse query time, but significantly better creation time:

THEOREM **4.2** *(Overmars) We can represent n points in U^2 using $\mathcal{O}(n \log n)$ space in such a way that range queries take $\mathcal{O}\left(k + \sqrt{\log|U|}\right)$ time, where k is the number of returned results. This data structure can be built in $\mathcal{O}(n \log n)$ time.*

Neither of the range-query data structures permits efficient insertion or deletion. For more information on these structures, see [O88].

4.1. *Descendant queries*

A common query on documents is to find all elements of a certain type that are descendants of a given element e. For example, given a `<page>` element, one may wish to find all `<damage>` elements contained within that element, either directly (as children) or indirectly. We can perform this operation by finding all descendants of e and reporting only those of the requested type.

Recall from Definition 2.1 that the descendants of e are those elements which begin no earlier than e, end no later than e, and do not both begin and end at the same point as e. In terms of the segment grid, $p(\mathrm{I}(x))$ is a descendant of e if $\mathrm{I}(x) \neq \mathrm{I}(e)$ and $p(\mathrm{I}(x))$ lies in the rectangle bounded by the points $(\alpha(e), \alpha(e))$ and $(\omega(e), \omega(e))$. Using the data structure from Theorem 4.2, we can find all such points in $\mathcal{O}(k + \sqrt{\log M})$ time, where M is the document's maximum offset and k is the number of results.

4.2. *Overlap queries*

Another useful query is: given an element e, find all elements which overlap e. If e_1 overlaps e_2, e_1 contains at least one of the endpoints of e_2. Conversely, if e_1 contains at least one endpoint of e_2, either e_1 overlaps e_2, $e_1 = e_2$, $e_1 \sqsubset e_2$, or $e_2 \sqsubset e_1$. This suggests the following:

THEOREM **4.3** *Let D be a document containing the element e. Let k be the number of elements overlapping e, d the number of descendants of e, a the number of ancestors of e, and M the maximum off-*

set of D. We can find all elements overlapping e in $\mathcal{O}(k+d+a+\log M)$ time.

We first find the sets $B(e)$ and $E(e)$ of all elements whose intervals contain $\alpha(e)$ and $\omega(e)$, respectively, using stabbing queries in a segment tree. $B(e)$ contains at most $k+d+a+1$ elements, and likewise for $E(e)$. The stabbing queries can therefore be performed in time $\mathcal{O}(k+d+a+\log M)$. We then iterate through the results, reporting those which actually overlap e. Testing whether a given segments overlaps e requires constant time, so this step does not increase the complexity.■

We can improve on this bound somewhat by making use of range queries. If e_1 overlaps e_2, e_1 contains *exactly* one endpoint of e_2. In the segment grid, segments containing $\alpha(e)$ but not $\omega(e)$ lie in the rectangle R_α bounded by the points $(1,\alpha(e))$ and $(\alpha(e),\omega(e))$. Likewise, segments containing $\omega(e)$ but not $\alpha(e)$ lie in the rectangle R_ω bounded by $(\alpha(e),\omega(e))$ and $(\omega(e),M)$, where M is the maximum offset of the document. While these rectangles contain all the elements which overlap e, they do not contain *only* such elements. As with the stabbing queries described above, the range queries also return e, and may return some ancestors and descendants of e, so we must remove these from the result set. It is clear, however, that the rectangles do not contain any element x such that $x \prec e$ or $e \prec x$.

THEOREM **4.4** *Let D be a document containing the element e. Let k be the number of elements overlapping e, d the number of descendants of e, a the number of ancestors of e, and M the maximum offset of D. We can find all elements overlapping e in $\mathcal{O}(k+d+a+\sqrt{\log M})$ time.*

Each range query returns at most $k+d+a+1$ elements, and can therefore be performed in time $\mathcal{O}(k+d+a+\sqrt{\log M})$. As with the stabbing query, we then iterate through the results of the range query, reporting those elements which overlap e; again, this does not affect the overall complexity of the overlap query.■

The running time of overlap queries can be improved still further if we impose additional restrictions on the endpoints of elements:

THEOREM **4.5** *Let D be a document such that no two elements share an endpoint in common, and let e be an element in D. If k is the number of elements overlapping e and M is the maximum offset of D, we can find all elements overlapping e in $\mathcal{O}(k+\sqrt{\log M})$ time.*

5. Conclusion

We have presented a formalism that connects the realm of computational geometry to algorithmic problems that we have faced in the marking up of image-based electronic editions. As examples of geometric structures we have discussed segment trees and range query structures and have applied them to the well-recognized and inherently difficult problem of non-hierarchical markup. Two-dimensional aspects of manuscript marking, such as marginalia, paleographic features and damages will involve additional geometric structures.

Acknowledgement

The authors acknowledge a partial support from the NSF ARCHway: Architecture for Research in Computing for Humanities through Research, Teaching and Learning project. We are thankful to Kevin Kiernan, the Boethius Project and the British Library Board for allowing us to use the image. We also thank our colleagues from the Research Computing for Humanities Lab at the University of Kentucky for many useful discussions.

Bibliography

[B95] Barnard, D., et al. Hierarchical Encoding of Text: Technical problem and SGML Solutions, *Computers and the Humanities* 29/3 (1995) 211-231.

[BW80] Bentley, J. L., and D. Wood. An optimal worst-case algorithm for reporting intersections of rectangles, *IEEE Trans. on Computers* C-29 (1980) 571-577.

[B94] Brown, M. P. *Understanding Illuminated Manuscripts: a Guide to Technical Terms.* London: The British Library, 1994.

[KJDP03] Kiernan, K., J. W. Jaromczyk, A. Dekhtyar, D. C. Porter, et al. The ARCHway Project: Architecture for Research in Computing for Humanities through Research, Teaching, and Learning. To be published in *Literary and Linguistic Computing*, 2003.

[O87] Overmars, Mark H. Efficient data structures for range searching on a grid, *J. Algorithms* 9 (1988) 254-275.

[PS85] Preparata, F. P. and M. I. Shamos. *Computational Geometry: an Introduction.* New York: Springer-Verlag, 1985.

[RMD93] Renear, A., E. Mylonas, and D. Durand. Refining our Notion of What Text Really Is: The Problem of Overlapping Hierarchies. In *Research in Humanities Computing*, eds. N. Ide and S. Hockey. Oxford: Oxford Univeristy Press, 1993.

[S99] Samet, H.. Multidimensional Data Structures. In *Algorithms and Theory of Computation Handbook*, ed. M. J. Atallah. Boca Raton: CRC Press, 1999.

[XML-W3C] Bray, T., J. Paoli, C. M. Sperberg-McQueen, and E. Maler, eds. Extensible Markup Language (XML) 1.0, W3C Recommendation. `http://www.w3.org/TR/2000/REC-xml-20001006`. W3 Consortium, 2000.

Balanced Intervals of Two Sets of Points on a line or circle

Atsushi Kaneko [a] and M. Kano [b]

[a] *Department of Computer Science and Communication Engineering*
Kogakuin University, Nishi-Shinjuku, Shinjuku-ku, Tokyo, 163-8677, Japan
[b] *Department of Computer and Information Sciences*
Ibaraki University, Hitachi, Ibaraki, 316-8511, Japan
kano@cis.ibaraki.ac.jp *http://gorogoro.cis.ibaraki.ac.jp*

Abstract

Let n, m, k, h be positive integers such that $1 \leq n \leq m$, $1 \leq k \leq n$ and $1 \leq h \leq m$. Then we give a necessary and sufficient condition for every configuration with n red points and m blue points on a line or circle to have an interval containing precisely k red points and h blue points.

Key words: balanced interval, interval, two sets of points, line, circle

1. A balanced interval on a line

In this section we shall prove the following theorem.

Theorem 1 *Let n, m, k, h be integers such that $1 \leq n \leq m$, $1 \leq k \leq n$ and $1 \leq h \leq m$. Then for any n red points and m blue points on a line in general position (i.e., no two points lie on the same position.), there exists an interval that contains precisely k red points and h blue points if and only if*

$$\left(\left\lfloor \frac{n}{k+1} \right\rfloor + 1\right)(h-1) < m < \left(\left\lfloor \frac{n}{k-1} \right\rfloor\right)(h+1), \tag{1}$$

where the rightmost term is an infinite number when $k = 1$.

We begin with an example of our theorem. Consider a configuration consisting of 10 red points and 20 blue points on a line in general position. Then by the above theorem, we can easily show that if $k \in \{1, 2, 3, 5, 10\}$, then such a configuration has an interval containing exactly k red points and $2k$ blue points; otherwise (i.e., $k \in \{4, 6, 7, 8, 9\}$) there exist a configuration that has no such an interval (Fig. 1). We call an interval that contains given number of red points and blue points *a balanced interval*.

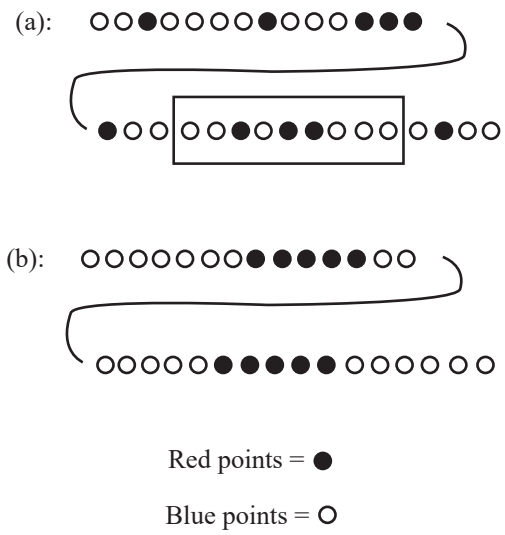


Fig. 1. (a): An interval containing 3 red points and 6 blue points; (b): A configuration that has no interval containing exactly 4 red points and 8 blue points.

Theorem 1 is an easy consequence of the following five lemmas.

For a configuration with red and blue points on a line, we denote by R and B the sets of red points and blue points, respectively. A configuration X with n red points and m blue points on the line is expressed as

$$\{x_1\} \cup \{x_2\} \cup \cdots \cup \{x_{n+m}\},$$

where each x_i denotes a red point or a blue point ordered from left to right. The configuration X is also expressed as

$$R(1) \cup B(1) \cup \cdots \cup R(s) \cup B(s),$$

where $R(i)$ and $B(i)$ denote disjoint subsets of R and B, respectively, and some of them may be empty sets. For a set Y, we denote by $|Y|$ the cardinality of Y.

Lemma 2 *If*

$$m \leq \Big(\Big\lfloor \frac{n}{k+1} \Big\rfloor + 1\Big)(h-1), \tag{2}$$

then there exists a configuration with n red points and m blue points that has no interval containing exactly k red points and h blue points.

PROOF. Let $t = \lfloor \frac{n}{k-1} \rfloor$. Then $n \leq (t+1)(k-1)$, and $m \geq t(h+1)$ by (4). Hence we can obtain the following configuration with n red points and m blue points:

$$R(1) \cup B(1) \cup \cdots \cup R(t+1) \cup B(t+1),$$

where $|R(i)| \leq k-1$ for every $1 \leq i \leq t+1$, $|R(1) \cup \cdots \cup R(t+1)| = n$, $|B(i)| = h+1$ for every $1 \leq i \leq t$, $|B(t+1)| = m-(h+1)t \geq 0$ and $|B(1) \cup \cdots \cup B(t+1)| = m$. Then this configuration obviously has no interval containing exactly k red points and h blue points since every interval containing k red points must include $B(j)$ for some $1 \leq j \leq t$.

Lemma 3 *If*

$$m > \Big(\Big\lfloor \frac{n}{k+1} \Big\rfloor + 1\Big)(h-1), \tag{3}$$

then every configuration with n red points and m blue points on a line has an interval containing exactly k red points and at least h blue points.

PROOF. Let $t = \lfloor \frac{n}{k+1} \rfloor$. Let X be a configuration with n red points and m blue points. Suppose that X has no desired interval. Namely, we assume that every interval containing exactly k red points has at most $h-1$ blue points.

Let $r_1, r_2, \cdots, r_n$ be the red points of X ordered from left to right. For integers $1 \leq i < j \leq n$, let $I(i,j)$ denote an open interval (r_i, r_j), and let $B(i,j)$ denote the set of blue points contained in $I(i,j)$. Furthermore, $B(-\infty, i)$ denotes the set of blue points contained in the open interval $(-\infty, r_i)$, and $B(i, \infty)$ is defined analogously. Then for any integer $1 \leq s \leq t-1$, $I(s(k+1), (s+1)(k+1))$ contains exactly k red points $\{r_j \mid s(k+1)+1 \leq j \leq (s+1)(k+1)-1)\}$, and thus $|B(s(k+1), (s+1)(k+1))| \leq h-1$ by our assumption. Similarly, an open interval $(-\infty, r_{k+1})$ contains exactly k red points, and thus $|B(-\infty, k+1)| \leq h-1$. Moreover, since $n < (t+1)(k+1)$, $I(t(k+1), \infty)$ has at most k red points, and thus $B(t(k+1), \infty) \leq h-1$. Therefore

$$\begin{aligned} |B| &\leq |B(-\infty, k+1) \cup B(k+1, 2(k+1)) \cup \cdots \\ &\quad \cup B(t(k+1), \infty)| \\ &\leq (t+1)(h-1). \end{aligned}$$

This contradicts (3). Consequently the lemma is proved.

Lemma 4 *If $2 \leq k$ and*

$$m \geq \Big\lfloor \frac{n}{k-1} \Big\rfloor (h+1), \tag{4}$$

then there exists a configuration with n red points and m blue points on a line that has no interval containing exactly k red points and h blue points.

PROOF. Let $t = \lfloor \frac{n}{k-1} \rfloor$. Then $n \leq (t+1)(k-1)$, and $m \geq t(h+1)$ by (4). Hence we can obtain the following configuration with n red points and m blue points:

$$R(1) \cup B(1) \cup \cdots \cup R(t+1) \cup B(t+1),$$

where $|R(i)| \leq k-1$ for every $1 \leq i \leq t+1$, $|R(1) \cup \cdots \cup R(t+1)| = n$, $|B(i)| = h+1$ for every $1 \leq i \leq t$, $|B(t+1)| = m-(h+1)t \geq 0$ and $|B(1) \cup \cdots \cup B(t+1)| = m$. Then this configuration obviously has no interval containing exactly k red points and h blue points since every interval containing k red points must include $B(j)$ for some $1 \leq j \leq t$.

Lemma 5 *If $2 \leq k$ and*

$$m < \Big\lfloor \frac{n}{k-1} \Big\rfloor (h+1), \tag{5}$$

then every configuration with n red points and m blue points on a line has an interval containing exactly k red points and at most h blue points.

PROOF. Let $t = \lfloor \frac{n}{k-1} \rfloor$. Let X be a configuration with n red points and m blue points. Suppose that X has no desired interval. Namely, we assume

that every interval containing exactly k red points has at least $h+1$ blue points.

Let $r_1, r_2, \cdots, r_n$ be the red points of X ordered from left to right. For integers $1 \leq i < j \leq n$, let $I[i,j]$, denote a closed interval $[r_i, r_j]$, and let $B'(i,j)$ denote the set of blue points contained in $I[i,j]$.

Then for any integer $0 \leq s \leq t-2$, $I[k+s(k-1), k+(s+1)(k-1)]$ contains exatly k red points $\{r_j \mid k+s(k-1) \leq j \leq k+(s+1)(k-1))\}$, and thus $|B'(k+s(k-1), k+(s+1)(k-1))| \geq h+1$ by our assumption. Similarly, we have $|B'(1,k)| \geq h+1$. Therefore

$$\begin{aligned}|B| &\geq |B'(1,k) \cup B'(k,k+(k-1)) \cup \cdots \\ &\quad \cup B'(k+(t-2)(k-1), k+(t-1)(k-1))| \\ &\geq t(h+1).\end{aligned}$$

This contradicts (5). Consequently the lemma is proved.

Lemma 6 *Consider a configuration with n red points and m blue points on a line. Suppose that there exists two intervals I and J such that both I and J contain exactly k red points respectively, I contains at most h blue points, and that J contains at least h blue points. Then there exists an interval that contains exactly k red points and h blue points.*

PROOF. If the sets of red points contained in I and J, respectively, are the same, then the lemma immediately follows. Thus we may assume that $I \cap R \neq J \cap R$, where R denote the set of n red points. Without loss of generality, we may assume that the leftmost red point of I lies to the left of J.

We shall show that we can move I to J step by step in such a way that the number of red points is a constant k and the number of blue points changes ± 1 at each step. We first remove the blue points left to the leftmost red point of I one by one, and then add the consecutive blue points lying to the right of I one by one, and denote the resulting interval by I_1 (Fig. 3). We next simultaneously remove the leftmost red point of I_1 and add the red point lying to the right of I_1, and get an interval I_2, which also contains exactly k red points and whose blue points are the same as those in I_1 (Fig. 3). By repeating this procedure, we can get an interval whose red point set is equal to that of J. Therefore, we can move I to J in the desired way. Consequently, we can find the required interval, which contains exactly k red points and h blue points.

2. A balanced interval on a circle

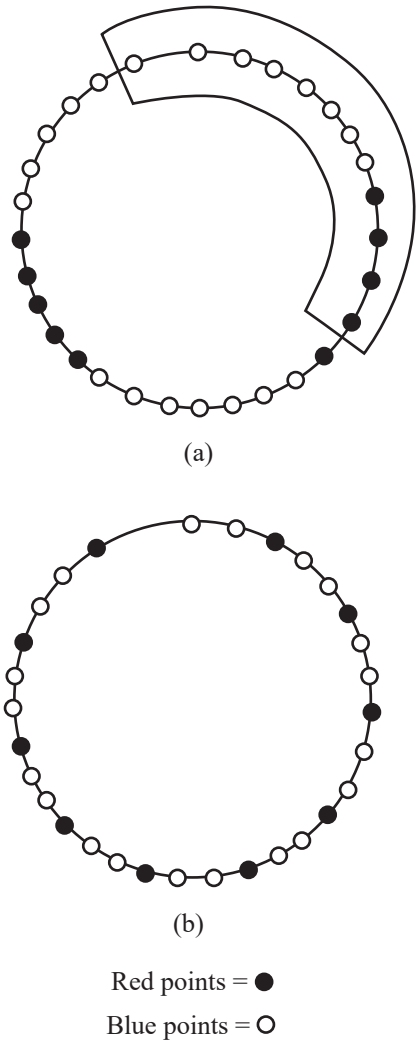


Fig. 2. (a): An interval containing 4 red points and 8 blue points; (b): A configuration that has no interval containing exactly 4 red points and 5 blue points.

In this section, we consider the following theorem, and give its example in Figure reffig:2

Theorem 7 *Let n, m, k, h be integers such that $1 \leq n \leq m$, $1 \leq k \leq n$ and $1 \leq h \leq m$. Then for any n red points and m blue points on a circle in general position (i.e., no two points lie on the same position.), there exists an interval that con-*

tains precisely k red points and h blue points if and only if

$$\frac{n}{k+1}(h-1) < m < \frac{n}{k-1}(h+1), \qquad (6)$$

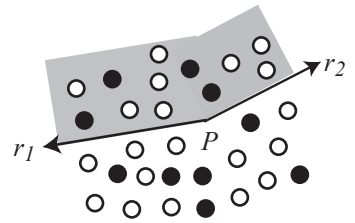
where the rightmost term is an infinite number when $k = 1$.

Theorem 8 can be proved by showing similar lemmas as in the case of line. We conclude the paper by the next conjecture.

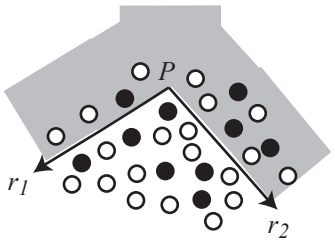
Conjecture 8 *Let* n, m, k, h *be integers such that* $1 \leq n \leq m$, $1 \leq k \leq n$ *and* $1 \leq h \leq m$. *Then for any n red points and m blue points in the plane in general position (i.e., no three points lie on the same.), there exists a wedge that contains precisely k red points and h blue points if and only if*

$$\frac{n}{k+2}(h-1) < m < \frac{n}{k-2}(h+1), \qquad (7)$$

where the rightmost term is an infinite number when $k = 1$.



(a)



(b)

Fig. 3. Wedges containing 4 red points and 8 blue points.

References

[1] Bárány, I., and Matoušek, J.; Simultaneous partitions of measures by k-fans, *Discrete Comput. Geom.* **25** (2001)317–334.

[2] Goodman, J. and O'Rourke, J.; *Handbook of Discrete and Computational Geometry*, CRC Press, (1997) .

[3] Kaneko, A. and Kano, M.; Discrete geometry on red and blue points in the plane — A survey, *Discrete and Computational Geometry*!&*The Goodman-Pollack Festschrift*!&*With contributions by numerous experts* (Springer) (2003) 551-570.

Similaritiy Search in Semialgebraic Pattern Spaces

(Extended Abstract)

Christian Knauer [a]

[a] *Institut für Informatik, Freie Universität Berlin, Takustraße 9, D-14195 Berlin, Germany*

Abstract

We describe a general technique to construct data structures for similarity search in semialgebraic pattern spaces. These spaces capture most known combinations of geometric patterns (e.g., point sets, polygons, polygonal curves) and geometric distance measures for them (e.g. Hausdorff-distance, area of overlap, Fréchet-distance) together with their quotients under various transformation classes (e.g., translations, rigid motions) and they provide the first non-trivial exact search structures in these settings.

Key words: Computational geometry, Shape matching, Similarity queries, Data structures

1. Introduction

Similarity search is a much-studied and practically important type of problem: *Given a set $\mathcal{D}$ of n patterns from a suitable class of valid geometric patterns (e.g., polygonal curves), preprocess them in such a way that we can determine quickly for a query pattern Q, which of the n preprocessed patterns is most similar to the query object (this is called a* similarity query*). We assume that we have an appropriate distance measure δ (e.g., the smallest Fréchet distance that can be achieved under translations) to asses the similarity of two patterns.*

Satisfactory algorithmic results exist only in the case that the patterns can be encoded in a Euclidean space, or an L_1- or L_∞-space such that δ is the corresponding metric [9], or in the case that they can be embedded in such spaces with a low distortion [7,6]. Then we have available the very powerful techniques of Voronoi decompositions. Some algorithms were formulated with 'vantage points' or similar devices [2,3,5,10], but then there are no general performance bounds: only under additional assumptions that enforce in some way that the distance measure is a metric that is 'similar' to a Euclidean metric it is possible to obtain nontrivial bound on the performance of these algorithms [9].

In this paper, we study the case where the size of the individual patterns is small, compared to n. To be more precise, we assume that $|I| = O(1)$ for all $I \in \mathcal{D}$. In that case the distances $\delta(Q, I)$, can be computed in $O(1)$ time (for reasonable δ). So the query can be answered in $O(n)$ time without additional storage and preprocessing, but up to now there are no algorithms and data structures that allow such queries with a nontrivial query time among *preprocessed* pattern sets. We describe a general technique to construct data structures for similarity queries for many combinations of geometric patterns (e.g., point sets, polygons, polygonal curves) and geometric distance measures (e.g. Hausdorff-distance, area of overlap, Fréchet-distance) together with their quotients under various transformation classes (e.g., translations, rigid motions). The solution achieves sublinear query time with quadratic preprocessing time and storage and provides the first non-trivial search structures in these settings.

A *pattern space* $\Pi = (\Omega, \delta)$ of dimension d is a set of geometric objects Ω, where each object can be described by exactly d real parameters, together with a distance measure δ that maps pairs of objects to non-negative real numbers. Usually

Email address: Christian.Knauer@inf.fu-berlin.de (Christian Knauer).

we identify Ω with $\mathbb{R}^d$, the *parameter space* of Π. The objects in Ω are called *patterns*. Intuitively, when $\delta(P, I)$ is small, we consider the patterns P and I to be similar. Note that we do not demand that δ has some special properties (like being a metric, etc.). As an example consider polygonal chains in the plane with at most r vertices and the Fréchet-distance as a distance measure. Each such chain can be encoded by a sequence of at most $2r$ real numbers, the coordinates of the vertices of the chain. If a chain has less than r vertices we can simply pad this description by repeating the last vertex; note that this padding does not interfere with the distance function.

A pattern space $\Pi = (\mathbb{R}^d, \delta)$ is called *semialgebraic*, if the set $\mathcal{F}_\Pi := \{(P, I, \epsilon) \in \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R} \mid \delta(P, I) \leq \epsilon\}$ is semialgebraic, i.e., there is a boolean formula B in s boolean variables $z_1, \ldots, z_s$, and there are s polynomials $g_1, \ldots, g_s$ in $(2d + 1)$ real variables $\boldsymbol{x}_P, \boldsymbol{x}_I, x_\epsilon$ ($\boldsymbol{x}_P$ and $\boldsymbol{x}_I$ are actually sequences of d variables each), such that

$$\begin{aligned}\delta(P, I) \leq \epsilon \iff \\ B(z_1 \leftarrow [g_1(\boldsymbol{x}_P \leftarrow P, \boldsymbol{x}_I \leftarrow I, x_\epsilon \leftarrow \epsilon) \geq 0], \\ \cdots \\ z_s \leftarrow [g_s(\boldsymbol{x}_P \leftarrow P, \boldsymbol{x}_I \leftarrow I, x_\epsilon \leftarrow \epsilon) \geq 0])\end{aligned}$$

is true

($\leftarrow$ denotes variable substitution and $[X]$ is the truth value of the predicate X).

As an example consider the set of all point sets in the plane with at most r points each and the directed Hausdorff-distance as a distance measure. Recall that for compact sets $P, I \subseteq \mathbb{R}^d$ the *directed Hausdorff distance from P to I*, is defined as $h(P, I) := \max_{x \in P} \min_{y \in I} ||x - y||$.

This is a pattern space of dimension $2r$, since each pattern P can be encoded by a sequence of at most $2r$ real numbers, the coordinates of the points in P. If P has less than r points we can simply pad this description by repeating some point; note that this padding does not interfere with the distance function. To see that it is actually a semialgebraic pattern space, note that for two finite point sets P, I and $\epsilon > 0$ we have that

$$\begin{aligned}h(P, I) \leq \epsilon &\iff \forall p \in P \exists i \in I : ||p - i||^2 - \epsilon^2 \leq 0 \\ &\iff \bigwedge_{p \in P} \exists i \in I : ||p - i||^2 - \epsilon^2 \leq 0 \\ &\iff \bigwedge_{p \in P} \bigvee_{i \in I} ||p - i||^2 - \epsilon^2 \leq 0.\end{aligned}$$

We will see in Section 3 that the notion of a semialgebraic pattern space captures most known combinations of geometric patterns (e.g., point sets, polygons, polygonal curves) and geometric distance measures (e.g. Hausdorff-distance, area of overlap, Fréchet-distance) together with their quotients under various transformation classes.

In this paper, we study the problem of similarity search among patterns from a semialgebraic pattern space where the size of the individual patterns is small, compared to their number: *Given a data set $\mathcal{D}$ that consists of n patterns from a semialgebraic pattern space $\Pi = (\Omega, \delta)$ of dimension d, where* d *is* constant*, preprocess $\mathcal{D}$ into a data structure to answer the following kind of* similarity queries*: For a query pattern $Q \in \Omega$, determine*

- $\Delta(Q, \mathcal{D}) := \text{minarg}_{I \in \mathcal{D}}\, \delta(Q, I)$*, the set of patterns in $\mathcal{D}$ to which Q has the smallest possible distance, and*
- $\Delta(\mathcal{D}, Q) := \text{minarg}_{I \in \mathcal{D}}\, \delta(I, Q)$*, the set of patterns in $\mathcal{D}$ that have the smallest possible distance to Q.*

Since d is constant, the distances $\delta(Q, I)$, can be computed in $O(1)$ time (for reasonable δ). So these queries can be answered in $O(n + k)$ time (where k is the size of the answer) without additional storage and preprocessing, but up to now there are no algorithms and data structures that allow such queries with a nontrivial query time among *preprocessed* pattern sets.

We will also consider the decision version of the similarity queries, called *ϵ-similarity queries*, where we are given an additional parameter $\epsilon > 0$, and we want to determine

- $\Delta(Q, \mathcal{D}, \epsilon) := \{I \in \mathcal{D} \mid \delta(Q, I) \leq \epsilon\}$, the set of patterns in $\mathcal{D}$ to which Q has distance at most ϵ, and
- $\Delta(\mathcal{D}, Q, \epsilon) := \{I \in \mathcal{D} \mid \delta(I, Q) \leq \epsilon\}$, the set of patterns in $\mathcal{D}$ that have distance at most ϵ to Q.

Again the brute-force approach can answer these queries in $O(n + k)$ time, but there are no data structures that allow such queries with a nontrivial query time among preprocessed pattern sets.

2. Similaritiy queries in pattern spaces

In this abstract we only consider the decision version of the similarity queries. We describe a data structure that answers $\Delta(Q, \mathcal{D}, \epsilon)$-queries (the case

of $\Delta(\mathcal{D}, Q, \epsilon)$-queries is completely symmetric). The same techniques apply to similarity queries as well and yield similar results. Our main result is the following

Theorem 1 *Suppose we are given a set $\mathcal{D}$ that consists of n patterns from a semialgebraic pattern space Π of dimension $d = O(1)$. Then we can build in $O(n^2)$ time a data structure of size $O(n^2)$ that answers ϵ-similarity queries in $O(n^{1-1/(2d-3)} + k)$ time, where k is the size of the answer.*

PROOF. The construction works in two steps. We first describe a data structure that can be built in $O(n^{2d-2})$ time (and requires the same amount of space) and can answer a query in $O(\log n + k)$ time, where k is the size of the answer. Then we use a simple partitioning approach to yield the desired result (in fact we prove a somewhat stronger tradeoff that implies the Theorem).

Since $\Pi = (\Omega, \delta)$ is a semialgebraic pattern space of dimension $d = O(1)$, there is a boolean formula B in $s = O(1)$ boolean variables $z_1, \ldots, z_s$, and there are s polynomials $g_1, \ldots, g_s$ in $(2d+1)$ real variables $\boldsymbol{x}_P, \boldsymbol{x}_I, x_\epsilon$, such that

$$\begin{aligned}\delta(Q, I) \leq \epsilon \iff \\ B(z_1 \leftarrow [g_1(\boldsymbol{x}_Q \leftarrow Q, \boldsymbol{x}_I \leftarrow I, x_\epsilon \leftarrow \epsilon) \geq 0], \\ \ldots \\ z_t \leftarrow [g_t(\boldsymbol{x}_Q \leftarrow Q, \boldsymbol{x}_I \leftarrow I, x_\epsilon \leftarrow \epsilon) \geq 0]).\end{aligned}$$

For $j = 1, \ldots, s$ and for all $I \in \mathcal{D}$, we compute the $(d+1)$-variate polynomials

$$g_{j,I}(\boldsymbol{x}_Q, x_\epsilon) := g_j(\boldsymbol{x}_Q, \boldsymbol{x}_I \leftarrow I, x_\epsilon).$$

This is done by substituting I for $\boldsymbol{x}_I$ in g_j and therefore takes time $O(1)$. The total time to compute all these polynomials is $O(n)$.

For the $g_{j,I}$ we compute a subdivision Ξ of $\mathbb{R}^{d+1}$ with the property that the sign of each $g_{j,I}$ remains constant on each cell of the subdivision. Such a subdivsion of size $O(n^{2d-3})$ can be computed in $O(n^{2d-3})$ time, along with a point-location data-structure $\mathcal{L}(\Xi)$ for the subdivision with $O(\log n)$ query-time, c.f., [8]; the computation also yields for each cell $\chi \in \Xi$ a point $(Q_\chi, \epsilon_\chi) \in \chi$.

In a next step we process each cell $\chi \in \Xi$ in turn and compute a set $\mathcal{D}_\chi \subset \mathcal{D}$, which is initially empty. For all $I \in \mathcal{D}$ we do the following: First compute for $j = 1, \ldots, s$ the numbers

$$\gamma_{i,I,\chi} := g_{j,I}(\boldsymbol{x}_Q \leftarrow Q_\chi, x_\epsilon \leftarrow \epsilon_\chi).$$

Next, we compute the truth-value

$$B_{I,\chi} := B(z_1 \leftarrow [\gamma_{1,I,\chi} \geq 0], \ldots, z_s \leftarrow [\gamma_{s,I,\chi} \geq 0]).$$

If $B_{I,\chi}$ is true, we have that $\delta(Q_\chi, I) \leq \epsilon_\chi$ and we add I to $\mathcal{D}_\chi$. We augment the data-structure $\mathcal{L}(\Xi)$ by storing the set $\mathcal{D}_\chi$ for each cell $\chi \in \Xi$. The total time needed to compute it is $O(n^{2d-2})$, and it needs $O(n^{2d-2})$ space.

To answer a query $(Q, \epsilon) \in \Omega \times \mathbb{R}$, we proceed as follows: Using $\mathcal{L}(\Xi)$ we locate the cell $\chi \in \Xi$ with $(Q, \epsilon) \in \chi$ in $O(\log n)$ time. Since the sign of the $g_{j,I}$'s is constant on each cell of Ξ, we have that $\delta(Q_\chi, I) \leq \epsilon_\chi$ iff $\delta(Q, I) \leq \epsilon$, so we can report $\mathcal{D}_\chi$ as the answer to the query. The total time required is $O(\log n + k)$, where $k = |\mathcal{D}_\chi|$ is the size of the answer.

Now we use a simple partitioning approach to yield the desired result: We split $\mathcal{D}$ into $g = \Theta(n/m)$ groups $\mathcal{D}_1, \ldots, \mathcal{D}_g$, each of size $\Theta(m)$, where $1 \leq m \leq n$ is a suitable parameter (see below). Then we build the aforementioned data-structure for each $\mathcal{D}_i$ separately. To answer a query $(Q, \epsilon) \in \Omega \times \mathbb{R}$, we query each data-structure separately and combine the individual answers. The total time needed to compute all the structures is $O(gm^{2d-2}) = O(nm^{2d-3})$ (this is also the total space requirement), and the query time is $O(g \log n) = O((n/m) \log n)$. Setting $m = n^{1/(2d-3)}$ proves the claimed result. □

3. More semialgebraic pattern spaces

In the following we show that the notion of a semialgebraic pattern space captures many combinations of geometric patterns (e.g., point sets, polygons, polygonal curves) and geometric distance measures (e.g., Hausdorff-distance, area of overlap, Fréchet-distance) together with their quotients under various transformation classes (e.g., translations, rigid motions).

Since the set accepted by an algebraic decision tree is semialgebraic we get the following

Lemma 2 *Let Π be a pattern space. If $\mathcal{F}_\Pi$ can be decided by an algorithm in the algebraic decision tree model, then Π is semialgebraic.*

It is straightforward to verify that many of the algorithms for deciding the most prominent distance measures can be implemented in the algebraic decision tree model, c.f., [1]. This shows for example that polygonal curves on k vertices in $\mathbb{R}^d$

wrt. the Fréchet-distance constitute a semialgebraic pattern space.

Let $\Pi = (\mathbb{R}^d, \delta)$ be a semialgebraic pattern space, and let $t : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^f \to \mathbb{R}^d \times \mathbb{R}^d$ be a function. This function induces a new distance measure and thus a new pattern space $\Pi_t := (\mathbb{R}^d, \delta_t)$ as follows:

$$\delta_t(P, I) := \min_{v \in \mathbb{R}^f} \delta(t(P, I, v)).$$

We call δ_t the *quotient of* δ *under* t and f the *number of degrees of freedom of* t.

As an example consider again the pattern space of all point sets in the plane with at most r points each and the directed Hausdorff-distance $h(,)$ as a distance measure. The function $t(P, I, v) := (P + v, I)$ has two degrees of freedom and we have that $h_t(P, I) = \min_{v \in \mathbb{R}^2} h(P + v, I)$ is the smallest directed Hausdorff distance that a *translate* of P has to I.

Theorem 3 *If t is rational, then Π_t is semialgebraic. In that case a semialgebraic description of $\mathcal{F}_{\Pi_t}$ can effectilvey be computed from such a description of $\mathcal{F}_\Pi$.*

PROOF. First, observe that

$$\delta_t(Q, I) \leq \epsilon \iff \exists v \in \mathbb{R}^f : \delta(t(Q, I, v)) \leq \epsilon.$$

Since $\Pi = (\Omega, \delta)$ is a semialgebraic pattern space and t is rational, there is a boolean formula B in s boolean variables $z_1, \ldots, z_s$, and there are s polynomials $g_1, \ldots, g_s$ in $(2d + 1)$ real variables $\boldsymbol{x}_P, \boldsymbol{x}_I, x_\epsilon$, such that

$$\begin{aligned}
&\delta_t(Q, I) \leq \epsilon \iff \\
&\quad \exists v B(z_1 \leftarrow [g_1(t(\boldsymbol{x}_Q \leftarrow Q, \boldsymbol{x}_I \leftarrow I, \boldsymbol{x}_v \leftarrow v), \\
&\qquad\qquad x_\epsilon \leftarrow \epsilon) \geq 0], \\
&\qquad\qquad \ldots \\
&\qquad z_s \leftarrow [g_s(t(\boldsymbol{x}_Q \leftarrow Q, \boldsymbol{x}_I \leftarrow I, \boldsymbol{x}_v \leftarrow v), \\
&\qquad\qquad x_\epsilon \leftarrow \epsilon) \geq 0]
\end{aligned}$$

($\boldsymbol{x}_v$ is a sequence of f new variables).

In general $g_i(t(),)$ is not a polynomial. However, since t is rational, the conditions '$g_i(t(),) \geq 0$' can be rewritten as equivalent *polynomial* inequalities. This shows that $\mathcal{F}_{\Pi_t}$ is a Tarski-set, and therefore semialgebraic. Using standard quantifier elimination techniques [4], a semialgebraic description of $\mathcal{F}_{\Pi_t}$ can effectilvey be computed. □

If the dimension d of Π is $O(1)$ (relative to $n = |\mathcal{D}|$), then the dimension d' of Π_t is also $O(1)$ (unfortunately, in general, d' is doubly exponential in d) and a semialgebraic description of $\mathcal{F}_{\Pi_t}$ can be computed in $O(1)$ time.

This shows for example that polygonal curves on k vertices in $\mathbb{R}^d$ wrt. the smallest Fréchet-distance that can be attained under rigid motions constitute a semialgebraic pattern space of dimension $O(1)$ if $k, d = O(1)$.

Acknowledgments. The author would like to thank Peter Braß for fruitful discussion on the subject.

References

[1] H. Alt and L. J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 121–153. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[2] S. Brin. Near neighbor search in large metric spaces. In *The VLDB Journal*, pages 574–584, 1995.

[3] K. L. Clarkson. Nearest neighbor queries in metric spaces. In *Proc. 29th Annu. ACM Sympos. Theory Comput.*, pages 609–617, 1997.

[4] G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proc. 2nd GI Conference on Automata Theory and Formal Languages*, volume 33 of *Lecture Notes Comput. Sci.*, pages 134–183. Springer-Verlag, 1975.

[5] E.Chávez, G. Navarro, R.Baeza-Yates, and J.Marroquín. Searching in metric spaces. Technical Report TR/DCC-99-3, Dept. of Computer Science, Univ. of Chile, 1999.

[6] P. Indyk. Approximate nearest neighbor algorithms for frechet distance via product metrics. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 102–106. ACM Press, 2002.

[7] P. Indyk and M. Farach-Colton. Approximate nearest neighbor algorithms for Hausdorff metrics via embeddings. In *Proc. 40th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 171–180, 1999.

[8] V. Koltun. Almost tight upper bounds for vertical decompositions in four dimensions. In *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.*, 2001.

[9] S. A. Nene and S. K. Nayar. A simple algorithm for nearest neighbor search in high dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:989–1003, 1997.

[10] P. N. Yianilos. Excluded middle vantage point forests for nearest neighbor search. Technical report, NEC Research Institute, 1999.

Triangulations Without Pointed Spanning Trees Extended Abstract [1]

Oswin Aichholzer [a], Clemens Huemer [a], and Hannes Krasser [b,2]

[a] Institute for Softwaretechnology
Graz University of Technology
Graz, Austria

[b] Institute for Theoretical Computer Science
Graz University of Technology
Graz, Austria

Abstract

Problem 50 in the Open Problems Project [2] asks whether any triangulation on a point set in the plane contains a pointed spanning tree as a subgraph. We provide a counterexample. As a consequence we show that there exist triangulations which require a linear number of edge flips to become Hamiltonian.

Key words: triangulation, spanning tree, pointed pseudo-triangulation, Hamiltonian cycle, edge flip

1. Introduction

Let S be a finite set of points in the plane in general position (no three points are on a common line), and let G be a straight-line graph (drawing in the plane) with vertex set S and edges E. A point $p \in S$ is *pointed* in G if there exists an angle less than π that contains all edges incident to p in G. The graph G is pointed if all its vertices are pointed.

A *triangulation* of S is a maximal planar straight-line graph on top of S. A *spanning tree* on S is a connected, acyclic graph with vertex set S. Several interesting relations between triangulations and spanning trees exist. For example it is well known that the Delaunay triangulation of S contains a minimum spanning tree of S as a subgraph. Another example is a result of Schnyder [3] who shows that every triangulation of a point set with three extreme vertices allows a partition of its interior edges into three trees.

In this note we disprove the following conjecture which was posed as Problem 10 at the First Gremo Workshop on Open Problems (Stels, Switzerland) in July 2003 (by Bettina Speckmann) and at the CCCG 2003 open-problem session (Halifax, Canada) in August 2003. It later on became Problem 50 in the Open Problems Project of the computational geometry community [2].

Conjecture 1 *Every triangulation of a set of points in the plane (in general position) contains a pointed spanning tree as a subgraph.*

This conjecture arose while proving substructure properties when investigating flips in pointed and non-pointed pseudo-triangulations [1]. *Pseudo-triangulations* are a generalization of triangulations. A *pseudo-triangle* is a planar polygon with exactly three interior angles less than π. A pseudo-triangulation of S is a partition of the convex hull of S into pseudo-triangles whose vertex set is S. Pseudo-triangulations have become a versatile data structure. Beside several applications

Email addresses: oaich@ist.tugraz.at (Oswin Aichholzer), chuemer@ist.tugraz.at (Clemens Huemer), hkrasser@igi.tugraz.at (Hannes Krasser).

[1] Research supported by Acciones Integradas 2003-2004, Proj.Nr.1/2003

[2] Research supported by FWF (Austrian Fonds zur Förderung der Wissenschaftlichen Forschung).

in computational geometry, the rich combinatorial properties of pseudo-triangulations stimulated research, see e.g. [1] and references therein.

Obviously Conjecture 1 would be true if a triangulation always contained a Hamiltonian path or a pointed pseudo-triangulation as a subgraph. Several triangulations not containing these structures are known, but for each example it is still easy to find a pointed spanning tree as a subgraph. This observation supported the general belief that the conjecture should be true. However, in the next section we provide a (non-trivial) counterexample. In Section 3 we discuss some implications of this result, like a lower bound for the number of necessary edge flips to transform a given triangulation such that it contains a Hamiltonian cycle.

2. A Counterexample

Figure 1(a) shows the simplest example of a connected straight-line graph not containing a pointed spanning tree as a subgraph. We call this graph a *3-star* and it is a spanning tree which is not pointed at it's central point.

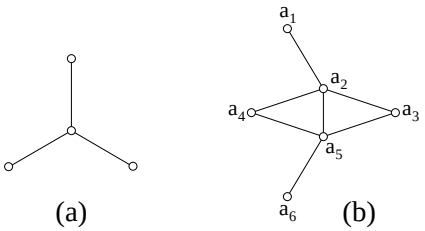


Fig. 1. Small connected straight-line graphs that do not contain a pointed spanning tree as a subgraph: (a) the 3-star (b) the bird graph

The graph on top of the points $a_1, .., a_6$ in Figure 1(b) is called the *bird graph*. It can be seen as a triangulated composition of two 3-stars. In the next lemma we show that the bird graph does not contain a pointed spanning tree either.

Lemma 2 *The bird graph does not contain a pointed spanning tree as a subgraph.*

Proof Assume that the bird graph contains a pointed spanning tree T as a subgraph. Because of connectivity, the edges a_1a_2 and a_5a_6 are in T. The edge a_2a_5 cannot be in T, as otherwise any edge incident to a_3 would violate the pointedness condition in either a_2 or a_5. Thus, either a_3 or a_4 has to be connected to both, a_2 and a_5. This prevents the other vertex, a_4 resp. a_3, to be connected anyhow. □

In a next step, we extend the bird graph by two additional points b_1, b_2, see Figure 2. Intuitively speaking b_1 and b_2, respectively, are connected by edges to each visible point of the bird graph. Moreover we add the edge b_1, b_2. We call the resulting full triangulation of the triangle b_1, b_2, a_1 with interior points $a_2, .., a_6$ the *bird cage graph*. The following lemma shows that it will play a crucial role in the construction of a triangulation not containing a pointed spanning tree.

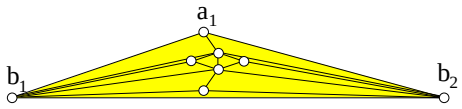


Fig. 2. The bird cage graph: any connected, pointed, spanning subgraph contains at least one interior edge incident to b_1 or b_2, respectively.

Lemma 3 *Any connected, pointed, spanning subgraph of the bird cage graph contains at least one interior edge incident to b_1 or b_2, respectively.*

Proof Let A be a connected, pointed, spanning subgraph of the bird cage graph. By Lemma 2, the subgraph B of A induced by the points $a_1, .., a_6$ does not contain a pointed spanning tree. That is, B consists of at least two components. Because of connectivity, A has to include edges that connect these components. For this, at least one interior edge incident to b_1 or b_2, respectively, has to be in A. □

We are now ready to prove the main theorem of this note.

Theorem 4 *There exist triangulations on top of a point set in the plane in general position that do not contain a pointed spanning tree as a subgraph.*

Proof As indicated in Figure 3 we connect three points a, b, c pairwise by bird cage graphs (shaded triangles), such that for each bird cage graph the

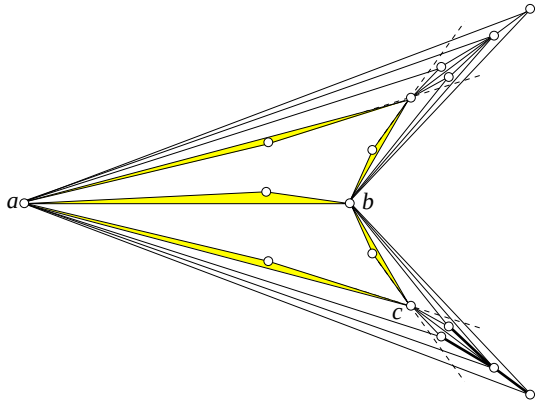


Fig. 3. A bird wing graph containing five bird cage graphs.

connected vertices correspond to b_1 and b_2, respectively. Furthermore, we add four points near point c. This four points form a three star and are connected to a, b and c as shown in the figure. Next the whole construction, except the bird cage graph connecting a to b, is mirrored along the line a,b. We call the resulting graph a bird wing graph, cf. Figure 3. Note that all edges incident to c form three wedges in an obvious way. If we group the edges of a wedge togehter the resulting graph corresponds to a 3-star with c as its center. The same holds for b and its incident edges. To complete our construction we finally form another 3-star like graph with center a' by joining three bird wing graphs at their a-vertices, see Figure 4. Let us denote the resulting graph by $\mathcal{G}$ and its vertex set by $\mathcal{S}$.

Let G be any planar straight-line graph drawing on top of $\mathcal{S}$ which contains $\mathcal{G}$ as a subgraph. Note that G might be a complete triangulation of $\mathcal{S}$. Assume that there exists a pointed spanning tree PST as a subgraph of G. By similar argumentation as for the 3-star, PST does not contain any edge incident to a' in at least one of the three bird wing graphs. W.l.o.g. let b be a vertex of this bird wing graph. Applying Lemma 3, PST has to contain at least one edge in the interior of the bird cage graph between a' and b incident to b. From the property of the 3-star we can find a bird cage graph B incident to b such that PST does not contain any edge incident to b in B. W.l.o.g. let c be the vertex at the other end of B. By Lemma 3, PST contains at least one edge incident to c in B. The same holds for the bird cage graph between a' and c. Moreover, the four additional points near

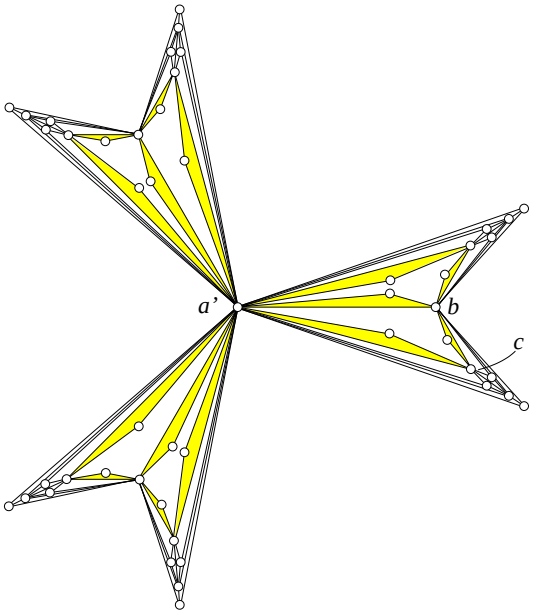


Fig. 4. A graph consisting of three bird wing graphs.

point c cannot be connected by any edge to either a', b, or c without destroying pointedness at one of these points. Since PST is connected, the three bold edges in Figure 3 have to be in PST. However, the resulting graph is not pointed any more, as the three bold edges form a 3-star. Thus, there exists no pointed spanning tree as a subgraph for G. Note that all arguments work for any planar straight-line graph containing $\mathcal{G}$ as a subgraph, because our argumentation is solely based on the interior of fully triangulated areas. □

3. Some Implications

The example in the proof of Theorem 4 is constructed on top of a point set S with 124 points. Adding more points to S in the outer face of $\mathcal{G}$ and completing the extended graph with edges to a full triangulation still gives a triangulation that does not contain a pointed spanning tree. Thus, Theorem 4 also holds if we put additional restrictions on the size of the convex hull of the underlying point set.

As a more interesting consequence we get a lower bound on the minimum number of edge flips that might be necessary to transform a triangula-

tion such that it contains a pointed spanning tree as a subgraph. To this end, we combine a linear number of disjoint copies of the 124-point example. After completing the graph on the resulting point set to a full triangulation, at least one flip has to be executed in each copy.

Corollary 5 *There exist triangulations on top of an n-point set in the plane in general position that require $\Omega(n)$ edge flips to contain a pointed spanning tree as a subgraph.*

From the Delaunay flip algorithm it follows that a quadratic number of flips is always sufficient to obtain a pointed spanning tree as a subgraph. So far no better upper bound on this flip distance is known.

Of particular interest is the investigation of Hamiltonicity of triangulations. A triangulation is called Hamiltonian if it contains a Hamiltonian cycle. Let T be a non-Hamiltonian triangulation. What is the minimum number of edge flips that is always sufficient to come from T to a Hamiltonian triangulation T'? Note that Hamiltonicity implies the existence of a pointed spanning tree, whereas the reverse is not true in general. Therefore, we conclude from Corollary 5:

Corollary 6 *There exist non-Hamiltonian triangulations on top of an n-point set in the plane in general position that require $\Omega(n)$ edge flips to become Hamiltonian.*

The last statement can also be shown in a more direct way. Let a point set S with $|S| = n$ be given such that the convex hull of S contains 3 points. Then a triangulation on top of S has $2n - 5$ triangles. We place one additional point into each of these triangles and connect it by edges to the corners of the triangle. The set A of inserted points is independent, meaning that there is no edge between any two points of A in the resulting triangulation T on top of $S \cup A$. Assume there exists a sequence δ of vertices forming a Hamiltonian cycle. Between any two vertices of A in δ there must be at least one vertex of S because A is an independent set. Since $|A| > |S|$, T cannot be Hamiltonian, i.e., δ cannot exist. If we want T to become Hamiltonian we must perform a sequence of flips which reduces $|A|$. Any flip connects at most two elements of A and therefore reduces $|A|$ by at most 1. Hence, a linear number of flips is necessary because $|A|$ is about twice the cardinality of S. □

4. Open Problems

There are several related open questions. First, is there a smaller (in the number of points) counterexample to Conjecture 1? Moreover, how fast can we decide whether a given triangulation contains a pointed spanning tree as a subgraph? And if the answer is positive, how fast can we compute this tree? Regarding flipping, what are tight bounds on the required number of edge flips to transform a triangulation such that it contains a pointed spanning tree or a Hamiltonian cycle, respectively?

Acknowledgments

The innocent looking conjecture that any triangulation contains a pointed spanning tree as a subgraph has fascinated several people. On countless (unsuccessful) attempts to prove it true, several colleagues generously shared their inspiring ideas with us. We want to thank Franz Aurenhammer, Sergei Bespamyatnikh, Prosenjit Bose, Stefan Langerman, Pat Morin, Bettina Speckmann, Csaba D. Tóth, and David R. Wood for enjoyable discussions.

References

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

[2] E.D. Demaine, J.S.B. Mitchell, J. O'Rourke, editors, The Open Problems Project, `http://cs.smith.edu/~orourke/TOPP/Welcome.html`.

[3] W. Schnyder, *Embedding planar graphs on the grid.* Proc. of the First Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 138-148, SIAM, Philadelphia, 1990.

Distributed Ranking Methods for Geographic Information Retrieval

Marc van Kreveld Iris Reinbacher Avi Arampatzis Roelof van Zwol
Institute for Information & Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands
{*marc,iris,avgerino,roelof*}*@cs.uu.nl*

Abstract

Geographic Information Retrieval is concerned with retrieving documents that are related to some location. This paper addresses the ranking of documents by both textual relevance and spatial relevance. To this end, we introduce *distributed ranking*, where similar documents are ranked spreaded in the list instead of sequentially. The effect of this is that documents close together in the ranked list have less redundant information. We present various ranking methods and efficient algorithms for them.

1. Introduction

The most common way to return a set of documents obtained from a Web query is by a ranked list. The search engine attempts to determine which document seems to be the most relevant to the user and will put it first in the list. In short, every document receives a *score*, or *distance to the query*, and the returned documents are sorted by this score or distance.

There are situations where the sorting by score may not be the most useful one. When a more complex query is done, composed of more than one query term or aspect, documents can also be returned with two or more scores instead of one. This is particularly useful in *geographic information retrieval* [5,6,8]. For example, the Web search could be for castles in the neighborhood of Koblenz, and the documents returned ideally have a score for the query term "castle" and a score for the closeness to Koblenz. This implies that a Web document resulting from this query can be mapped to a point in the 2-dimensional plane.

A cluster of points in this plane could be several documents about the same castle. If this castle is in the immediate vicinity of Koblenz, all of these documents would be ranked high in the sorted list, provided that they also have a high score on the term "castle". However, the user probably also wants documents about other castles that may be a bit further away, especially when these documents are more relevant for the term "castle". To incorporate this idea in the ranking, we introduce *distributed ranking* in this paper. We present various models that generate ranked lists that also have diversity. We also present efficient algorithms that compute the distributed rankings. To keep server load low, it is important to have efficient algorithms.

There are other situations where ranking based on two scores shows up: scores of two textual terms, or of a textual term and metadata information. A common example of metadata is the number of hyperlinks that link to a document. Standard in information retrieval is to combine the two scores into a single score (e.g., by a weighted sum), which produces the ranked list by sorting. Besides the problem that it is unclear how the two scores should be combined, it also makes distributed ranking impossible. Two documents with the same combined score could be similar documents or quite different. If two documents have two scores that are the same, one has more reason to suspect that the documents themselves are similar.

The topic of geographic information retrieval is studied in the SPIRIT project [5]. The idea is to build a search engine that has spatial intelligence because it will understand spatial relationships like

close to, *to the North of*, *adjacent to*, and *inside*, for example. The core search engine will process a user query in such a way that both the term relevance and the spatial relevance of a document is obtained in a score. This is possible because the search engine will not only have a term index, but also a spatial index. These two indices provide the two scores that are needed to obtain a distributed ranking. The ranking study presented here will form part of the geographic search engine to be developed for the SPIRIT project.

Related research has been conducted in various papers [1,3,6]. They address geographic information retrieval or text document analysis, and use more than one score to rank results. The scores are combined into one score by a weighting formula. In some papers, the *additional* relevance is determined, which depends on documents ranked earlier. This is quite similar to our approach. However, we have a different starting point because we assume that we have two scores of one document based on two aspects. Furthermore, other papers do not give algorithms and efficiency analyses.

2. Distributed Ranking Methods

In this section we will present specific ranking methods. We will focus on the two dimensional case only. So we assume that a Web query has been done, and a number of relevant documents were found. Each document is associated with two scores, for example a term score and a spatial score. The relevant documents are mapped to points in the plane, and the query is also mapped to a point. We perform the mapping in such a way that the query is a point Q at the origin, and the documents are mapped to points $p_1, \ldots, p_n$ in the upper right quadrant of Q where documents with high scores are points close to Q. The point with the smallest Euclidean distance to the query is considered the most relevant document and is always first in any ranking. The remaining points are ranked with respect to already ranked points. At any moment during the ranking, we have a subset $R \subset P$ of points that have already been ranked, and a subset $U \subset P$ of points that are not ranked yet. We choose from U the "best" point to rank next, where "best" is determined by a scoring function that depends both on the distance to the query Q and on the set R of ranked points. Intuitively, an unranked point p has a higher added value or relevance if it is not close to any ranked points in R.

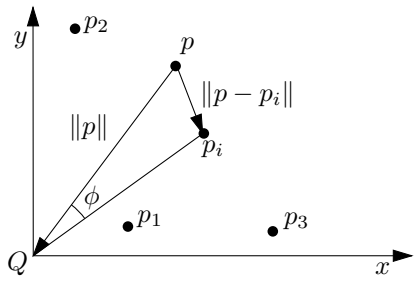


Fig. 1. An unranked point p and ranked points p_1, p_2, p_3, p_i, where p is closest to p_i by distance and by angle.

For every unranked point p, we consider only the closest point $p_i \in R$, where closeness is measured either in the Euclidean sense, or by angle with respect to the query point Q. This is illustrated by $\|p - p_i\|$ and ϕ, respectively, in Figure 1. Using the angle to evaluate the similarity of p and p_i seems less precise than using the Euclidean distance, but it allows for more efficient algorithms, and certain extensions of angle-based ranking methods give well-distributed results.

2.1. *Distance to query and angle to ranked*

Our first ranking method uses the angle measure to obtain the similarity between an unranked point and a ranked point. In Figure 1, consider the angle $\phi = \phi(p, p_i)$ and rank according to the score $S(p, R) \in [0, 1]$, which can be derived from the following normalized equation:

$$S(p, R) = \min_{p_i \in R} \left(\frac{2(\phi(p, p_i) + c)}{\pi + 2c} \cdot \left(\frac{1}{1 + \|p\|}\right)^k \right) \tag{1}$$

Here, k denotes a constant; if $k < 1$, the emphasis lies on the distribution, if $k > 1$, we assign a bigger weight to the proximity to the query. The additive constant $0 < c \ll 1$ ensures that all unranked points $p \in U$ are assigned a positive score. During the ranking algorithm, we always choose the unranked point p that has the highest $S(p, R)$ score and rank it next. This implies an addition to the set R, and hence, recomputation of the scores of unranked points may be necessary. We first give a generic, quadratic time algorithm.

Algorithm 1:

(i) Rank the point p closest to the query Q first. Delete it from the point set P.
(ii) For every unranked point $p \in P$ do
  (a) Store with p the point $p_i \in R$ with the smallest angle to p.
  (b) Compute the score $S(p, R) = S(p, p_i)$.
(iii) Choose the point with the highest score $S(p, R)$ as next in the ranking; add it to R and delete it from P.
(iv) Compute for every point $p' \in P$ the angle to the last ranked point p. If it is smaller than the angle of the point stored with p', then store p with p' and update the score $S(p', R)$. Continue with step (iii).

This simple, quadratic time algorithm can easily be modified to work for different score functions.

Theorem 1 *A set of n points in the plane can be ranked according to the distance-to-query and angle-to-ranked model in $O(n^2)$ time.*

2.2. *Distance to query and distance to ranked*

We next study a model based the distance to the closest ranked point. In Figure 1, consider the distance $\|p - p_i\|$ from p to the closest ranked point p_i and rank according to the outcome of the following equation:

$$S(p, R) = \min_{p_i \in R} \left(\frac{\|p - p_i\|}{\|p\|^2} \right) \tag{2}$$

A normalized equation such that $S(p, R) \in [0, 1]$ is the following:

$$S(p, R) = \min_{p_i \in R} \left((1 - \mathrm{e}^{-\lambda \cdot \|p - p_i\|}) \cdot \frac{1}{1 + \|p\|} \right) \tag{3}$$

Here, λ is a constant that defines the slope of the exponential function. Algorithm 1 can be modified to work here as well with the same running time.

Theorem 2 *A set of n points in the plane can be ranked according to the distance-to-query and distance-to-ranked model in $O(n^2)$ time.*

2.3. *Addition models*

So far, our distributed methods were all based on a formula that divided angle or distance to the closest ranked point by the distance to the query. In this way, points closer to the query get a higher relevance. We can obtain a similar effect but a different ranking by adding up these values. It is not clear beforehand which model will be more satisfactory for users, so we analyze these models as well.

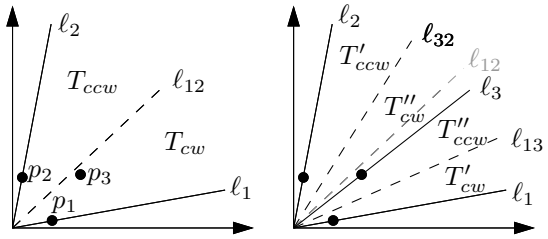


Fig. 2. The split and concatenate of trees in Algorithm 2.

$$S(p, R) = \min_{p_i \in R} \Big(\alpha \cdot (1 - \mathrm{e}^{-\lambda \cdot (\|p\| / \|p_{max}\|)}) + (1 - \alpha) \cdot \phi(p, p_i) \cdot \frac{2}{\pi} \Big) \tag{4}$$

In this equation, p_{max} is the point with maximum distance to the query, $\alpha \in [0, 1]$ denotes a variable which is used to put an emphasis on either distance or angle, and λ is a constant that defines the slope of the exponential function. Algorithm 1 can be modified for this addition model, but because of the fact that the angle is now only an additive and not a multiplicative part of the score equation, we can give algorithms with better running time.

The point set is initially stored in the leaves of a binary tree T, sorted by counterclockwise (ccw) angle to the y-axis. In every leaf of the tree we also store: (i) ccw and clockwise (cw) angle to y and x-axis respectively; (ii) the distance to the query; (iii) ccw and cw score. We augment T as follows (see e.g. (Cormen et al. 1990) for augmenting data structures): In every internal node we store the best cw and ccw score per subtree. In the root we additionally store the angle of the closest ccw and cw ranked point and whether the closest ranked point is in cw or ccw direction. Additionally we store the best score per tree in a heap for quicker localization. As shown left in Figure 2, between two already ranked points p_1 and p_2, indicated by ℓ_1 and ℓ_2, there are two binary trees, cw and ccw of the bisecting barrier line ℓ_{12}. All the points in T_{ccw} are closer in angle to p_2 and all the points in T_{cw} are closer in angle to p_1. If we insert a new point p_3 to the ranking, this means we insert a new imaginary line ℓ_3 through p_3 and we need to perform the following operations on the trees:

(i) Split T_{cw} and T_{ccw} at the angle-bisectors ℓ_{32} and ℓ_{13}, creating the new trees T'_{cw} and T'_{ccw}

and two intermediate trees $\overline{T}_{cw}$ and $\overline{T}_{ccw}$

(ii) Concatenate the intermediate trees from (i), creating one tree $\overline{T}$.

(iii) Split $\overline{T}$ at the newly ranked point p_3, creating T''_{cw} and T''_{ccw}.

Figure 2 right, shows the outcome of these operations. Whenever we split or concatenate the binary trees we need to make sure that the augmentation remains correct, which is standard. We also need to update the information in the root of each tree about the closest cw and ccw ranked point and the best scores. As the scores are additive, and all scores for points in the same tree are calculated with respect to the same ranked point, we simply subtract the cw (ccw) angle of the closest ranked point from the cw (ccw) best score to get the new best score for the tree. Furthermore we need to update the score information in the heap. Now we can formulate an algorithm for the addition-model that runs in optimal $O(n \log n)$ time.

Algorithm 2:

(i) Create T with all points of P, the augmentation and a heap that contains only the point p closest to the query Q.

(ii) Choose the point p with the highest score $S(p, R)$ as next in the ranking by deleting the best one from the heap.

(iii) For every last ranked point p do:

  (a) Split and concatenate the binary trees as described above and update the information in their roots.

  (b) Update the best-score information in the heap: delete the remaining best score of the old tree that did not contain p and insert the four best scores of the new trees.

Theorem 3 *A set of n points in the plane can be ranked according to the angle-distance addition model in $O(n \log n)$ time.*

Another, similar, addition model adds up the distance to the query and the distance to the closest ranked point. Algorithm 2 is not applicable for this addition model, because the distance to the closest ranked point does not change by the same amount for a group of points as the angle. This implies that the score for every unranked point needs to be adjusted individually when adding a point to R. We can modify Algorithm 1 for this addition model. Alternatively, we can use an algorithm that has $O(n^2)$ running time in the worst case, but a typical running time of $O(n \log n)$, as in [4,7]. The idea is to maintain the Voronoi diagram of the ranked points, and with each Voronoi cell, maintain the unranked points in it. This practically fast algorithm can be applied to all ranking methods described thus far.

3. Conclusions

This paper introduced distributed relevance ranking for documents that have two scores. In the full paper we present more models and also several extensions of the models. We have implemented the rankings to inspect how well they spread, while not sacrificing distance to the query too much. It seems that certain extensions of the models presented here perform best. However, user evaluation is needed (and is planned) to test this properly.

References

[1] J.G. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Research and Development in Information Retrieval*, pages 335–336, 1998.

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.

[3] J. Goldstein, V.O. Mittal, J.G. Carbonell, and J.P. Callan. Creating and evaluating multi-document sentence extract summaries. In *Proc. CIKM*, pages 165–172, 2000.

[4] P. S. Heckbert and M. Garland. Fast polygonal approximation of terrains and height fields. Report CMU-CS-95-181, Carnegie Mellon University, 1995.

[5] C.B. Jones, R. Purves, A. Ruas, M. Sanderson, M. Sester, M.J. van Kreveld, and R. Weibel. Spatial information retrieval and geographical ontologies – an overview of the SPIRIT project. In *Proc. 25th Annu. Int. Conf. on Research and Development in Information Retrieval (SIGIR 2002)*, pages 387–388, 2002.

[6] E. Rauch, M. Bukatin, and K. Naker. A confidence-based framework for disambiguating geographic terms. In *Proc. Workshop on the Analysis of Geographic References*, 2003.

[7] M. van Kreveld, R. van Oostrum, and J. Snoeyink. Efficient settlement selection for interactive display. In *Proc. Auto-Carto 13: ACSM/ASPRS Annual Convention Technical Papers*, pages 287–296, 1997.

[8] U. Visser, T. Vögele, and C. Schlieder. Spatio-terminological information retrieval using the BUSTER system. In *Proc. of the EnviroInfo*, pages 93–100, 2002.

Improved results for the k-Centrum straight-line location problem

Antonio J. Lozano [a,1], Juan A. Mesa [*,b,1] and Frank Plastria [c]

[a] *Dpto. de Matemáticas, Universidad de Huelva, Spain*

[b] *Dpto. de Matemática Aplicada II, Escuela Superior de Ingenieros (Universidad de Sevilla), Camino de los Descubrimientos, s/n, 41092, Sevilla, Spain*

[c] *Dep. of Management Informatics, Vrije Universiteit, Brussel, Belgium*

Abstract

The k-Centrum problem consists in finding a point that minimises the sum of the distances to the k farthest points out of a set of given points. It encloses as particular cases to two of the most known problems in Location Analysis: the center, also named as the minimum enclosing circle, and the median. In this paper the k-Centrum criteria is applied to obtaining a straight line-shaped facility. A reduced finite dominant set is determined and an algorithm with lower complexity than the previous one obtained.

Key words: Location, Duality

1. Introduction

Given a set of n demand points, the median straight line problem consists in finding the straight line that minimises the sum of the distances to the points. The center straight line problem looks for the straight line that minimises the distance to the farthest point. A k-Centrum straight line minimises the sum of the distances to the k farthest demand points. If $k = 1$ then the corresponding problem is that of the center and for $k = n$ the median, thus enclosing both problems. However, last is not the only reason for the relevancy of this problem: it provides the decision maker a more flexible tool.

The k-Centrum criteria has been applied in point location facility in different contexts. When the underlying structure is a graph, Tamir [9] has devised efficient algorithms for different cases (chain graph, tree, general graphs, multiple case). The corresponding repulsive problem, that of maximising the sum of the distances to the k nearest points has also been studied both in graphs [5] and in the Euclidean plane [4]. Some of the classic location problems have been extended to extensive facilities (i.e. those that can not be represented as isolated points). In particular, the center and the median line location problem have been intensively researched (e.g. Lee and Wu [3], Morris and Norback [7], Schöebel [8].) The only work on the k-Centrum straight line problem is [6] in which a finite dominating set is determined and an $O(n^4 \log n)$ time algorithm described.

2. Finite Dominating Set

Given a set of points $P = \{p_1, p_2, \ldots, p_n\} \subset \mathbb{R}^2$ and the weight set associated $W = \{w_1, w_2, \ldots, w_n\}$ the straight line k-Centrum problem consists in finding a line such that it minimises the function:

$$f(l) = \max_{Q \subset P, |Q|=k} \sum_{p \in Q} w_p d(p, l)$$

where $d(\cdot, \cdot)$ is the Euclidean point-line distance.

The problem can be stated in the dual plane as finding the point l^* such that it minimises the dual objective function:

* Corresponding author

Email addresses: antonio.lozano@dmat.uhu.es (Antonio J. Lozano), jmesa@us.es (Juan A. Mesa), Frank.Plastria@vub.ac.be (Frank Plastria).

[1] Partly supported by project MCyT BFM2000-1052-C02-01

$$f^*(l^*) = \max_{Q^* \subset P^*, |Q^*|=k} \sum_{p^* \in Q^*} \frac{w_p d_v(l^*, p^*)}{\sqrt{1 + {l_x^*}^2}}$$

where $d_v(\cdot,\cdot)$ is the vertical point-line distance and l_x^* is the x-coordinate of the point l^*.

In the dual plane let us consider the **V**ertical **D**istance **C**ompletely **O**rdered **L**ine **V**oronoi **D**iagram: **VDCOLVD**(P^*). The set

$$B_v(P^*) = \{\text{bis}_v(p_i^*, p_j^*); 1 \leq i \leq n, 1 \leq j \leq n, i \neq j\}$$

induces that diagram. The Voronoi regions are the union of (convex) polygonal regions neither necessarily bounded nor simply connected.

Lemma 1 *The* **VDCOLVD**(P^*) *induces a partition in the set of non-vertical straight lines of the Euclidean plane such that the collection of ordered distances to the points of P remains constant in each class.*

PROOF. For each region R of the Voronoi diagram **VDCOLVD**(P^*) there exists a permutation σ of the set $\{1, 2, \cdots, n\}$ such that

$$w_{\sigma(1)} d_v(l^*, p_{\sigma(1)}^*) \leq \cdots \leq w_{\sigma(n)} d_v(l^*, p_{\sigma(n)}^*)$$

Dividing by $\sqrt{1 + {l_x^*}^2}$ it follows

$$w_{\sigma(1)} d(p_{\sigma(1)}, l) \leq \cdots \leq w_{\sigma(n)} d(p_{\sigma(n)}, l)$$

Lemma 2 *The objective function in the dual plane f^* is quasiconcave in each connected component of each Voronoi region.*

PROOF. In each such a region CR the order of distances from lines p_i^* to points in it, remains constant, i.e. $\forall\, l^* \in CR$

$$w_{\sigma(1)} d_v(l^*, p_{\sigma(1)}^*) \leq \cdots \leq w_{\sigma(n)} d_v(l^*, p_{\sigma(n)})$$

Thus the k largest weighted vertical distances remains constant in CR and, therefore, the function

$$f^*(l^*) = \frac{\sum_{i=n-k+1}^{n} d_v(l^*, p_{\sigma(i)}^*)}{\sqrt{1 + {l_x^*}^2}}$$

is the ratio between a sum of linear functions and a positive convex function, therefore becoming a quasiconcave function.

Let us note that each extreme point of the Voronoi regions could be either

(i) a vertex of a bisector of two elements of P^*,
(ii) an intersection of two edges corresponding to the bisectors of two pairs of P^* with a common element,
(iii) an intersection of two edges corresponding to the bisectors of two disjoined pairs of elements.

However, not all of these extreme points are images of candidates for the problem in the primal space.

Theorem 3 *A finite dominant set for the straight line k-Centrum problem is composed by the elements of the set of straight lines passing through two points of P and those of the set of straight lines at equal weighted distances from three points of P.*

PROOF. Candidate straight lines correspond under the geometrical dual map to candidate points for the dual objective function. Points (i) are images of straight lines passing through two points. Points (ii) correspond to straight lines at equal distances from three points.

However, points (iii) cannot be optimal points. In effect,let l^* be such a point; then there exists two pairs $\{p_i^*, p_j^*\}$ and $\{p_s^*, p_t^*\}$ such that l^* is the intersection of the corresponding bisectors. Therefore, $w_i d_v(l^*, p_i^*) = w_j d_v(l^*, p_j^*)$ and $w_s d_v(l^*, p_s^*) = w_t d_v(l^*, p_t^*)$. Ruling out the degenerate case in which coincide the four weighted distances, we may assume that $w_i d_v(l^*, p_i^*) < w_s d_v(l^*, p_s^*)$. Then, the only relevant case to be consider is when one of the two weighted distances, say $w_i d_v(l^*, p_i^*)$, is k-rank at l^*, but in this case in each halfplane above and below the $w_i d_v(l^*, p_i^*) = w_j d_v(l^*, p_j^*)$ line and within a small enough neighbourhood of l^* the set of k-closest points does not change, so the formula for f^* remains exactly the same in each of these half-neighbourhoods of l^*, so is quasiconcave in this half-neighbourhood, in which l^* is not a vertex, so cannot be optimal.

Finally, unbounded extreme points of regions in the dual space correspond to vertical lines in the primal. A small rotation argument shows that vertical straight lines not holding one of the two above conditions cannot be optimal.

3. Algorithm

In order to examine the candidates we will consider each pair $\{p_i, p_j\}$ of points and the bisector $\text{bis}_v(p_i^*, p_j^*)$ of their dual points. For the sake of simplicity the reasoning will be done on one of the two lines of the bisector. If C_{ij}^* is the set of candidate points on this line, then C_{ij} is a family of straight lines with a common point $O_{ij} = (\text{bis}_v(p_i^*, p_j^*))^*$. In order to compute the objective function, the outer hull of the given points will be used. With this purpose the problem will be transformed in a equivalent non weighted one, with given point set

$$P' = \{p'_t = O_{ij} + w_t + \overrightarrow{O_{ij}p_t} \,/ p_t \in P\}.$$

Let us note that under this transformation the weighted distance from a point p of P to a straight line r of C_{ij} is equal to the unweighted distance from p' to r. Furthermore, for each straight line $r \in C_{ij}$ their k farthest points in P' can be found in the k outer convex hulls of P'.

A short description of the algorithm follows:

- Step 1: Compute the best vertical line.
- Step 2: For each pair $\{p_i, p_j\} \in P$,
  - Obtain the sets C_{ij} and P'.
  - Compute the k outer convex hulls for P'.
  - For each $r \in C_{ij}$ compute its objective value by obtaining their k farthest point.
- Step 3: Select the best candidate from Steps 1 and 2.

Let us note that by means of this algorithm the i-centrum ($1 \leq i \leq k$) straight lines can be computed in $\text{O}(n^3 \log n)$ time, when k is fixed.

4. Prospects

The finite dominant set given in Section 2 could be too large when it is desired to find the k-Centrum straight line only for a fixed k. In this case an alternative set would result from considering the k-1 level of a plane arrangement. Each weighted straight line of the dual plane could be substituted by two planes such that the computation of the k farthest straight lines to a point is equivalent to finding the k highest planes to a point. Then by using the results in [2] and [1] the time complexity of the finite dominant set we expect would be $O(n \log n + n^{1+\epsilon})$.

References

[1] Agarwal, P.K., Aronov, B., Chan, T.M. and Sharir, M., *On levels in arrangement of lines, segments, planes and triangles*, Discrete Computational Geometry, **19**, 315–331, 1998.

[2] Chan, T.M., *Output-sensitive results on convex hulls, extreme points and related problems*, Proc. 11th Annu. ACM Sympos. Comput. Geom., 10–19, 1995.

[3] Lee, D.T. and Wu, Y.F. *Geometric complexity of some locations problems,* Algorithmica, **1**, 193–211, 1986.

[4] Lozano, A.J., Mesa, J.A. and Plastria, F., *The Euclidean Anti-k-Centrum Problem,* Report BEIF/117, Vrije Universiteit Brussel, 1999.

[5] Lozano, A.J., Mesa, J.A. and Plastria, F., *El problema del anti-k-centrum en grafos,* ACTAS III Jornadas de Matemática Discreta y Algorítmica 110–113, Sevilla, 2002.

[6] Lozano, A.J., Mesa, J.A. and Plastria, F. *Ubicación de un servicio rectilíneo atractivo con el criterio k-centrum,* ACTAS III Encuentro Andaluz de Matemática Discreta 43–46, Almería, 2003.

[7] Morris, J.G. and Norback, J.P. *Linear facility location–Solving extensions of the basic problem,* European Journal of Operational Research, **12**, 90–94, 1983.

[8] Schöbel, A. *Locating Lines and Hyperplanes,* Kluwer Academic Publishers, 1999.

[9] Tamir, A., *The k-centrum multi-facility location problem,* Discrete Applied Mathematics, **109**, 293–307, 2001.

ON FENCING PROBLEMS

C. Miori [a] C. Peri [b] S. Segura Gomis [c]

[a] *Departamento de Análisis Matemático, Universidad de Alicante, Campus de San Vicente del Raspeig, E-03080-Alicante, Spain; borsista dell'Università Cattolica S. C. di Brescia, Italy*

[b] *Universià Cattolica S. C., Largo Gemelli 1, I-20123 Milano,Italy*

[c] *Departamento de Análisis Matemático, Universidad de Alicante, Campus de San Vicente del Raspeig, E-03080-Alicante, Spain*

Abstract

Fencing problems deal with the bisection of a convex body in a way that some geometric measures are optimized. We study bisections of planar bounded convex sets by straight line cuts and also bisections by hyperplane cuts for convex bodies in higher dimensions.

Key words: fencing problems, optimization, relative diameter.

1. Introduction

Fencing problems regard the division of a given set into two subsets of equal area in a way that some geometric measure is maximized or minimized.

Last year we pointed out results on centrally symmetric convex sets in two dimension. Now we present a statement in the general planar case, where K is a general convex body. Moreover we study fencing problems in higher dimension.

First we need to recall definitions and some results.

Definition *Let K be a planar, bounded, convex set and let $K_1, K_2 \subset K$. We say that $\{K_1, K_2\}$ is a* ***bisection*** *of K if the following conditions hold:*

(i) $K = K_1 \cup K_2$.
(ii) $int(K_1) \cap int(K_2) = \emptyset$.
(iii) $V(K_1) = V(K_2) = \frac{1}{2}V(K)$, *where $V(.)$ is the area functional.*
(iv) K_1 *and* K_2 *are connected subsets such that* $\gamma = \partial K_1 \cap \partial K_2$ *is an arc of continuous curve joining two points in* ∂K.

We say that γ ***bisects*** *K.*

In the planar case we replace the volume functional $V(.)$ by the area functional $A(.)$.

We define the *relative diameter* $d(K_1; K)$ as follows:

$$d(K_1; K) = max\{D(K_1), D(K \setminus K_1)\},$$

where $D(\cdot)$ is the diameter functional, and $\{K_1, K_2\}$ is a bisection of K.

The following propositions give us properties regarding subdivisions 1) by straight lines, 2) by general curves.

Theorem 1 *Let $\{K_1, K_2\}$ be a bisection of a planar, centrally symmetric, and bounded convex set K by a straight line l. Then*

(i) One of the ends of the diameter of K_1 is one of the intersection points of l with ∂K. Denote this point by M.
(ii) The other end of the diameter of K_1 is the intersection of ∂K with the smallest circle centered at M and containing K_1; the radius of this circle is obviously $d(K_1; K)$.

Theorem 2 *Let K be a planar, centrally symmetric, and bounded convex set. Let γ be a continuous curve bisecting K into two connected subsets E_1 and E_2. Suppose that $d(E_1; K)$ realizes the minimum of the relative diameter for all the curves bisecting K. Then there exist a straight line, passing through the*

Email addresses: cm4@alu.ua.es (C. Miori), carla.peri@unicatt.it (C. Peri), Salvador.Segura@ua.es (S. Segura Gomis).

center of K, bisecting K into two subsets $\{F_1, F_2\}$ such that $d(F_1; K) = d(E_1; K)$.

Moreover we obtained a global estimate for the ratio between the area and the relative diameter:

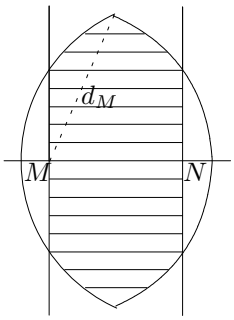
Theorem 3 *Let K be a planar, bounded, and centrally-symmetric convex set with area A. For every bisection $\{K_1, K_2\}$ of K the relative diameter satisfies the following inequality*

$$d(K_1; K) \geq C\sqrt{A}$$

where

$$C \cong 0.8815....$$

Equality holds if K is the body K_{φ_0} described in the figure:



2. Minimizing the relative diameter for planar convex sets

Now we shall consider bisections of a general convex set by straight line cuts. We prove that centrally symmetric convex sets minimize the relative diameter.

First of all we need to introduce the following notations. Let us consider a general convex set K, and let l be the straight line bisecting K; let l^+ and l^- be the half-planes defined by such a line. Let us define:

$$d(K) = min_{l \cap K \neq \emptyset} d_l(K),$$

where $d_l(K) = max\{D(K \cap l^+), D(K \cap l^-)\}$.

Theorem 4 *In the class of convex sets K with area A the minimum of the relative diameter, with respect to straight line cuts, is attained on a centrally symmetric convex set.*

PROOF. Let the area A of K be fixed, and let l_0 be the line such that

$$d_{l_0}(K) = d(K).$$

Let us choose the origin to be the midpoint of the chord $K \cap l_0$. Let us apply Steiner symmetrization with respect to $l_0^{\perp}$. Then we obtain a new body $K^{'}$ such that $A(K^{'} \cap l^+) = A(K^{'} \cap l^-) = \frac{1}{2}A$ and

$$D(K^{'} \cap l_0^+) \leq D(K \cap l_0^+),$$

$$D(K^{'} \cap l_0^-) \leq D(K \cap l_0^-),$$

so that

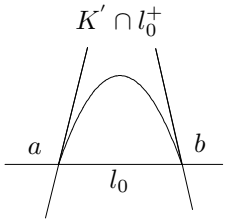
$$d_{l_0}(K^{'}) = max\{D(K^{'} \cap l_0^+), D(K^{'} \cap l_0^-)\} \leq$$

$$\leq max\{D(K \cap l_0^+), D(K \cap l_0^-)\} = d_{l_0}(K).$$

In order to minimize the relative diameter for fixed area it is enough to consider bodies which are symmetric with respect to $l_0^{\perp}$, where l_0 realize the minimal cut. Let $K^{'}$ be such a body.

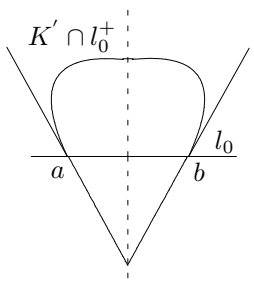
Let now $K^{'} \cap l_0^+$ the part of $K^{'}$ such that $d(K^{'}) = D(K^{'} \cap l_0^+)$. We consider two cases.

1) First case: the support lines of $K^{'}$ at a, b (endpoints of $K^{'} \cap l_0$) meet at a point $p \in l^+$ or are parallel. Then we have the following situation.



Applying the symmetrization with respect to the line l_0 we obtain a new body which is centrally symmetric and has the same relative diameter as $K^{'}$. We apply then the argument for centrally symmetric bodies in order to get the best constant.

2) Second case: let us suppose that the support lines of $K^{'}$ at a, and b meet at a point $p \in l^-$.



Since

$$d(K') = D(K' \cap l_0^+) \geq D(K' \cap l_0^-)$$

then

$$\forall x \in K' \cap l_0^+ : dist(a,x) \leq d(K')$$
$$\forall x \in K' \cap l_0^- : dist(b,x) \leq d(K')$$

Thus K' is contained in the intersection of the two disks with centers a, b respectively and radius $d(K')$. Moreover $K' \cap l_0^- \in T$ where T is the intersection of the previous lens with the triangle in the half-plane l_0^- determined by the support lines passing trough a and b. So if we take as new body $\widetilde{K}$ given by the union of T and its symmetric with respect to l_0 we have a centrally symmetric convex body where l_0 realizes a bisection having $d(\widetilde{K}) = d(K')$ and area of $\widetilde{K}$ greater then the area of K'.

3. Fencing problems on higher dimension

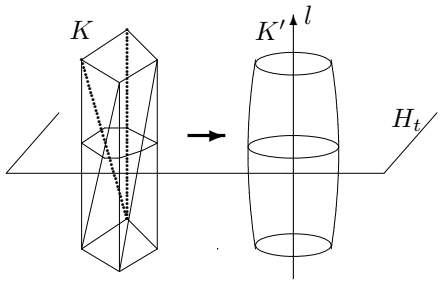
We now consider fencing problems on higher dimension determined by hyperplane cuts. We say that a convex body K on $\mathbb{E}^d$ is a *minimizer* for such a fencing problem if the relative diameter of a bisection $\{K_1, K_2\}$ of K attain the minimum value.

We give a necessary condition for a convex body K to be the minimizer. We also describe geometric properties of best bisections of centrally symmetric convex bodies and obtain a lower bound for the ratio

$$\frac{d(K_1;K)}{V(K)^{1/d}}.$$

Theorem 5 *In the class of convex bodies in $\mathbb{E}^d$ with volume V the minimum of $d(K_1;K)$, with respect to hyperplane cuts, is attained on a centrally symmetric convex body of revolution.*

PROOF. Let $K \subset \mathbb{E}^d$ be a body for which $d(K_1;K)$ attains its minimum, and let H be the hyperplane cutting K into two parts K_1, K_2 such that $d(K_1;K) = D(K_1)$. By applying Schwarz rounding symmetrization (see [2] and [12]) with respect to a line l perpendicular to H, we obtain a body of revolution K', with same volume V, which is bisected by H into two parts K_1', K_2' such that $V(K_1') = V(K_2') = V/2$. As the Schwarz symmetrization does not increase the diameter, the value of $d(K_1;K)$ does not increase, so that by the minimality of K, the body K' also realizes the minimum of $d(K_1;K)$.



Let us consider the tangent cone Γ of K' along the $(d-1)$-sphere $K' \cap H$. We distinguish two cases.

(i) Suppose that Γ is a right cylinder and denote by K_2'' the image of K_1' by the orthogonal reflection with respect to the hyperplane H. Then the convex body $K'' = K_2'' \cup K_1'$ is a centrally symmetric convex body of revolution with a bisection $\{K_2'', K_1'\}$ for which $d(K_1;K)$ attains its minimum.

(ii) Suppose that Γ is a cone with apex $v \in l$. Denote by H^+ the half-space bounded by H which contains v. Let $K_1'' = K' \cap H^+$ and let K_2'' denote the image of K_1'' by the orthogonal reflection with respect to the hyperplane H. Then the convex body $K'' = K_2'' \cup K_1^{''}$ is a centrally symmetric convex body of revolution with a bisection $\left\{K_2'', K_1^{''}\right\}$ for which $d(K_1;K) = D(K_2'') = D(K_1^{''}) = D(K' \cap H^+) \leq D(K_1)$. Thus $d(K_1;K)$ attains its minimum on K'' as well.

Therefore, in order to minimize the ratio

$$\frac{d(K_1;K)}{V(K)^{1/d}}$$

it is enough to find for fixed $d(K_1; K)$ the body with maximum volume within the class of centrally symmetric bodies of revolution. To this end we first extend Proposition 1 on the planar centrally symmetric case.

Theorem 6 *Let $\{K_1, K_2\}$ be a bisection of a centrally symmetric convex body K by a hyperplane H, and let $D(K_1) = \max\{D(K_1), D(K_2)\}$. Then*
- *one of the end points of the diameter of K_1 belongs to $H \cap \partial K$. Let M be such a point.*
- *The other endpoint of the diameter is the intersection of ∂K with the sphere centered at M with radius $D(K_1)$.*

PROOF. Let A, B denote the endpoints of the diameter of K_1. We can assume that the center of K is the origin O. Suppose by contradiction that $A, B \notin H \cap \partial K$. By construction the points A, B, O are not collinear. Let us consider the plane π passing through A, B, O. We define $K' = K \cap \pi$, $K'_1 = K_1 \cap \pi$, $K'_2 = K_2 \cap \pi$. We obtain a bisection of the centrally symmetric planar convex body K' such that $\max\{D(K'_1), D(K'_2)\}$ is attained on the segment with endpoints A, B. This contradicts Proposition 1 on the planar centrally symmetric case.

References

[1] A. S. Bernstein: Über die isoperimetrsche Eigenscheft des kreises auf der Kugeloberfläche und in der Ebene, *Math.Ann* **60**(1905), 117-136.

[2] T. Bonnesen, W. Fenchel: Theorie der konvexen Körper, *Springer-Verlag, Berlin*(1934).

[3] A. Cianchi: On relative isoperimetric inequalities in the plane, *Boll. Unione Mat. Italiana* **7** 3-B (1989), 289–325.

[4] A. Cerdán, U.Schnell, S. Segura Gomis: On Relative Geometric Inequalities, (2003 Preprint).

[5] H. T. Croft, K. J. Falconer, R. K. Guy: Unsolved problems in Geometry, 1991, **A26**.

[6] H.G. Eggleston: The maximal length of chords bisecting the area or perimeter length of plane convex sets,*Journal London Math. Soc.*, 1961,**36**.

[7] H. Martini: Cross-Sectional Measures, *Colloquia Mathematica Societatis*, 1991, **63**, 269-310.

[8] C. Peri: On relative isoperimetric inequalities, *Conferenze del Seminario di Matematica dell'Università di Bari*,2001, **279**.

[9] J. Pál: Ein minimunmproblem für Ovale, *Math. Ann.*, **83**, (1921), 311-319.

[10] G. Pólya: Aufgabe 283, *Elem.Math.*, 1958, **13**, 40-41.

[11] K. Radziszewski: Sur les cordes qui partagent le périmètre d'un ovale en 2 parties égales, *Ann. Univ. Mariae Curie-Sklodowska Sect.* A**8**(1954) 89-92.

[12] J. R. Sangwine-Yager: Mixed Volumes, *Handbook of Convexity (P. M. Gruber and J. M. Wills eds.)* (1993), 43-71.

[13] L. A. Santaló: An inequality between the parts into which a convex body is divided by a plane section, *Rend. Circ. Mat. Palermo (2)* **32** (1983) 124-130.

Space-Efficient Geometric Divide-and-Conquer Algorithms

P. Bose [a,1], A. Maheshwari [a,1], P. Morin [a,1], J. Morrison [a,1], M. Smid [a,1], J. Vahrenhold [b,2]

[a] *School of Computer Science, Carleton University, Ottawa, Ontario, Canada K1S 5B6*

[b] *Westfälische Wilhelms-Universität, Institut für Informatik, 48149 Münster, Germany*

Abstract

We present an approach to simulate divide-and-conquer algorithms in a space-efficient way, and illustrate it by giving space-efficient algorithms for the closest-pair, bichromatic closest-pair, all-nearest-neighbors, and orthogonal line segment intersection problems.

Key words: Computational geometry, space-efficient algorithms, all-nearest-neighbors, orthogonal line segment intersection

Researchers have studied space-efficient algorithms since the early 70's. Examples include merging, (multiset) sorting, and partitioning problems; see [3,7,6]. Brönnimann *et al.* [2] were the first to consider space-efficient geometric algorithms and showed how to compute the convex hull of a planar set of n points in $O(n \log h)$ time using $O(1)$ extra space, where h denotes the size of the output.

In this paper, we present a space-efficient scheme for performing divide-and-conquer algorithms without using recursion. We apply this scheme to several problems.

1. Space-efficient divide-and-conquer

In this section, we describe a simple scheme for space-efficiently performing divide-and-conquer. Using the standard recursive approach requires $\Omega(\log n)$ pointers for maintaining a recursion stack as noted in the context of space-efficiently implementing the Quicksort algorithm [5,8]. Our technique traverses the recursion tree in the same manner without requiring the extra pointers. In many cases, the same type of result can be obtained using bottom-up merge, however, it is well known that bottom-up merge has poor performance in the presence of caches.

The main idea (which is probably folklore, even though we have not seen it in the literature) is to simulate a post-order traversal of the recursion tree. We assume for simplicity that the data to be processed is stored in an array $\mathtt{A}$ of size $n = 2^k$ for some positive integer k. The recursion tree corresponding to a divide-and-conquer scheme is a perfectly balanced binary tree, in which each node at depth $0 \leq i < k$ corresponds to a subarray of the form $\mathtt{A}[j \cdot 2^{k-i} \ldots (j+1) \cdot 2^{k-i} - 1]$ for some integer $0 \leq j \leq i$.

Our scheme is presented in Algorithm 1. We maintain two indices b and e that indicate the subarray $\mathtt{A}[b \ldots e-1]$ currently processed. We will use the binary representation of the index e to implicitly store the current status of the post-order traversal, i.e., the node of the simulated recursion tree currently visited.

Algorithm 1 Stackless simulation of a post-order traversal.

1: Let $b = 0$ and $e = 1$.
2: **while** $b \neq 0$ or $e \leq n$ **do**
3: Let i be the index of the least significant bit of e (note the lowest index is 1).
4: **for** $c := 1$ to $i - 1$ **do**
5: Set $b := e - 2^c$.
6: Merge the two subarrays in $\mathtt{A}[b \ldots e-1]$
7: **end for**
8: Let $e := e + 1$.
9: **end while**

Determining the value of i (Step 3) can be done in $O(1)$ amortized time without extra space us-

[1] These authors were supported by NSERC.

[2] Part of this work was done while visiting Carleton University. Supported in part by DAAD grant D/0104616.

ing a straightforward implementation of a binary counter.

2. Nearest Neighbor Problems

2.1. *Closest Pair*

Given a set P of n points in the plane stored in an array $A[0 \dots n-1]$, the closest pair is the pair of points in P whose Euclidean distance is smallest among all pairs of points. The above scheme can be used to modify an algorithm by Bentley and Shamos [1] to compute the closest pair in space efficient manner using only $O(1)$ extra space.

We first outline Bentley and Shamos' algorithm.

Algorithm 2 Divide-and-Conquer algorithm for finding a closest pair [1].

Require: All points in the input array A are sorted according to x-coordinate.

Ensure: All points in the array A are sorted according to $<_y$, and two points realizing a closest pair in A are known.

1: If A has a constant number of points, sort the points according to $<_y$, compute a closest pair using a brute-force algorithm, and return.
2: Subdivide the array based upon the median x-coordinate and recurse on both parts.
3: Determine the minimal distance δ given by the two (locally) closest pairs of the subarrays to be merged. Set the closest pair for the current subarray to be the closer of those two points.
4: For both subarrays, extract the points that fall within a strip of width 2δ centered at the median x-coordinate.
5: Simultaneously scan through both sets containing the extracted points (backing up at most a constant number of steps for each point examined) and determine whether there is a pair of points with distance smaller than δ. Update δ and the closest pair as necessary.
6: Merge both subarrays such that all points are sorted according to $<_y$.
7: Return the closest pair.

To make this algorithm *in-place*, we require that for each "recursive" call on a subarray $A[b \dots e], e > b$, the following invariants are fulfilled as a postcondition:

Invariant 1: The first two entries $A[b]$ and $A[b+1]$ of the subarray contain two points p and q that form a closest pair in $A[b \dots e]$.

Invariant 2: $A[b] <_y A[b+1]$.

Invariant 3: If $e > b+1$, then $A[b+2 \dots e]$ is sorted according to $<_y$.

These invariants can be enforced trivially *in-place* in Step 1 of the above algorithm. After returning from the "recursive" call on $A[0 \dots n-1]$, i.e., at the end of the algorithm, the first two entries $A[0]$ and $A[1]$ contain two points that realize a closest pair.

We can transform the above algorithm into an *in-place* space-efficient variant as follows: The precondition of the algorithm can be met by running any *in-place* sorting algorithm, e.g., *heapsort*, to sort the points according to their x-coordinate.

The merge step of the divide-and-conquer scheme can be realized *in-place* as well. We partition each of the subarrays into two parts where the front part contains the points within the 2δ strip in sorted order by y-coordinate and the back part contains the points outside the strip in sorted order by y-coordinate. We use a stable *in-place* partitioning algorithm, e.g. [6], to partition each subarray.

Upon completion of the scan in step 5, to compute the sorted order of the subarrays, we simply need to perform a merge of four sorted parts (two parts are points in the strip and two parts are points outside the strip). This merging can be done *in-place* using the algorithm by Geffert et al. [3].

Theorem 1 *Given a set P of n points in the plane stored in an array $A[0 \dots n-1]$, the closest pair in P can be computed in $O(n \log n)$ time using $O(\log n)$ extra bits of storage.*

2.2. *Bichromatic Closest Pair*

In the Bichromatic Closest Pair Problem, we are given a set R of red points and a set B of blue points in the plane. The problem is to return the pair of points, one red and one blue, whose distance is minimum over all red-blue pairs. For simplicity of exposition, we assume that $|R| = |B| = n$ with the red set stored in an array $R[0 \dots n-1]$ and blue set in an array $B[0 \dots m-1]$.

We first consider solving the problem when the red and blue sets are separated by a vertical line, with red points on the left of the line and blue points on the right of the line. The approach is similar to the merge step in the previous section, except that we no longer have the luxury of the value δ. To circumvent this problem we proceed in the following way.

The above algorithm runs in linear-expected time since each time through the loop, with con-

Algorithm 3 Bichromatic closest pair when R and B are separated by a vertical line.

Require: All points in R and B are sorted by x-coordinate.

Ensure: All points R and B are sorted by y-coordinate, and the pair $R[0], B[0]$ realize the bichromatic closest pair.

1: If both R and B store only a constant number of points, sort the points according to $<_y$, compute a bichromatic closest pair using a brute-force algorithm, and return.
2: Assume $|R| \geq |B|$, otherwise reverse the roles of R and B
3: Pick a random element r from R.
4: Find the closest element of $b \in B$ to r.
5: Compute the *left envelope* of disks having radius $|rb|$ centered at each of the points in B.
6: Remove all elements of R that are outside the envelope

stant probability, we reduce the size of R or B by a constant fraction.

To implement this algorithm in-place, we observe all steps except Steps 5 and 6 are trivial to implement in-place.

Step 5, computing the left-envelope (portions of the disks visible from the point $(+\infty, 0)$), is very similar to the convex hull problem and can be solved in $O(n)$ time with an algorithm identical to Graham's scan since the points are sorted by $<_y$. The implementation of Graham's scan given by Brönnimann et al.[2] does this in-place and results in an array that contains the elements that contribute to the left envelope in the first part of the array and the elements that do not contribute in the second part of the array. Also, we observe that it is not particulary difficult to run Graham's scan "in reverse" to restore the $<_y$ sorted order of the elements in $O(n)$ time once we are done with the left envelope. Details are included in the full version of the paper.

To perform Step 6 in-place we simultaneously scan the left envelope and the red points from top to bottom and move the discarded points, using swaps to the end of the array. Again the details are exactly like the implementation of Graham's scan given by Bronnimann et al. and the algorithm can be run in reverse to recover the $<_y$ sorted order of the points. Unfortunately, the algorithm is run recursively on the sorted prefix of the array, so we need $O(\log^2 n)$ extra bits to remember the subproblem sizes at the $O(\log\ n)$ levels of recursion. Note, it may be possible to apply the algorithm by Katajainen and Pasanen [6] to remove a $\log n$ factor from the space requirement.

Theorem 2 *Given sets R and B of n points in the plane, the closest bichromatic pair can be computed in $O(n \log n)$ time using $O(\log^2)$ extra bits of storage.*

2.3. *The all-nearest-neighbors problem*

In this section, we apply the divide-and-conquer scheme of Section 1 to solve the all-nearest-neighbors problem space-efficiently. Again, we present a modification of Bentley and Shamos' algorithm.

Algorithm 4 Computing all nearest neighbors [1].

Require: All points in the input array A are sorted according to x-coordinate.

Ensure: All points in A are sorted according to $<_y$, and for each point, its nearest neighbor in A is known.

1: If A stores only a constant number of points, sort the points according to $<_y$, compute all nearest neighbors using a brute-force algorithm, and return.
2: Subdivide the array based upon the median x-coordinate $x = \ell$ and recurse on both subarrays A_0 and A_1.
3: Simultaneously scan through both subarrays A_0 and A_1, and for each point $p \in A_i$ check all points in A_{1-i} whose nearest-neighbor ball contains the projection of p onto the line $x = \ell$. Update the nearest neighbor information as necessary.
4: Merge the points according to $<_y$.

We can make this algorithm space-efficient using the framework of Section 1. One detail, however, needs special attention. Bentley and Shamos argue that for each of the points in the sets to be merged, only a constant number of other points needs be examined [1, p. 229]. More specifically, this number is four for points in two dimensions under the Euclidean metric [1, p. 228].

Note that when processing a point p, the four points above and below p's y-coordinate whose nearest neighbor balls intersect the vertical line may be interspersed (with respect to the $<_y$ order) by linearly many points. In order to provide constant-time access to these points, we proceed

as follows. We use $2n \cdot \log_2 n$ bits of extra space to maintain the following invariants that have to be fulfilled as a postcondition after each "recursive" call on a subarray $A[b \dots e], e > b$:

Invariant 1: $A[b \dots e]$ is sorted according to $<_y$.

Invariant 2: Any point $p \in A[b \dots e]$ stores an index $i \in [b \dots e]$ such that $A[i]$ is the nearest neighbor of p with respect to $A[b \dots e]$.

If these invariants are maintained throughout the algorithm, each point will store the index of its global nearest neighbor at the end of the algorithm. The main problem in performing the merge step is that while simultaneously scanning the two sorted arrays of points, i.e., *before* merging them, for each point we need to compute the index of its nearest neighbor with respect to the *merged* array. This is done in two phases. First, we do a linear scan to compute the index of each element in the *merged* array and store this index with the element (this is where we use $n \cdot \log_2 n$ extra bits). Next, we perform the merge step, and maintain a look-ahead queue of length 8 for each point set. In this queue, we maintain the indicies of the next 4 and the last 4 points seen whose nearest neighbor balls intersect the vertical line at the median x-coordinate. It is easy to see that these queues can be maintained space-efficiently while not increasing (asymptotically) the running time.

Theorem 3 *All nearest neighbors in a set of n points in the plane can be computed in $O(n \log n)$ time using $2n \log_2 n + O(\log n)$ extra bits of space.*

3. Orthogonal line segment intersection

In this section, we present a space-efficient algorithm for the orthogonal line segment intersection problem. Our algorithm can be seen as a variant of the (external-memory) *distribution sweeping* approach taken by Goodrich *et al.* [4].

The (internal memory) version of this algorithm is a top-down divide-and-conquer algorithm, that is, the (algorithmic) work is done prior to recursion. As a precondition, assume that all vertical segments are sorted according to the y-coordinate of their lower endpoint and that all horizontal segments are sorted according to their y-coordinate. In each step of the recursion, the set of vertical segments is split according to the median x-coordinate. These sets define two *slabs* that are swept top-down. All horizontal segments that completely span a slab are pushed onto a stack corresponding to their slab. Whenever (the lower endpoint of) a vertical segment is encountered, the stack corresponding to the slab containing the segment is scanned and all intersecting segments are reported. After this sweep all horizontal segments (or fragments thereof) not completely spanning a slab are distributed to the corresponding slab which in turn is processed recursively.

The first observation that helps making this algorithm space-efficient is that the "stack" used in the top-down sweep is never accessed using *push*-operations. Instead, all horizontal segments are pushed onto this stack in sorted order. This means that any sorted (part of an) array in connection with a single pointer indicating the current "top" of the "stack" can be used to implement this part of the algorithm.

A much more challenging problem is that the recursion tree corresponding to the algorithm is traversed in an *inorder* fashion, i.e., the general framework of Section 1 cannot be used. In the full paper, we will show how this algorithm can nevertheless be made space-efficient.

Theorem 4 *All k intersections in a set of n horizontal and vertical line segments can be computed in $O(n \log n + k)$ time using $O(\log n)$ bits of extra space.*

References

[1] J. L. Bentley and M. I. Shamos. Divide-and-Conquer in multidimensional space. *STOC*, pp. 220–230, 1976.

[2] H. Brönnimann, J. Iacono, J. Katajainen, P. Morin, J. Morrison, and G. T. Toussaint. Optimal in-place planar convex hull algorithms. *LATIN*, pp. 494–507, 2002.

[3] V. Geffert, J. Katajainen, and T. Pasanen. Asymptotically efficient in-place merging. *Theoretical Comp. Sci.*, 237(1–2):159–181, April 2000.

[4] M. T. Goodrich, J.-J. Tsay, D. E. Vengroff, and J. S. Vitter. External-memory computational geometry. *FOCS*, pp. 714–723, 1993.

[5] B.-C. Huang and D. E. Knuth. A one-way, stackless quicksort algorithm. *BIT*, 26:127–130, 1986.

[6] J. Katajainen and T. Pasanen. Stable minimum space partitioning in linear time. *BIT*, 32:580–585, 1992.

[7] J. Katajainen and T. Pasanen. Sorting multisets stably in minimum space. *Acta Informatica*, 31(4):301–313, 1994.

[8] L. M. Wegner. A generalized, stackless quicksort algorithm. *BIT*, 27:44–48, 1987.

Region inter-visibility in terrains

Marc van Kreveld Esther Moet René van Oostrum
Institute for Information & Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands
{*marc,esther,rene*}*@cs.uu.nl*

Abstract

A polyhedral terrain is the image of a piecewise linear continuous function defined over the triangles of a triangulation in the xy- plane. Given a terrain with n vertices, two simply-connected regions (subsets of the triangles), and any constant $\epsilon > 0$, we can determine in $O(n^{2+\epsilon})$ time and storage whether or not the two regions are completely inter-visible, which improves the $O(n^3)$ time complexity of a brute-force algorithm.

Key words: Terrains, visibility, GIS, data structures

1. Introduction

A terrain T is a triangulated polyhedral surface with n vertices $V(T) = \{v_1, v_2, ..., v_n\}$. Each vertex v_i is specified by three real numbers (x_i, y_i, z_i), which are its cartesian coordinates. Every vertical line intersects the terrain at most once, which means it can also be viewed as a piecewise linear function on $\mathbb{R}^2$.

Every triangle t on a terrain T has a normal vector associated with it. The terrain divides space into two parts; the outward normal on a triangle corresponds to the part above T, $above(T)$ and the inward normal to the part below T, $below(T)$. Because of the natural meaning, visibility between two points only makes sense if both points are on or above the terrain.

Given a terrain T in 3D and two points p and q on T, we say that p sees q if the line segment pq does not intersect $below(T)$, that is, visibility is not blocked by grazing contact with the terrain. A triangle t in T is *weakly visible* from a point $p \in \mathbb{R}^3$ if there exists at least one point on t that is visible from p. A triangle t is *strongly visible* from p if every point on t is visible from p.

A *region* is a connected subset of triangles in T. We develop an algorithm to decide whether two given regions R_1 and R_2 are completely *inter-visible*, which is equivalent to region R_1 being strongly visible from every point in region R_2 (if so, it automatically holds the other way around).

2. Related Work

Visibility computations in terrains have their main application in geographic information systems (GIS), for example computations regarding horizon pollution and signal transmission (e.g. mobile phone networks). Most of the early research considering visibility in terrains dealt with grid-based *Digital Elevation Models* (DEM), as opposed to *Triangular Irregular Networks* (TIN) which are common in recent GIS research. In algorithms research, TINs are called (polyhedral) terrains.

Algorithmic explorations of visibility in terrains were described in [4] and [10–12], for example the shortest watchtower problem. Later, more complex visibility problems were discussed: visibility computations from a moving point of view are discussed in [2], and an efficient and dynamic algorithm to maintain a visibility map for a certain viewpoint is introduced in [8].

Agarwal and Sharir [1] obtained tight bounds on the maximum number of combinatorially different views of a terrain. Finally, [6] is a recent overview of various problems concerning visbility in terrains.

3. Strong region inter-visibility

First, we look at the orientations of R_1 and R_2 in space. For the time being, we suppose no other part of T can block visibility between R_1 and R_2 than the regions themselves.

We denote the plane in 3D that contains a given triangle t as $\mathcal{P}(t)$. The half-space induced by $\mathcal{P}(t)$ that corresponds to the space above $\mathcal{P}(t)$ is denoted by $HS(t)$. Visibility is only defined above the terrain, so if a triangle t sees a point p, then p lies in $HS(t)$. If two connected sets of triangles in 3D, R_1 and R_2, can see each other, it implies that all vertices of R_1 lie in the intersection of the half-spaces induced by the triangles of R_2, and vice versa:

$$\forall v_1 \in R_1 : v_1 \in \bigcap_{t_2 \in R_2} HS(t_2) \wedge \forall v_2 \in R_2 : v_2 \in \bigcap_{t_1 \in R_1} HS(t_1) \qquad (1)$$

If a vertex from one region does not lie in the half-space intersection of the other region, it is not seen by at least one of the triangles in that region. Thus, condition (1) is necessary (but not sufficient) for strong visibility between two regions in a terrain. We say regions R_1 and R_2 are *facing each other* if condition (1) is satisfied.

When two regions R_1 and R_2 on a terrain T are facing each other, we can start to take the rest of the terrain into account. Visibility between points from different regions now depends on the terrain not blocking the view.

We limit the number of points from R_1 and R_2 we have to check for inter-visibility.

Lemma 1 *Two connected sets of triangles on a terrain R_1 and R_2 are strongly inter-visibile, if and only if*

(i) R_1 and R_2 are facing each other, and
(ii) ∂R_1 and ∂R_2 are strongly inter-visible

PROOF. The necessity of the first condition follows from the discussion above.

The necessity of the second condition follows easily. Because a region $R = int(R) \cup \partial R$, the boundaries of two regions must see each other if the entire regions see each other.

For the sufficiency of the two conditions, assume that $p \in int(R_1)$ and $q \in int(R_2)$ do not see each other. We assume that R_1 and R_2 are facing each other, and prove that there exist two points on ∂R_1 and ∂R_2 that cannot see each other.

Consider the vertical plane μ containing p and q. Let $T_\mu = \mu \cap T$ be the cross-section of the terrain, which is a lower-dimensional terrain itself. The set $\mu \cap R_1$ consists of one or more connected components, and p lies in the interior of one of them. The same statement holds for R_2 and q. We only have to consider visibility of p and q in μ. Because p and q are facing each other, the line segment $\overline{pq}$ does not intersect $below(T_\mu)$ in a neighborhood of p, nor in a neighborhood of q. Let r be the point on T_μ closest to p that is in the closure of $\overline{pq} \cap below(T_\mu)$. If p and r are in the same component of $\mu \cap R_1$, then the triangle containing r is not facing the triangle containing q. If p and r are not in the same component, then between p and r in T_μ there is a point $p' \in \partial R_1$ than cannot see q either. We repeat the argument with q and p' to find a point $q' \in \partial R_2$ that cannot see p'. Hence, if two points interior to R_1 and R_2 cannot see each other, then there exist two boundary points of R_1 and R_2 that cannot see each other. □

Checking only strong inter-visibility of the vertices on the boundary of the two regions is not sufficient, because a small peak of the terrain can block two boundary edges from being strongly inter-visible, while their endpoints can indeed see each other.

We define S to be the set containing all triangles that are defined by either a vertex of ∂R_1 and an edge of ∂R_2, or by an edge of ∂R_1 and a vertex of ∂R_2. Because grazing contact with the terrain is permitted, inter-visibility is blocked if and only if there is a triangle $t \in S$ for which $t \cap below(T) \neq \emptyset$. The proof of the following lemma is straightforward and is therefore omitted.

Lemma 2 *Given a terrain T and an arbitrary triangle t with vertices in $V(T)$. The intersection $t \cap below(T)$ is non-empty if and only if one of the following two situations occurs:*

(i) a vertex of T lies strictly above t, or
(ii) an edge of T lies strictly above an edge of t.

4. Algorithm and data structures

Now that we know what to compute, how can we compute it efficiently? The terrain has $O(n)$ vertices and triangles. The regions R_1 and R_2 are generally smaller, so we say they have $O(m)$ complexity. In the remainder of this paper we use $O^*(f(n,m))$ as a shorthand for a bound $O(f(n,m) \cdot n^\epsilon)$, where $\epsilon > 0$ is an arbitrarily small constant. The $O^*(..)$-notation supresses polylogarithmic factors of n, and factors n^ϵ.

4.1. *Facing the correct way*

For each region R_i, $i = 1,2$, we compute $\bigcap_{t \in R_i} HS(t)$, which is the intersection of $O(m)$ half-spaces. Because half-spaces are convex, this intersection can be computed in $O(m \log m)$ time [5]. Next, we check for all $O(m)$ vertices of the other region whether they are contained in this intersection. We preprocess the convex volume for point location in $O(m \log m)$ time and then query with $O(m)$ points in $O(\log m)$ time per query. Concluding, we can check if two $O(m)$ regions on the terrain are facing each other in $O(m \log m)$ time using $O(m)$ storage.

4.2. *Visibility blocked by the terrain*

We have to check for intersections between a set of $O(n)$ terrain triangles, and the elements of S, a set of $O(m^2)$ triangles between the edges and vertices of ∂R_1 and ∂R_2. We check all triangles in S for intersection with the terrain. In a brute-force algorithm, this leads to $O(nm^2)$ time. But can we do better?

By Lemma 2, we have to check two things to decide whether a triangle t from S intersects $\mathcal{T}$: either a vertex of T lies above t, or an edge of T lies above an edge of t. We discuss these two situations next.

4.2.1. *Terrain vertices*

To reduce the time complexity, it is necessary to take a different look at the geometry. For a given vertex v of the terrain, we want to check whether there is any triangle in S that lies strictly below v. Obviously, we want to limit the number of triangles to check in height against v.

If v lies above a triangle t in S, then the projection of v on the xy-plane lies in the projection of t. S contains $O(m^2)$ triangles that possibly overlap each other; we want to retrieve exactly those triangles $S(v) \subseteq S$ that contain the projection of v in their projections. Then the remaining question is: does v lie on or below all triangles of $S(v)$?

This question can also be simplified by looking at the geometry. If a point p in 3D lies above a set of triangles $S(v)$, it lies in the polyhedron $\bigcup_{t \in S(v)} HS(t)$. These geometric observations lead us to the data structure described next.

To find the triangles from S that contain a given point in their projections on the xy-plane, we construct a partition tree of $O(m^2)$ size [5]. Given a query point p, it returns $O^*(m)$ canonical subsets of triangles from S that contain p in their projection. We give every node μ in the partition tree an associated data structure of linear complexity in the cardinality of the canonical subset $c(\mu)$. This structure is used to determine whether the query point lies in the intersection of the half-spaces induced by all triangles in $c(\mu)$. This query can be answered in $O(\log m)$ time after $O(m \log m)$ preprocessing time and with linear storage, by using the Dobkin-Kirkpatrick hierarchy [7,9].

The construction time for a partition tree of size $O(m^2)$ is $O(m^{2+\epsilon})$ for any constant $\epsilon > 0$ [5]. Fortunately, the space required for the partition tree's associated structure does not increase the total storage space much, in particular, nothing at all in $O^*(..)$-notation. The total data structure requires $O(m^2 \log m)$ space. The construction time is $O^*(m^2)$ and the total query time is n times $O^*(m)$, which is $O^*(mn)$ [5].

4.2.2. *Terrain edges*

The second situation that must be tested to discover whether the terrain blocks visibility is if a terrain edge lies above an edge of a triangle from S. We project the terrain edges onto the xy-plane and for every edge from S, we want to find the terrain edges whose projections intersect the projection of the query edge. We can treat these terrain edges as full lines in 3D, and the same is true for the query edge. The objective is to find out whether the line supporting the query edge lies above all lines supporting the selected terrain edges.

The data structure we use stores the projections of the terrain edges in a cutting tree of size $O(n^2)$

[5]. We perform m^2 queries on this tree with edges from S, each taking $O(\log n)$ query time. The query returns all intersecting terrain edges in $O(\log n)$ canonical subsets. We give each node μ in the cutting tree an associated structure of size $O(n^{2+\epsilon})$ for the canonical subset $c(\mu)$ of edges as the supporting lines in space [3]. Consequently, we can decide whether a query line lies above all lines in the canonical subset in $O(\log n)$ query time.

Because the storage required for the cutting tree dominates the storage of the associated structure, the total data structure requires $O(n^{2+\epsilon})$ storage and can be constructed in $O^*(n^2)$ time. The total query time is $O(m^2 \log n)$, or $O^*(m^2)$, and thus the total time complexity to determine if any terrain edge is above any edge of S is $O^*(n^2 + m^2)$.

4.3. *Total time and space complexity*

In the previous section, we investigated the running times and storage space needed to determine whether two regions in a terrain completely see each other. We now summarize the time and storage complexities for the partition tree with its associated structures (PT) and the cutting tree with its associated structures (CT).

	Preprocessing	Total queries	Storage
PT	$O^*(m^2)$	$O^*(mn)$	$O(m^2)$
CT	$O^*(n^2)$	$O^*(m^2)$	$O^*(n^2)$

Because m is in the worst case $O(n)$, we have a total time complexity of $O^*(n^2)$ which is a considerable improvement over the $O(n^3)$ time of a brute-force algorithm.

5. Concluding remarks

We developed an algorithm to determine in $O^*(n^2)$ time whether two regions in a terrain are strongly inter-visible. The algorithm uses as data structures a partition tree and a cutting tree, both with associated structures, leading to a total of $O^*(n^2)$ storage.

Another natural problem would be determining weak inter-visibility between two regions, which means that every point in one region sees at least one point in the other region. Because weak visibility has less constraints than strong visibility, it is more difficult to compute. We are currently working on algorithms and data structures for this problem.

References

[1] P. K. Agarwal and M. Sharir. On the number of views of polyhedral terrains. In *Proc. 5th Canad. Conf. Comput. Geom.*, pages 55–60, 1993.

[2] M. Bern, D. Dobkin, D. Eppstein, and R. Grossman. Visibility with a moving point of view. *Algorithmica*, 11:360–378, 1994.

[3] B. Chazelle, H. Edelsbrunner, L. J. Guibas, M. Sharir, and J. Stolfi. Lines in space: Combinatorics and algorithms. *Algorithmica*, 15:428–447, 1996.

[4] R. Cole and M. Sharir. Visibility problems for polyhedral terrains. *J. Symbolic Comput.*, 7:11–30, 1989.

[5] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, Germany, 2nd edition, 2000.

[6] L. De Floriani and P. Magillo. Intervisibility on terrains. In P.A.Longley, M.F.Goodchild, D.J.Maguire, and D.W.Rhind, editors, *Geographic Information Systems: Principles, Techniques, Managament and Applications*, chapter 38, pages 543–556. John Wiley & Sons, 1999.

[7] D. P. Dobkin and D. G. Kirkpatrick. Determining the separation of preprocessed polyhedra – a unified approach. In *Proc. 17th Internat. Colloq. Automata Lang. Program.*, volume 443 of *Lecture Notes Comput. Sci.*, pages 400–413. Springer-Verlag, 1990.

[8] P. Magillo and L. De Floriani. Computing visibility maps on hierarchical terrain models. In *Proc. 2nd ACM Workshop on Advances in GIS*, 1994.

[9] D. Mount. Geometric intersection. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 33, pages 615–632. CRC Press LLC, Boca Raton, FL, 1997.

[10] G. Nagy. Terrain visibility. Technical Report, Rensselaer Polytech. Inst., Troy, NY, 1982.

[11] M. Sharir. The shortest watchtower and related problems for polyhedral terrains. *Inform. Process. Lett.*, 29(5):265–270, 1988.

[12] B. Zhu. Improved algorithms for computing the shortest watchtower of polyhedral terrains. In *Proc. 4th Canad. Conf. Comput. Geom.*, pages 286–291, 1992.

Guarding Art Galleries by Guarding Witnesses

Kyung-Yong Chwa [a] Byung-Cheol Jo [b] Christian Knauer [c] Esther Moet [d]
René van Oostrum [d] Chan-Su Shin [e]

[a] *Department of Computer Science, KAIST, Daejon, Korea. Email: kychwa@tclab.kaist.ac.kr*
[b] *Taff System, Co. Ltd., Seoul, Korea. Email: mrjo@taff.co.kr*
[c] *Freie Universität Berlin, Takustraße 9, D-14195 Berlin, Germany. Email: knauer@inf.fu-berlin.de*
[d] *Institute of Information & Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands. Email: {esther,rene}@cs.uu.nl*
[e] *School of Electronics and Information Engineering, Hankuk University of Foreign Studies, Yongin, Korea. Email: cssin@hufs.ac.kr*

Abstract

Let P be a simple polygon. We define a *witness set* W to be a set of points such that if any (prospective) guard set G guards W, then it is guaranteed that G guards P. We show that not all polygons admit a finite witness set. If a finite minimal witness set exists, then it cannot contain any witness in the interior of P; all witnesses must lie on the boundary of P, and there can be at most one witness in the interior of any edge. We give an algorithm to compute a minimal witness set for P in $O(n^2 \log n)$ time, if such a set exists, or to report the non-existence within the same time bounds. We also outline an algorithm that uses a witness set for P to test whether a (prospective) guard set sees all points in P.

1. Introduction

Approximately seven years ago, Joseph Mitchell posed the *Witness Problem* to Tae-Cheon Yang during a research visit of the latter: "Given a polygon P, does it admit a *witness set*, i.e., a set of objects in P such that any (prospective) guard set that guards the witnesses is guaranteed to guard the whole polygon?"

In this paper we consider *point witnesses* that are allowed to lie anywhere in the interior or on the boundary of the polygon. We want to determine for a given polygon P whether a finite witness set exists, and if this is the case, to compute a *minimal* witness set.

A preliminary full version of this paper is available as technical report [3]. Due to space limitations, we omitted several lemmas of minor importance and the proofs of the remaining lemmas in this abstract.

2. Preliminaries

Throughout this paper, P denotes a simple polygon with n vertices $V(P) = \{v_0, v_1, \ldots, v_{n-1}\}$; we assume that the vertices are ordered in counterclockwise direction. The edges of P are denoted with $E(P) = \{e_0, e_1, \ldots, e_{n-1}\}$, with $e_i = (v_i, v_{i+1 \bmod n})$. We consider an edge e_i to be the closed line segment between its incident vertices, and P to be a closed subset of $\mathbb{E}^2$.

A point p in P *sees* a point q in P if the line segment pq is contained in P. Since polygons are closed regions, the line-of-sight pq is not blocked by grazing contact with the boundary of P; this definition of visibility is commonly used in the Art Gallery literature [7].

We say that a point p in P *sees past* a reflex vertex v of P if p sees v, and the edges incident to v do not lie on different sides of the line through p and v (i.e., one of the edges may lie on this line).

Let p be a point in P. The *visibility polygon* of p is the set of points in P that are visible to p. We denote the visibility polygon by $\mathrm{VP}(p)$. The *visibility kernel* of a point p is the kernel of its visibility polygon and is denoted by $\mathrm{VK}(p)$.

Definition 1 *A* witness set *for a polygon P is a point set W in P for which the following holds: if, for any arbitrary set of points G in P, each element of W is visible from at least one point in G, then* every *point in P is visible from at least one point in G.*

The following theorem states the necessary and sufficient conditions on witness sets:

Theorem 2 *A point set W is a witness set for a polygon P if and only if the union of the visibility kernels of the elements of W covers P completely.*

We also apply the concept of witnesses to individual points. For two points p and q in a polygon P, we say that p is a witness for q (or alternatively, that p witnesses q), if any point that sees p also sees q. The following lemma is analogous to Theorem 2:

Lemma 3 *If p and q are points in a polygon P, then p witnesses q if and only if q lies in* $\mathrm{VK}(p)$.

The following lemma shows that witnessing is transitive:

Lemma 4 *Let P be a polygon, and let p, q, and r be points in P. If p witnesses q and q witnesses r, then p witnesses r.*

This leads to the notion of *minimal witness sets*:

Definition 5 *Let P be a polygon and let W be a witness set for P. W is called a* minimal witness set *for P if, for any $w \in W$, $W \setminus \{w\}$ is* not *a witness set for P.*

Lemma 6 *Let P be a polygon, and let W be a witness set for P. W is a minimal witness set for P if and only if for any $w \in W$, w does not lie in $\mathrm{VK}(w')$ for any $w' \in W, w' \neq w$.*

Lemma 7 *Let P be a polygon. If W is a witness set for P, then (i) there exists a subset $W' \subseteq W$ such that W' is a minimal witness set for P, and (ii) for any superset $W'' \supseteq W$, W'' is a witness set for P.*

Observe that not all polygons are witnessable with a finite witness set; see Figure 1. The polygon on the left is witnessable by three witnesses (the black dots), but the polygon on the right needs an infinite number of witnesses. The visibility kernels of the witnesses indicated at four of the vertices of the polygon do not cover the complete polygon.

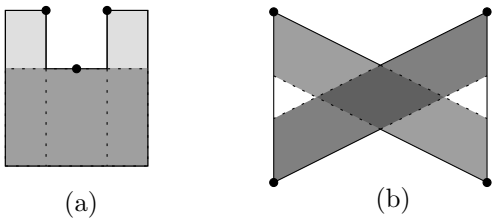


Fig. 1. The polygon on the left is witnessable with three witnesses, while the polygon on the right needs an infinite number of witnesses.

Adding witnesses to the remaining vertices does not help, as these vertices are already witnessed and witnessing is transitive. It turns out that we would need to cover both unwitnessed segments on the boundary of the polygon completely with witnesses to get an (infinite) witness set for this polygon.

3. Visibility kernels

In this section we study several properties of visibility kernels, that are used in the next section to establish our main results on finite witness sets.

Let P be a polygon with n vertices and edges, as defined in Section 2. It is well-known that the kernel of a polygon P is the intersection of the positive halfspaces of its edges; when this kernel is non-empty, the polygon is said to be *star-shaped*. The visibility polygon $\mathrm{VP}(p)$ of a point p in P is star-shaped by definition (the kernel contains at least p). However, there is an alternative way of describing $\mathrm{VK}(p)$ that turns out to be useful.

The edges of the visibility polygon $\mathrm{VP}(p)$ can be classified into two groups (see Figure 2):

(i) An edge e of $\mathrm{VP}(p)$ coincides with the part of an edge e' of P that is visible from p.

(ii) An edge e of $\mathrm{VP}(p)$ is induced by the directed line $\ell(p, v)$ through p and a reflex vertex v of P such that p sees past v.

The visibility kernel $\mathrm{VK}(p)$ is the intersection of the closure of the positive half-spaces induced by the lines through all edges of $\mathrm{VP}(p)$. The reader may wonder why we introduce this seemingly complicated alternative representation of the edges of $\mathrm{VP}(p)$. The reason is that for any p, there may be many edges in group (ii), but at most two of these contribute to $\mathrm{VK}(p)$. This helps us to reduce the complexity of the data structures involved in com-

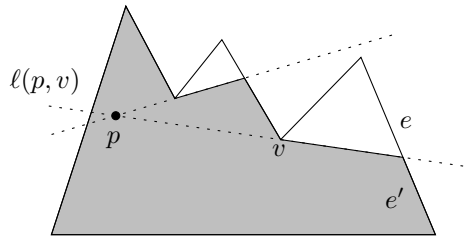


Fig. 2. Two types of edges of $\mathrm{VP}(P)$.

puting the union of a set of visibility kernels; see Section 5.

We conclude this section with a property of visibility kernels that is of use in the remainder of this paper.

Lemma 8 *If a point p in a polygon P sees past a reflex vertex $v \in V(P)$, then p lies on the boundary of* $\mathrm{VK}(p)$.

4. Finite witness sets

We would like to determine for a given simple polygon P whether a finite witness set for P exists, and if so, to compute such a set.

We have seen that a point p witnesses a point q if q lies in $\mathrm{VK}(p)$. This means that for a witness set W, the union $\bigcup_{w \in W} \mathrm{VK}(w)$ must cover the whole polygon p.

For a polygon P and a set W of points in P, let $\mathcal{A}(W)$ be the arrangement in P induced by the supporting lines of the line segments of type (i) and of those of type (ii) that contribute to $\mathrm{VK}(w)$, for every witness $w \in W$.

For any cell c of $\mathcal{A}(W)$ and any point $w \in W$, c lies either completely inside or completely outside $\mathrm{VK}(w)$. This means that W is a witness set for P if and only if every cell of $\mathcal{A}(W)$ is contained in $\mathrm{VK}(w)$ for at least one $w \in W$.

We denote the cardinality of W by m. Because for every $w \in W$, there are at most two vertices in group (ii) that contribute to $\mathrm{VK}(w)$, there are in total at most $n + 2m$ line segments that define $\mathcal{A}(W)$, and therefore the complexity of $\mathcal{A}(W)$ is $O((n+m)^2)$. We discuss how to test the cells of $\mathcal{A}(W)$ on containment in visibility kernels in Section 5.

Via several lemmas that are derived from Lemmas 4 and 8, we arrive at the following lemma:

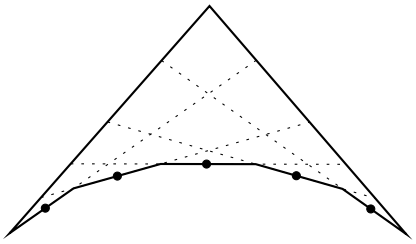


Fig. 3. For any n there is a polygon with n vertices that is witnessable with no less than $n-2$ witnesses. Witnesses in the example are indicated with black dots.

Lemma 9 *Let P be a simple polygon. If W is a finite minimal witness set for P, then no element of W lies in $\mathrm{int}(P)$.*

Note that a convex polygon can be witnessed by a single point in its interior. However, such a one-element witness set is not minimal, as the empty set is also a witness set for any convex polygon.

Given the above lemma, we only need to concentrate on witnesses that lie on the boundary of P. Analyzing the possible configurations of witness sets, we arrive at the following theorem:

Theorem 10 *Let P be a simple polygon with n edges. If a finite minimal witness set W for P exists, then all witnesses $w \in W$ lie on the boundary of P. Each edge has at most one witness $w \in W$ in its interior. If an edge has one or two witnesses on its incident vertices, then there cannot be any witness in the interior of the edge. Finally, for each $n \geq 4$ there is a polygon that needs no less than $n-2$ witnesses to be witnessed.*

The lowerbound construction is given in Figure 3.

5. Algorithms

In this section we outline an algorithm that computes a minimal finite witness set W for a simple polygon P, if such a set exists, or reports the non-existence of such a set otherwise. We also outline an algorithm that uses a witness set W for P to test whether a set of points G in P guards the whole polygon.

The algorithm to compute a minimal witness set W for a given simple polygon P with n vertices works as follows:

- First, we place witnesses at candidate positions. We place a witness at every vertex of P, and one halfway each edge of P. This step runs in $O(n)$ time.
- Next, using $\mathcal{A}(W)$, we test whether W' is a witness set for P with a sweepline approach. If it is, we extract a minimal witness set W from W' in the next step; otherwise, we report that no finite witness set for P exists. This step takes $O(n^2 \log n)$ time.
- We extract a minimal witness set W by repeatedly removing an unnecessary witness, i.e. witnesses that are witnessed by another witness in W'. This step takes $O(n^2 \log n)$ time.

This leads to the following theorem:

Theorem 11 *Let P be a simple polygon with n vertices. If a finite witness set for P exists, a finite minimal such set W can be computed in $O(n^2 \log n)$ time. Otherwise, if no finite witness set for P exists, than this can be reported in the same running time.*

Next, we need an algorithm for testing whether a set G of g guards in a polygon P together see the whole polygon. A straightforward check, without using witnesses, can be performed in $O((g^2 + gn \log g) \log(g + n))$ time [2,4].

Can we do better if a witness set W of size m for P is given? We test for each witness whether it can be seen by a guard by performing (at most) g ray shooting queries, or $O(gm)$ queries in total. P can be preprocessed for ray shooting in $O(n)$, after which a query takes $O(\log n)$ time [5]. So the total preprocessing time, including the computation of W, becomes $O(n^2 \log n)$ time, and the query time is $O(gm \log n)$. Note that in the worst case $m = \theta(n)$. This query time is faster then the straightforward approach described above, but not necessarily very much (how much precisely depends on the parameters g and m). If m (the number of witnesses) is small and g (the number of guards) is large, then the gain is big.

6. Concluding remarks

We showed that if a polygon P admits a finite witness set, then any minimal witness set W for P has no witnesses in the interior of P, there is at most one witness in the interior of each edge of P. If an edge has one or two witnesses on its incident vertices, then there cannot be a witness in the interior of this edge. It follows that any minimal witness set for P has at most n elements. Furthermore, for any $n \geq 4$, there is a polygon for which the *minimum* size witness set has $n - 2$ witnesses. A minimal finite witness set for P can be computed in $O(n^2 \log n)$ time, if it exists.

It remains open whether the problem of finding a *minimum* size witness set for a given polygon is computable. It is well-known that the problem of finding a minimum size guard set is NP-complete [1,6]. We conjecture, however, that finding a minimum size witness set is computable in polynomial time, and we are currently working towards turning our conjecture into a theorem.

Another interesting direction for further research is to consider other types of witnesses, such as (a subset of) the edges of the polygon. We believe that we can extend our current lemmas, theorems, and algorithms to test whether a polygon is witnessable by an minimal infinite witness set, where all the witnesses lie on the boundary of the polygon.

References

[1] Alok Aggarwal. *The art gallery problem: Its variations, applications, and algorithmic aspects.* PhD thesis, Dept. of Comput. Sci., Johns Hopkins University, Baltimore, MD, 1984.

[2] O. Cheong and R. van Oostrum. The visibility region of points in a simple polygon. In *Proc. 11th Canad. Comf. Comp. Geom.*, pages 87–90, 1999.

[3] Kyung-Yong Chwa, Byung-Cheol Jo, Christian Knauer, Esther Moet, René van Oostrum, and Chan-Su Shin. Guarding art galleries by guarding witnesses. Report UU-CS-2003-044, Institute for Information & Computing Sciences, Utrecht University, Utrecht, Netherlands, 2003.

[4] L. Gewali, A. Meng, Joseph S. B. Mitchell, and S. Ntafos. Path planning in $0/1/\infty$ weighted regions with applications. *ORSA J. Comput.*, 2(3):253–272, 1990.

[5] J. Hershberger and Subhash Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. *J. Algorithms*, 18:403–431, 1995.

[6] D. Lee and A. Lin. Computational complexity of art gallery problems. *IEEE Trans. Inform. Theory*, 32(2):276–282, 1986.

[7] J. O'Rourke. *Art Gallery Theorems and Algorithms.* The International Series of Monographs on Computer Science. Oxford University Press, New York, NY, 1987.

Verification of Partitions of 2d and 3d Objects

Leonidas Palios

Department of Computer Science, University of Ioannina, 45110 Ioannina, Greece

Abstract

We consider the problems of deciding whether a given collection of polygons (polyhedra resp.) forms (i) a partition or (ii) a cell complex decomposition of a given polygon (polyhedron resp.). We describe simple $O(n \log n)$-time and $O(n)$-space algorithms for these problems, where n is the total description size of the input. If, in the input, vertices are referenced by means of indices to an array of distinct vertices, then our cell complex decomposition verification algorithms run in $O(n)$ time.

1. Introduction

In recent years, there has been growing interest in algorithms that enable us to check the output of a program. This issue has been considered from different viewpoints, and the research yielded results ranging from characterizations of problems that are checkable [2] to special-purpose checkers. In particular, in computational geometry, this issue proves to be crucial as the newer and more efficient algorithms are more and more complicated. Usually, the authors of an algorithm for computing some geometric object describe properties of the object which can be used to verify the correctness of the computation (see e.g. [1] and [4] for 2d and 3d triangulations of point sets). Often, however, more machinery is needed.

The first verification algorithms ("checkers") were provided by Mehlhorn *et al.* who defined the properties that a checker should have (correctness, simplicity, efficiency) and described checkers for convex polyhedra and convex hulls [5]; they have also studied checkers for trapezoidal decompositions, planar point location structures for trapezoidal decompositions, and Voronoi and Delaunay diagrams. Devillers *et al.* extended the notion of checkers and provided checkers for convex polytopes in two and higher dimensions, and for various types of planar subdivisions [3]. Their algorithms run in linear time, assuming that the input is a 2d or a 3d ordered geometric graph.

Email address: palios@cs.uoi.gr (Leonidas Palios).

In this paper, we present simple and efficient verification algorithms for partitions and cell complex decompositions of simple polygons and polyhedra. The algorithms rely on appropriately matching the edges of the 2d decompositions and the facets of the 3d decompositions, and run in $O(n \log n)$ time using $O(n)$ space, where n is the total size of the input. If, in the input, vertices are referenced by means of indices to an array of distinct vertices, then our cell complex decomposition verification algorithms run in optimal $O(n)$ time.

Note: In the following, a polygon or a polyhedron is understood to be a *simple* one.

2. The Two-dimensional Case

We assume that each polygon in the input is described by the sequence of its vertices along its boundary, where each vertex is given by its coordinates.

2.1. *Verification of a partition of a polygon*

Lemma 1 *Let P and $\mathcal{C}$ be a polygon and a collection of polygons respectively. Additionally, let $E_I = \{e - \partial P \mid e \text{ is an edge of a polygon in } \mathcal{C}\}$ and $E_B = \{e \cap \partial P \mid e \text{ is an edge of a polygon in } \mathcal{C}\}$. Then, the collection $\mathcal{C}$ forms a partition of P if and only if*

(i) the set E_I of (parts of) edges of the polygons in $\mathcal{C}$ can be partitioned into sets S and S' such that

- $\bigcup_{s \in S} closure(s) = \bigcup_{s \in S'} closure(s)$,

- *the closures of the segments in S and S' define a partition of $\bigcup_{s\in S}$ closure(s), and*
- *for each (part of) edge e in S (resp. S'), if $S'[e]$ (resp. $S[e]$) is the set of segments of S' (resp. S) that are collinear with and intersect e, then, locally around e, the interior of the polygon in $\mathcal{C}$ with edge e and the interiors of the polygons in $\mathcal{C}$ whose edges contributed the elements of $S'[e]$ (resp. $S[e]$) lie on opposite sides of the line supporting e;*

(ii) the closures of the segments in E_B form a partition of the boundary ∂P of P, and for each (part of) edge e in E_B, locally around e, the interior of the polygon in $\mathcal{C}$ with e as an edge and the interior of P lie on the same side with respect to the line supporting e.

The partition verification algorithm applies Lemma 1. It uses an (initially empty) array A of size equal to the total number of edges of the polygons in $\mathcal{C}$ and of the polygon P.

2d Partition Verification Algorithm

1. Orient the boundary of polygon P and the boundaries of the polygons in $\mathcal{C}$ in a compatible fashion (e.g., the boundary is traversed in a ccw fashion).
2. For each polygon Q in $\mathcal{C}$ do
   for each edge $\overrightarrow{uv}$ of Q do
   if u is lex-smaller than v
   then add the entry $(u, v, +1)$ in A;
   else add the entry $(v, u, -1)$ in A;
3. For each edge $\overrightarrow{uv}$ of P do
   if u is lex-smaller than v
   then add the entry $(u, v, -1)$ in A;
   else add the entry $(v, u, +1)$ in A;
4. Sort the entries $(u_i, v_i, \pm 1)$ of the array A by slope of the line $u_i v_i$ and, in case of ties, lexicographically taking into account the coordinates of the vertices u_i and v_i.
5. For each slope separately, traverse the related entries of A verifying that those with "+1" and those with "−1" partition the same set of segments. If this terminates successfully, then the collection $\mathcal{C}$ of polygons defines a partition of the polygon P, otherwise it does not.

Time complexity. Steps 1, 2, and 3 can be completed in time linear in the total description size n of the input, while Step 4 takes $O(n \log n)$ time. Step 5 takes $O(n)$ time: for each slope, the sorting of the entries implies that each new entry with "+1" either is the extension of the latest entry with "+1" or starts a new segment in the union of the "+1"-entries (if not, $\mathcal{C}$ does not form a partition of P), and similarly for the "−1"-entries. The algorithm uses $O(n)$ space.

2.2. *Verification of a cell complex decomposition of a polygon*

Lemma 2 *Let P and $\mathcal{C}$ be a polygon and a collection of polygons respectively. Then, the collection $\mathcal{C}$ forms a cell complex decomposition of P if and only if the set of edges of the polygons in $\mathcal{C}$ can be partitioned into two sets E_I and E_B such that*

(i) for each edge e in E_I there exists exactly one other edge, say d, in E_I such that $e = d$ and, locally around e, the interiors of the polygons in $\mathcal{C}$ with edges e and d lie on opposite sides of the line supporting e;

(ii) the edges in E_B form a partition of the boundary ∂P of P and for each edge e in E_B, locally around e, the interior of the polygon in $\mathcal{C}$ with edge e and the interior of P lie on the same side with respect to the line supporting e.

The cell complex verification algorithm applies Lemma 2; it too uses an (initially empty) array A of size equal to the total number of edges of the polygons in $\mathcal{C}$.

2d Cell Complex Verification Algorithm

1. Collect all the vertices and assign to each one of them a distinct positive integer $id(\ )$.
2. Orient the boundary of polygon P and the boundaries of the polygons in $\mathcal{C}$ in a compatible fashion.
3. For each polygon Q in $\mathcal{C}$ do
   for each edge $\overrightarrow{uv}$ of Q do
   if u is lex-smaller than v
   then add $(id(u), id(v), +1)$ in A;
   else add $(id(v), id(u), -1)$ in A;
4. Sort the array A lexicographically.
5. Traverse the sorted array A deleting matching entries (i.e., $(k, k', -1)$ and $(k, k', +1)$), which now appear next to each other.
6. Collect the unmatched entries and by applying a depth-first traversal construct the graph that these entries form. If this graph is a closed path identical to the boundary of the polygon P, then the collection $\mathcal{C}$ of poly-

gons defines a cell complex decomposition of P, otherwise it does not.

Time complexity. If n is the total description size of the input, Step 1 can be executed in $O(n \log n)$ time by means of a balanced binary search tree to determine identical vertices. Steps 2 and 3 take $O(n)$ time; so does Step 4 (by using radix sorting), and Steps 5 and 6. Thus, the algorithm takes a total of $O(n \log n)$ time. The space required by the algorithm is $O(n)$.

Observe that all the steps of the above algorithm but Step 1 take $O(n)$ time. Moreover, if the input consists of the list V of distinct vertices of the polygon P and of the polygons in the collection $\mathcal{C}$, followed by the vertices of each polygon, where each vertex is referenced by its index to the list V, then we do not need to execute Step 1. Such a representation is commonly used, and is easily produced by partitioning programs. Then, it can be determined whether the collection $\mathcal{C}$ forms a cell complex decomposition of P in $O(n)$ time using $O(n)$ space.

3. The Three-dimensional Case

We assume that each polyhedron in the input is described by a list of its facets, each represented by the sequence of its vertices along its boundary, where each vertex is given by its coordinates.

Our verification algorithms rely on two lemmata similar to Lemma 1 and Lemma 2.

3.1. *Verification of a partition of a polyhedron*

Lemma 3 *Let P and $\mathcal{C}$ be a polyhedron and a collection of polyhedra respectively. Additionally, let $F_I = \{f - \partial P \mid f$ is a facet of a polyhedron in $\mathcal{C}\}$ and $F_B = \{f \cap \partial P \mid f$ is a facet of a polyhedron in $\mathcal{C}\}$. Then, the collection $\mathcal{C}$ forms a partition of P if and only if*

(i) the set F_I of (parts of) facets of the polyhedra in $\mathcal{C}$ can be partitioned into two sets S and S' such that the closures of the polygons in each of these sets defines a partition of $\bigcup_{s \in S} closure(s)$, and for each (part of) facet f in S (S' resp.), the interior of the polyhedron of $\mathcal{C}$ with facet f and the interiors of the polyhedra of $\mathcal{C}$ with facets the polygons of S' (S resp.) which intersect with f lie (locally around f) on opposite sides of the plane supporting f;

(ii) the closures of the polygons in F_B form a partition of the boundary ∂P of P, and for each (part of) facet f in F_B, locally around f, the interior of the polyhedron of $\mathcal{C}$ with facet f and the interior of P lie on the same side with respect to the plane supporting f.

In light of Lemma 3, we have:

3d Partition Verification Algorithm

1. Orient the facets of the polyhedron P and of the polyhedra in $\mathcal{C}$ in a compatible fashion (i.e., in counterclockwise order as seen from outside the polyhedron).
2. For each polyhedron Q in $\mathcal{C}$ do
   for each facet f of Q do
   $u \leftarrow$ the lex-smallest vertex of f;
   if f's boundary forms a left turn at u
   then add the entry $(f, +1)$ in A;
   else add the entry $(f, -1)$ in A;
3. For each facet f of P do
   $u \leftarrow$ the lex-smallest vertex of f;
   if f's boundary forms a left turn at u
   then add the entry $(f, -1)$ in A;
   else add the entry $(f, +1)$ in A;
4. Sort the entries $(f_i, \pm 1)$ of the array A by slope of the plane supporting f_i and, in case of ties, lexicographically on the second field of the entry.
5. For each slope separately, verify that the related facet records with "+1" and those with "−1" partition the same polygonal regions. If this matching terminates successfully, then the collection $\mathcal{C}$ defines a partition of the polyhedron P, otherwise it does not.

Time complexity. Steps 1, 2, and 3 can be completed in time linear in the total description size n of the input. Step 4 takes $O(n \log n)$ time. Step 5 can also be completed in $O(n \log n)$ time: for each different plane slope, collect the entries with "+1" as second field, and apply on the associated facets Steps 2 and 4 of the 2d partition verification algorithm (see Section 2.1); Step 5 of that algorithm is applied next, except that the difference of the union of the "+1" and the "−1" entries is computed and used to form a graph; the same procedure is applied to the entries of the array A with "−1" as second field; finally, the two resulting graphs are checked to see whether they are identi-

cal. Thus, the algorithm takes a total of $O(n \log n)$ time. The algorithm uses $O(n)$ space.

3.2. *Verification of a cell complex decomposition of a polyhedron*

Lemma 4 *Let P and $\mathcal{C}$ be a polyhedron and a collection of polyhedra respectively. Then, the collection $\mathcal{C}$ forms a cell complex decomposition of P if and only if the set of facets of the polyhedra in $\mathcal{C}$ can be partitioned into two sets F_I and F_B such that*

(i) for each facet f in F_I there exists exactly one other facet, say f', in F_I such that $f = f'$ and, locally around f, the interiors of the polyhedra in $\mathcal{C}$ with edges f and f' lie on opposite sides of the plane supporting f;

(ii) the edges in F_B form a partition of the boundary ∂P of P and for each facet f in F_B, locally around f, the interior of the polyhedron of $\mathcal{C}$ with facet f and the interior of P lie on the same side with respect to the plane supporting f.

The cell complex verification algorithm applies Lemma 4. It uses two auxiliary arrays A and B, which are initially empty; it works as follows:

3d Cell Complex Verification Algorithm

1. Collect all the vertices and assign to each one of them a distinct positive integer $id(\ )$.
2. Orient the facets of the polyhedron P and of the polyhedra in $\mathcal{C}$ in a compatible fashion.
3. For each polyhedron Q in $\mathcal{C}$ do
   for each facet f of Q do
   $a \leftarrow$ the lex-smallest vertex of f;
   $b \leftarrow$ the lex-largest vertex of f;
   $c \leftarrow$ f's vertex farthest away from the line $\overline{ab}$ (and lex-smallest, if ties);
   if f's boundary is directed $a \rightsquigarrow c \rightsquigarrow b$
   then add $((id(a), id(b), id(c)), +1)$ in A;
   else add $((id(a), id(b), id(c)), -1)$ in A;
4. Sort the array A lexicographically.
5. Traverse the sorted array A, verify that matched entries (which now appear next to each other) correspond to matching facets and delete them.
6. For each facet f whose entry in A has not been matched do
   for each edge $\overrightarrow{uv}$ of f do
   if u is lex-smaller than v
   then add $(id(u), id(v), +1, f)$ in B;
   else add $(id(v), id(u), -1, f)$ in B;
7. Sort the array B lexicographically.
8. If the entries in B do not appear in matching pairs (i.e., $(k, k', -1, f)$ and $(k, k', +1, f')$), then the collection $\mathcal{C}$ of polyhedra does not define a cell complex decomposition of the given polyhedron P.
9. For each matching pair $(id(u), id(v), -1, f)$ and $(id(u), id(v), +1, f')$ in B do
   if the two facets f, f' are not coplanar
   then make u and v adjacent;
10. Apply a depth-first traversal and check that the graph formed matches the edge skeleton of the polyhedron P. If it does, then the collection $\mathcal{C}$ of polyhedra defines a cell complex decomposition of P, otherwise it does not.

Time complexity. Step 1 takes $O(n \log n)$ time, as described in Section 2.2. Steps 2, 3, 5, and 6 take $O(n)$ time. Steps 4 and 7 can be completed in linear time as well by using radix sorting. Finally, Steps 8, 9, and 10 also take linear time. Thus, the algorithm requires a total of $O(n \log n)$ time. The space required by the algorithm is $O(n)$.

Similarly to the two-dimensional case, if the polyhedron P and the polyhedra in the collection $\mathcal{C}$ are represented by a list of facets, each described by a sequence of indices to an array of distinct vertices which indicates the order of the vertices around the boundary of the facet, then it can be determined whether the collection $\mathcal{C}$ forms a cell complex decomposition of P in time and space linear in the total description size of P and $\mathcal{C}$.

References

[1] D. Avis and H. El Gindy, Triangulating point sets in space, *Discrete and Computational Geometry* **2**, 99–111, 1987.

[2] M. Blum and S. Kannan, Designing programs that check their work, *Journal ACM* **42(1)**, 269–291, 1995.

[3] O. Devillers, G. Liotta, F.P. Preparata, and R. Tamassia, Checking the convexity of polytopes and the planarity of subdivisions, *Computational Geometry: Theory and Applications* **11(3-4)**, 187–208, 1998.

[4] H. Edelsbrunner, F. Preparata, and D. West, Tetrahedralizing point sets in 3 dimensions, *Journal of Symbolic Computation*, **10**, 335–347, 1990.

[5] K. Mehlhorn, S. Näher, T. Schilz, S. Schirra, M. Seel, R. Seidel, and C. Uhrig, Checking geometric programs or Verification of geometric structures, *Proc. 12th Symp. on Computational Geometry*, 159–165, 1996.

A Completion of Hypotheses Method for 3D-Geometry. 3D-Extensions of Ceva and Menelaus Theorems [1]

E. Roanes-Macías [a], E. Roanes-Lozano *,[a]

[a] *Dept. Algebra, Universidad Complutense de Madrid, Edificio "La Almudena", c/ Rector Royo Villanova s/n, 28040-Madrid, Spain*

Abstract

A method that automates hypotheses completion in 3D-Geometry is presented. It consists of three processes: defining the geometric objects in the configuration; determining the hypothesis conditions of the configuration (through a point-on-object declaration method); and applying an algebraic automatic theorem proving method to obtain and prove the sufficiency of complementary hypothesis conditions. To avoid as much as possible the appearance of rational expressions, projective coordinates are used (although affine and Euclidean problems can also be treated). A Maple implementation of the method has been used to extend to 3D classic 2D geometric theorems like Ceva's and Menelaus'.

Key words: 3D-Geometry, Simbolyc Computation, Automatic Theorem Proving

1. Brief Description of the Method

Hypotheses completion was already treated by Recio and Vélez [6]. The method presented in this paper automates hypotheses completion in 3D-Geometry. Let us give a brief description of its three processes.

1.1. *Defining the Geometric Objects in the Configuration*

Among the geometric objects in a configuration, some can be defined directly and others are determined through geometric operations (see Table 1). Other usual geometric objects included in the package (segment, midpoint, sphere, quadric,...) are omitted for the sake of space.

The desired configuration can be constructed through the adequate concatenation of these *elementary* commands. Note that in this Geometry not only the rule-and-compass *globally constructible* objects can be treated: those geometric objects such that any of their points can be constructed with rule-and-compass, can be treated too.

Projective coordinates are used. Command `intCoor` allows to substitute coordinates where rational expressions appear by the corresponding integer quaternions.

1.2. *Determining the hypothesis conditions of the configuration*

Hypothesis conditions are declared as membership relations between points and higher dimension geometric objects. To declare $P = [p_0, p_1, p_2, p_3]$ as a point on the object ϕ (being the equations of ϕ: $\phi_i(x_0, x_1, x_2, x_3) = 0$; $i = 1, ..., n$) is equivalent to impose that the *hypothesis conditions* $\phi_i(P_0, P_1, P_2, P_3) = 0$; $i = 1, ..., n$ are verified. Command `pointOnObject` takes care of adding these polynomials to a certain list, denoted $LREL$, where the *hypothesis polynomials* are stored, and to add the corresponding variables to the list VAR.

* Corresponding author

Email addresses: roanes@mat.ucm.es (E. Roanes-Macías), eroanes@mat.ucm.es (E. Roanes-Lozano).

[1] Partially supported by the research project TIC-2000-1368-C03-03 (MCyT, Spain).

Object	Input	Command	Output
initial point (free point)	four projective coordinates	`point`	list of 4 parameters
plane	three non-collinear points	`plane`	equation of the plane
line	two different points	`line`	list of equations of the line
point on line AB ($\overrightarrow{PB} = r \cdot \overrightarrow{PA}$)	two points (A, B) and a real number r	`rateOnLine`	list of coords. of point P
plane/line parallel to a given plane/line	one linear object and one point	`parallel`	equation(s) of the plane/line
plane/line perpendicular to a given line/plane	one linear object and one point	`perpendicular`	equation(s) of the plane/line
intersection of two objects (not necessarily linear)	two already defined objects	`intersection`	coords. of point(s) or equation(s) of linear objects or reduced list of eqs. (in GB sense)

Table 1
Geometric objects' definition

1.3. *Obtaining and Proving the Sufficiency of Complementary Hypothesis Conditions*

In most configuration geometric problems, the thesis is (or can be reduced to) a $P \in \phi$ membership condition (where P is a point and ϕ is a geometric object) or to a geometric relation among geometric objects in the configuration. In both cases the *thesis polynomial* admits a $\phi(P)$ form.

In case list $LREL$ is empty, to check that the thesis holds is equivalent to check that ϕ vanishes in P (i.e., that $\phi(P) = 0$). Command `isPlaced` applied to the pair (P, ϕ) takes care of performing all the corresponding computations.

In case list $LREL$ is not empty, to check that the thesis holds it is sufficient to check that ϕ can be expressed as an algebraic linear combination of the polynomials in list $LREL$, what can be effectively computed using Wu's techniques. A brief description of these automatic proving techniques can be found in [1], meanwhile a detailed description can be found, e.g., in [2,9]. These techniques were adapted to hypotheses completion in [5] and to geometric loci determining in [7]. The technique described in this paper is essentially that of [7], but has been adapted to the way hypothesis and thesis conditions are usually declared.

This process basically consists of two steps:

- to triangularize system $LREL$ w.r.t. the variables in list VAR, to obtain system $TRIP$
- to compute, starting with $\phi(P)$, the successive pseudo-remainders of dividing by the polynomials in $TRIP$ w.r.t. the variables in VAR, until the last pseudo-remainder (polynomial ω) is obtained.

That $\omega = 0$ is a *necessary condition* for the thesis to hold. Command `newHypot` of our package, applied to (P, ϕ), automatically computes ω.

But we would still have to check that $\omega = 0$ is a *sufficient condition* for the thesis $\phi(P) = 0$ to hold.

If a parametrization of $\omega = 0$ can be obtained, then we substitute in $\phi(x_0, x_1, x_2, x_3)$ the x_i by their corresponding parametric expressions. If the resulting polynomial vanishes, then condition $\omega = 0$ is also sufficient. Command `isPlaced` can take care of these computations.

If a parametrization of $\omega = 0$ can't be obtained, then ω is be added to list $LREL$, and the new variable appearing in ω but not in list VAR, is added to list VAR. The same process can be applied now,

and, if the last pseudo-remainder is 0, then condition $\omega = 0$ is also sufficient. Command `autProve` can take care of these computations.

2. 3D-Extension of Ceva and Menelaus Theorems

An application of the automatic theorem proving method described above is included as illustration afterwards. The goal is to determine conditions that make four points, lying on consecutive edge-lines of a tetrahedron, coplanary (see Figure 1). This problem was recently solved using synthetic techniques by H. Davis [3].

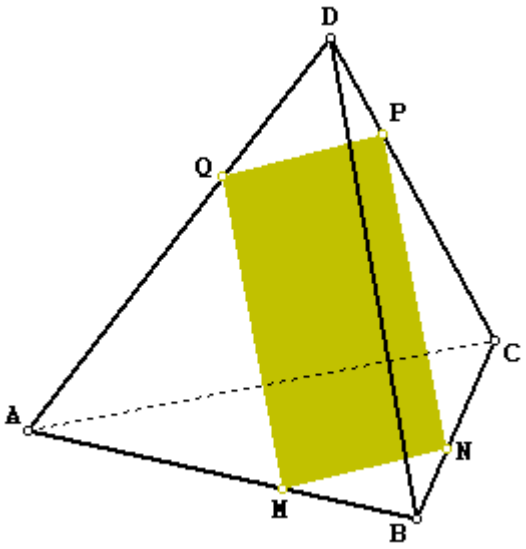


Fig. 1. Extending to 3D Ceva and Menelaus theorems

We can assume that the vertices are $A(1,0,0,0)$, $B(1,1,0,0)$, $C(1,\gamma_1,\gamma_2,0)$, $D(1,\delta_1,\delta_2,\delta_3)$ without any lack of generality (these points can be defined using command `point`). Given $m,n,p,q \in \mathbb{R}\cup\{\infty\}$, let M,N,P,Q be the points lying on the edge-lines AB, BC, CD, DA (respectively), and satisfying

$$\overrightarrow{MB} = m\cdot\overrightarrow{MA} \quad ; \quad \overrightarrow{NC} = n\cdot\overrightarrow{NB}$$
$$\overrightarrow{PD} = p\cdot\overrightarrow{PC} \quad ; \quad \overrightarrow{QA} = q\cdot\overrightarrow{QD}$$

(they can be defined using command `rateOnLine`). Then plane MNP can be defined (using command `plane`).

As detailed above, applying command `newHypot` to the pair (Q, MNP), a necessary condition for Q to lie on plane MNP (i.e., for M,N,P,Q to be coplanary): $-\gamma_2\cdot\delta_3\cdot(-1+m\cdot n\cdot p\cdot q)=0$, is obtained. As A,B,C,D are non-coplanary points, and consequently, $\gamma_2\neq 0\neq\delta_3$, what implies: $m\cdot n\cdot p\cdot q = 1$. To verify that is a sufficient condition, Q is particularized for $q = 1/(m\cdot n\cdot p)$, and applying command `isPlaced` to the pair (Q, MNP), 0 is obtained, what confirms that Q belongs to plane MNP. This leads to the following:

Theorem 1 *Points M,N,P,Q, lying on the oriented consecutive edge-lines AB, BC, CD, DA of tetrahedron $ABCD$ (respectively), are coplanary, if and only if:*

$$(MB/MA)\cdot(NC/NB)\cdot(PD/PC)\cdot(QA/QD) = 1$$

Observe that the points M,N,P,Q do lie on the consecutive oriented edge-lines AB, BC, CD, DA, but they can lie outside the edge-segments, and therefore this result doesn't only generalizes Ceva theorem, but also Menelaus theorem.

3. Comparison with Other Methods

As the automatic theorem proving technique used in this work is based on Wu's algorithm, it is of a lower computational complexity than those techniques based on the use of Groebner bases.

Comparing this method with others based on Wu's techniques, the main difference is the way the geometric objects of the configuration are defined and the way the hypotheses conditions are declared. In the method presented here the geometric objects and the hypotheses conditions are obtained in a natural way, following the geometric algorithm that generates the configuration, instead of translating into algebraic expressions the geometric relations that determine them (what is usually the case).

That happens, for instance, in Simson-Steiner-Guzmán theorem 3D-extension [4]. The goal is to determine the conditions so that the projections (in prefixed directions) of a point on the faces of a tetrahedron are coplanary. This problem was developed in [7], translating into algebraic expressions the geometric relations. Now it has been developed using the method detailed in section 1, in a more comfortable and faster way.

Other advantage of the method proposed in Section 1 is the simple way in which parameters and variables are distinguished (what is not straightforward in other approaches). With this method the parameters are the non-numeric coordinates of the initial points (that are preserved along all subsequent calculations), meanwhile the variables are the coordinates of the point-on-object objects defined using `pointOnObject` command.

Another advantage of the method proposed in Section 1 is the possibility to develop the geometric algorithm of the configuration using a Dynamic Geometry System, and to translate it to a Computer Algebra System syntax (interpreting it using the package considered here), as already done in 2D [8]. We plan to implement it in the near future.

4. Conclusions

The hypotheses completion in 3D-Geometry method described is convenient and efficient. It allows the user to obtain automatically the equations in the configuration, the hypothesis conditions obtained directly in the configuration and the complementary hypothesis conditions that have to be added for the thesis condition to hold.

References

[1] D. Cox, J. Little and D. O'Shea, *Ideals, Varieties, and Algorithms* (Springer, New York, 1991).

[2] S. C. Chou, *Mechanical Geometry Theorem Proving* (Reidel, Dordrecht, 1988).

[3] H. Davis, Menelaus and Ceva Theorems and its many applications, *http://hamiltonious.virtualave.negt/essays/othe/finalpaper4.htm*

[4] M. de Guzmán, An Extension of the Wallace-Simson Theorem: Projecting in Arbitrary Directions, *Mathematical Monthly* **106/6** (1999) 574–580.

[5] D. Kapur and J.L. Mundy, Wu's method and its application to perspective viewing, in: D. Kapur, J.L. Mundy, eds., *Geometric Reasoning* (MIT Press, Cambridge MA, 1989) 15–36.

[6] T. Recio and M. P. Vélez, Automatic Discovery of Theorems in Elementary Geometry, *Journal of Automated Reasoning* **23** (1999) 63–82.

[7] E. Roanes-Macías and E. Roanes-Lozano, Automatic determination of geometric loci, in: J. A. Cambell and E. Roanes-Lozano, eds., *Artificial Intelligence and Symbolic Computation*. (Springer's Lecture Notes in Artificial Intelligenge no. 1930, Berlin, 2000) 157–173.

[8] E. Roanes-Lozano, E. Roanes-Macías and M. Villar, A Bridge Between Dynamic Geometry and Computer Algebra, *Mathematical and Computer Modelling* **37/9-10** (2003) 1005–1028.

[9] W. T. Wu, *Mechanical Theorem Proving in Geometries* (Springer-Verlag's Text and Monographs in Symbolic Computation, Wien, 1994).

Computing the Fréchet Distance between Piecewise Smooth Curves ⋆

Günter Rote

Freie Universität Berlin, Institut für Informatik, Takustraße 9, 14195 Berlin, Germany, rote@inf.fu-berlin.de.

Abstract

We consider the Fréchet distance between two curves which are given as a sequence of $m+n$ curved pieces. If these pieces are sufficiently well-behaved, we can compute the Fréchet distance in $O(mn\log(mn))$ time. The decision version of the problem can be solved in $O(mn)$ time.

1. Introduction

The Fréchet distance is a distance measure between curves.

Definition 1 (Fréchet distance) *Let $f\colon I=[l_I,r_I]\to\mathbb{R}^2$ and $g\colon J=[l_J,r_J]\to\mathbb{R}^2$ be two planar curves, and let $\|\cdot\|$ denote the Euclidean norm. Then the* Fréchet distance $\delta_F(f,g)$ *is defined as*

$$\delta_F(f,g) := \inf_{\substack{\alpha\colon [0,1]\to I\\ \beta\colon [0,1]\to J}} \max_{t\in[0,1]} \|f(\alpha(t))-g(\beta(t))\|.$$

where α and β range over continuous and non-decreasing reparameterizations with $\alpha(0)=l_I$, $\alpha(1)=r_I$, $\beta(0)=l_J$, $\beta(1)=r_J$.

In contrast to other common distance measures like the Hausdorff distance, the Fréchet distance respects the one-dimensional structure of the curves and doesn't just treat them as a point set.

The study of the Fréchet distance from a computational point of view has been initiated by Alt and Godau [2]. The *decision problem* is the problem to decide, for a given ε, whether the Fréchet distance between two curves is at most ε.

Alt and Godau [2] treated the case of two polygonal curves. For two curves of m and n pieces, respectively, they showed how to solve the decision problem in $O(mn)$ time and the optimization problem in $O(mn\log(mn))$ time. Some related problems have also been considered, like minimizing the Fréchet distance under translations [3], or a generalized Fréchet distance between a curve and a *graph* [1]. In all cases, however, the objects are piecewise linear.

In this paper, we explore the Fréchet distance between more general curves. We assume that each input curve is given as a sequence of smooth curve pieces that are"sufficiently well-behaved", such as circular arcs, parabolic arcs, or some class of spline curves. Our algorithm will perform certain operations on these curves, like intersecting them with a circle.

We will show that the *combinatorial complexity*, i. e., the number of steps, for solving the decision problem is not larger than for polygonal paths, $O(mn)$. The complexity of the individual operations (the *algebraic complexity*) depends of course on the nature of the curves. Under the stronger assumption that the curves consist of algebraic pieces whose degree is bounded by a constant, we can solve the optimization problem in $O(mn\log(mn))$, thus matching the running time for the polygonal case. The elementary operations, however, are algebraic operations of higher degree.

We assume that each curve is given as a sequence of pieces which are connected at their endpoints. Every piece is a smooth curve of class C^2, i. e., the curvature is defined everywhere and varies continuously within a piece. We will not make any assumptions how the curves are given; it is only important that the necessary geometric operations

⋆ Partially supported by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-2000-26473 (ECG—Effective Computational Geometry for Curves and Surfaces).

can be carried out.

We need curves whose turning angle is bounded by π. Curves of larger bounding angle must be subdivided. For solving the decision problem with parameter ε, we subdivide the curve at all points where the curvature is $1/\varepsilon$, in order to ensure that in each piece of the curve, the curvature is either uniformly smaller or bigger than $1/\varepsilon$.

We have omitted most proofs, but we state one auxiliary lemma in order to illustrate the elementary arguments on which the results are based.

Lemma 2 *Let f be a smooth curve of turning angle at most π, and let c by a circle of radius r.*

(a) *If the curvature of f is at most $1/r$ everywhere, the curve can intersect c at most twice. If it intersects c twice, then its endpoints lie outside c or on the boundary, and the middle piece between the two intersections lies inside c.*

(b) *If the curvature of f is at least $1/r$ everywhere, the curve can intersect c at most twice.*

The full version of this paper is available as a technical report [5].

2. The Free Space Diagram

The main tool of the algorithm is the *free space diagram* which was introduced in [2]. It is a two-dimensional representation of all pairs of points on the two curves, together with the identification of those pairs which are closer than ε.

Definition 3 *Let $f\colon I \to \mathbb{R}^2$, $g\colon J \to \mathbb{R}^2$ be two curves, $I, J \subseteq \mathbb{R}$. The set*

$$F_\varepsilon(f,g) := \{\, (s,t) \in I \times J : \|f(s) - g(t)\| \le \varepsilon \,\}$$

denotes the free space *of f and g. the partition of $I \times J$ into the free space and its complement is called the* free space diagram.

Points in F_ε are called *feasible* or *free*, and they are usually drawn in white. The other points are called *forbidden points* or *obstacles*, see Figure 1. The following simple observation from [2] is crucial.

Lemma 4 *Let $f\colon I = [l_I, r_I] \to \mathbb{R}^2$, $g\colon J = [l_J, r_J] \to \mathbb{R}^2$ be two curves. Then $\delta_F(f,g) \le \varepsilon$ if and only if there exists a curve within $F_\varepsilon(f,g)$ from (l_I, l_J) to (r_I, r_J) which is monotone in both coordinates.* □

As f and g consist of several pieces, the free space diagram decomposes naturally into a grid of rectangular *cells*.

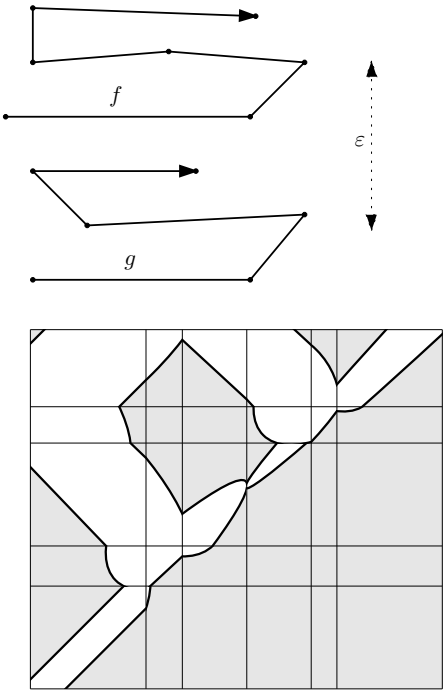


Fig. 1. Two polygonal curves and their free space diagram. The scale of the free space diagram is 50% reduced with respect to the curves.

3. Critical points

We regards as *critical points* on the boundary of F_ε those points which are local extrema in the horizontal or vertical direction. There are eight classes of critical points, shown in Figure 2.

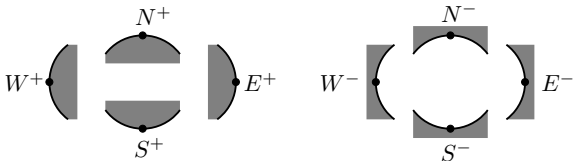


Fig. 2. The eight types of critical points. N, S, E, W refers to the direction in which the point is extreme, and the superscript tells whether the area in this direction is feasible (+) or forbidden (−).

In terms of the curves f and g, these points correspond to situations where a circle c of radius ε around a point of one curve is tangent to the other curve. For example, a critical point of type W^+ occurs in the situation where g touches the circle c of radius ε around a point x on f from inside. As x proceeds further away from g, a portion of g begins to stick out from c.

4. The structure of a single cell

The free space may be arbitrarily complicated even inside a cell. For example, if ε is very small, F_ε will contain isolated islands of free space for all intersections between f and g. However, we will show that the reachable points can be computed in a constant number of elementary geometric operations.

We have subdivided the curves, and consequently, the parameter intervals I and J into m and n pieces, respectively. Correspondingly, we cut the rectangle $I \times J$ into mn cells. On the boundaries of these cells, we compute all points which are *reachable* from the lower left corner (l_I, l_J) of the rectangle by a path in free space which is monotone in both directions. We do this incrementally from the lower left cell to the uppermost right cell.

A vertical line in the free-space diagram corresponds to a fixed point $f(s)$ on f. The points in F_ε on this line correspond to the points of g which lie inside a circle c of radius ε around $f(s)$. The boundary of F_ε corresponds to the intersections of c with g, and hence we can apply Lemma 2.

Lemma 5 *Inside a cell, a vertical or horizontal line intersects the boundary of F_ε at most twice.*

A vertical tangent line trough a critical point of type E or W or a horizontal tangent line trough a critical point of type N or S does not cross the boundary of F_ε in any other point. □

Lemma 6 *A curve forming a component of the boundary of the free space inside a cell can contain at most four critical points.* □

This lemma implies that there is a limited number of possibilities for such a boundary, the most complicated being an "s-shaped" path between between the left edge and the right edge of the rectangle, containing two critical points S^+ and N^-.

5. Processing a cell

We are given the reachable points on the left and bottom edge, and we compute the points on the right edge and on the top edge which are reachable from there.

On each edge of the rectangle there are at most two intervals of free points, by Lemma 5. Inside each interval of free points, there is only a single interval of reachable points because from every free point, everything which is to the right or to the top in the same free interval is reachable directly.

We will illustrate how to compute the *leftmost* reachable point in each free interval on the top edge from a given interval X on left edge. Other cases are similar.

We are given the lowest reachable point B in X. The upper end of X may be the upper left corner of the rectangle, or it may be a forbidden point which belongs to a component O of forbidden points. Similarly, the left endpoint F of Y may be part of a component of forbidden points, which we denote by O_2. (O and O_2 are not necessarily different, see Figure 3a.)

Lemma 7 *The leftmost point U in Y reachable from X depends only on the presence and the relative locations of O and O_2 and the horizontal line through B.*

PROOF. We have to show that any other "obstacles" of forbidden points do not play any role in this question. We show this by giving an algorithm for constructing U in all cases.

If the horizontal line through B intersects O or O_2, it is clear that one cannot reach Y, see for example the interval X_1 in Figure 3a or the interval X_2 in connection with Y_2 in Figure 3b. Otherwise, we claim that the desired point U lies directly above the rightmost point of O or of O_2, whichever is further to the right.

The monotone path from X to Y has to pass to the right of O and O_2. Thus, no point in Y left of U is reachable from X. To see that U is reachable, consider first the case that O exists, see the example of the interval X_1 in Figure 3b. Let A be the rightmost point of O. A can lie on the upper edge, or it can be a critical point of type E^+.

Assume first that A is a critical point of type E^+. The vertical line a through A lies completely in the free space, by Lemma 5, and O is the only obstacle left of a. By assumption, the horizontal line b from the lowest reachable point B in X does not intersect O before reaching a, and there are no other obstacles in this range. Thus, A is reachable from B, and the upper end A' of a is the leftmost reachable point on the top edge. If it lies in Y, we can take it as our point U, and we are done. (This is the case for the intervals X_1 and Y_1 in Figure 3b.) If Y lies left of a, we are done as well, as no points in Y are reachable from X. So let us deal with the

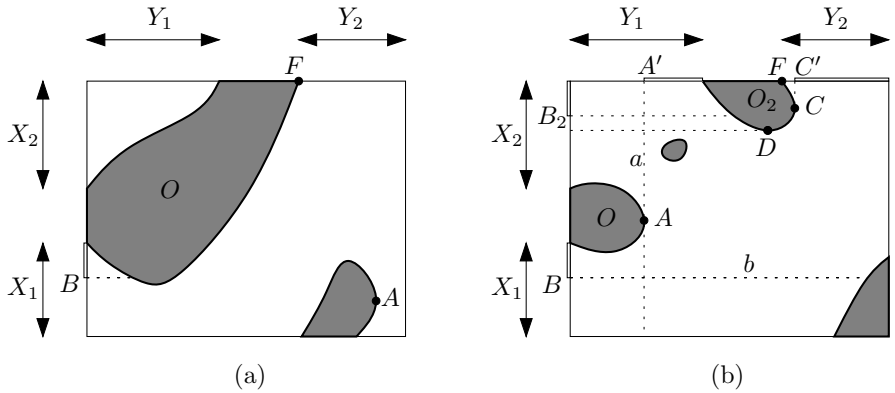


Fig. 3. Determining the reachable points on the top edge

only remaining case that Y lies to the right of a, and a is separated from Y by the obstacle O_2.

The lowest point D of O_2 must lie above O, and the horizontal line through D intersects a, which is reachable. Therefore D is reachable. From D we can reach the rightmost point C of O_2, which is either a critical point of type E^+ or the point F. In either case, we can indeed reach the point C' vertically above C as the leftmost point U.

The cases when O does not exist, or when the rightmost point A of O lies on the upper edge, can be treated similarly. □

Will have described our procedure in terms of geometric operations in the free space diagram, like finding the right-most point in a component of forbidden points. By working out what these operations mean in terms of the curves f and g, we obtain the following theorems.

Theorem 8 *Given the reachable points on the bottom edge and the left edge of a cell, the reachable points on the top edge and the right edge of the cell can be computed in a constant number of the following operations:*

- *Intersecting a circle of radius ε with one of the curves*
- *Finding the first intersection of one curve with an offset curve of the other curve at distance ε.*

In both cases, we must be able to find the parameter values on the respective curves, corresponding to the points that we have computed. □

Theorem 9 *Given two curves consisting of m and n pieces, respectively, where each piece has a turning angle at most π and has curvature $\geq \varepsilon$ or $\leq \varepsilon$ throughout, we can decide $O(m+n)$ space and in $O(mn)$ primitive operations of the type described in Theorem 8 whether their Fréchet distance is at most ε, for a given parameter ε.* □

6. The Minimization Problem

The minimization problem of *computing* the Fréchet distance can be solved by Megiddo's parametric search technique [4], closely following the approach of [2] for polygonal curves. The technical details are more involved, and we have to make some stronger assumptions on the curves.

Theorem 10 *Given two curves consisting of m and n pieces, respectively, of smooth algebraic curves of fixed maximum degree we can compute their Fréchet distance in $O(nm)$ space and in $O(mn\log(mn))$ algebraic operations, i.e., degree comparison between two real solutions of algebraic equations of bounded degree.* □

References

[1] H. Alt, A. Efrat, G. Rote, and C. Wenk, *Matching planar maps*, Journal of Algorithms **49** (2003), 262–283.

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

[3] H. Alt, C. Knauer, and C. Wenk, *Matching polygonal curves with respect to the Fréchet distance*, STACS 2001 (A. Ferreira and H. Reichel, eds.), Lect. Notes Comp. Sci., vol. 2010, Springer-Verlag, 2001, pp. 63–74.

[4] N. Megiddo, *Applying parallel computation algorithms in the design of serial algorithms*, J. Assoc. Comput. Mach. **30** (1983), 852–865.

[5] Günter Rote, *Computing the fréchet distance between piecewise smooth curves*, Tech. Report ECG-TR-241108-01, 2003.

On the Number of Pseudo-Triangulations of Certain Point Sets [1]

Oswin Aichholzer [a,2], David Orden [*,b,3], Francisco Santos [c,3] and Bettina Speckmann [d]

[a] *Institute for Software Technology, Graz University of Technology, Austria.*
[b] *Mathematics Department, University of Alcalá, Spain.*
[c] *Department of Mathematics, Statistics and Computer Science, University of Cantabria, Spain*
[d] *Department of Mathematics and Computer Science, Eindhoven University of Technology*

Abstract

We compute the exact number of pseudo-triangulations for two prominent point sets, namely the so-called double circle and the double chain. We also derive a new asymptotic lower bound for the maximal number of pseudo-triangulations which lies significantly above the related bound for triangulations.

Key words: Counting, Pseudo-triangulations, Triangulations, Double circle, Double chain.

1. Introduction

Pseudo-triangulations, a.k.a. geodesic triangulations, generalize triangulations and have found multiple applications in Computational Geometry in the last years. They were originally studied in the context of visibility [10,11] and ray shooting [5,6], but have been used in kinetic collision detection [1,8], rigidity [16], and guarding [15].

A *pseudo-triangle* is a polygon with exactly three vertices, called *corners*, with internal angles less than π. A *pseudo-triangulation* of a set S of points in the plane is a partition of the convex hull of S into pseudo-triangles whose vertex set is exactly S. A vertex is called *pointed* if it has an adjacent angle greater than π. *Pointed pseudo-triangulations*, are the ones with all vertices pointed.

The set of all pseudo-triangulations of a point set has somewhat nicer properties than that of all triangulations. For example, pseudo-triangulations of a point set with n elements form the vertex set of a certain polyhedron of dimension $3n-3$ [9]. The diameter of the graph of pseudo-triangulations is $O(n \log n)$ [3] versus the $\Theta(n^2)$ diameter of the graph of triangulations of certain point sets. For standard triangulations it is not know which sets of points have the fewest or the most triangulations, but it was shown in [2] that sets of points in convex position minimize the number of pointed pseudo-triangulations.

Let A be a point set and let A_I be its subset of interior points. The pseudo-triangulations of A can naturally be stratified into 2^{A_I} sets. More precisely, for each subset $W \subseteq A_I$ we denote by $PT_W(A)$ the set of pseudo-triangulations of A in which the points of W are pointed and those of $A_I \setminus W$ are non-pointed. For example, $PT_\emptyset(A)$ is the set of triangulations of A and $PT_{A_I}(A)$ is the set of pointed pseudo-triangulations of A. The following conjecture is implicit in previous work:

Conjecture 1 *For every point set A in general position in the plane, the cardinalities of $PT_W(A)$ are monotone with respect to W. That is to say, for any $W \subseteq A_I$ and for every $v \in W$, one has*

$$|PT_W(A)| \geq |PT_{W\setminus\{v\}}(A)|.$$

Note that [12] proves the following inequality in the other direction:

$$3\,|PT_{W\setminus\{v\}}(A)| \geq |PT_W(A)|.$$

* Corresponding author
Email addresses: oaich@ist.tugraz.at (Oswin Aichholzer), david.orden@uah.es (David Orden), santosf@unican.es (Francisco Santos), speckman@win.tue.nl (Bettina Speckmann).

[1] Parts of this work were done while the authors visited the Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya.
[2] Research partially supported by Acciones Integradas 2003-2 004, Proj.Nr.1/2003
[3] Research partially supported by Acción Integrada España-Austria HU2002-0010 and grant BFM2001-1123 of Spanish Dirección General de Investigación Científica

In this paper we compute the number of pseudo-triangulations, and so check Conjecture 1, for two prominent point sets, namely those with the asymptotically maximal and minimal number of triangulations known so far:

1.1. *Points in almost convex position.*

For any given pair of numbers (v, i), $3 \leq i \leq v$, a point set in almost convex position with parameters (v, i) consists of the v vertices of a convex v-gon and a set of i interior points, placed sufficiently close to i different edges of the v-gon.

This is a special case of the "almost-convex polygons" studied in [7]. There it is shown that the number of triangulations of such a point set does not depend on the choice of the i edges of the v-gon. Indeed, if we call this number $n(v, i)$ one has

$$n(v, i) = n(v + 1, i - 1) - n(v, i - 1)$$

and from this $n(v, i)$ can be computed recursively, starting with $n(v, 0) = C_{v-2}$ (the Catalan number). The array obtained by this recursion (difference array of Catalan numbers) appears in Sloane's Online Encyclopedia of Integer Sequences [14] with ID number A059346 (note that there $n(v, i)$ appears for every $i \geq 0$ and $v \geq 2$, although only the cases $v \geq \max\{i, 3\}$ have an interpretation as counting triangulations). Asymptotically, $n(v, i)$ equals $4^v 3^i$, modulo a polynomial factor.

The extremal case with $v = i = n/2$ (referred to as *double circle*), has asymptotically $\Theta(\sqrt{12}^n n^{-3/2})$ triangulations. It is conjectured in [4] that this is the smallest number of triangulations that a point set with n points can have.

1.2. *Double chain.*

For any two numbers $l, m \geq 0$, a double chain is a convex 4-gon with l and m points, respectively, placed forming concave chains next to opposite edges of the 4-gon in a way that the they do not cross the two diagonals of the convex 4-gon (see Fig. 1). The double chain decomposes into a convex $l+2$-gon, a convex $m+2$-gon, and a non-convex $l+m+4$-gon, which have C_l, C_m, and $\binom{l+m+2}{l+1}$ triangulations respectively. Hence, the double chain has

$$C_l C_m \binom{l + m + 2}{l + 1}$$

triangulations. In the extremal case $l = m = (n - 4)/2$ this gives $\Theta(8^n n^{-7/2})$. This is (asymptotically) the point set with the largest number of triangulations known so far.

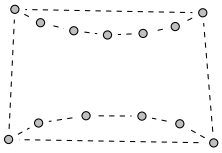


Fig. 1. A double chain: $l = 5$ and $m = 4$.

2. The double circle and its relatives

Fix two integers i, v, $3 \leq i \leq v$, and let A be a point set in almost convex position with parameters (v, i). Let p be a specific interior point of A and let qr be the convex hull edge which has p next to it. Let B and C be the point sets obtained respectively by deleting p from A and by moving p to convex position across the convex hull edge qr:

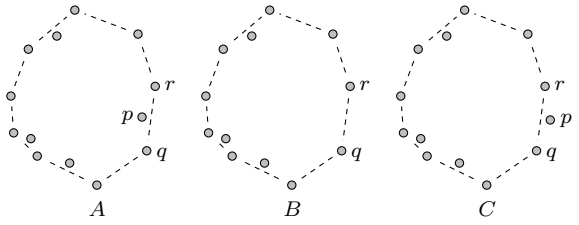


Fig. 2. The point sets A, B and C. Here $v = 9$ and $i = 4$.

Lemma 2 *Let W be a set of interior points of A not containing p (so that W is also a set of interior points of B and C). Then:*

(i) $|PT_W(A)| = |PT_W(C)| - |PT_W(B)|$.

(ii) $|PT_{W\cup\{p\}}(A)| = 2\,|PT_W(C)| - |PT_W(B)|$.

Corollary 3 *(i) A satisfies Conjecture 1.*

(ii) The numbers $|PT_W(A)|$ depend only on v, i and $k := |W|$.

We omit the proofs of these and other results due to the limited space available for this abstract.

Let $n(v, i, k)$ denote the numbers referred to in part 2 of the corollary. Since $n(v, i, 0) = 4^v 3^i$ (modulo a polynomial factor) and since

$$n(v, i, k) = 2(v + 1, i - 1, k - 1) - n(v, i - 1, k - 1),$$

we conclude that $n(v, i, k) \sim 4^v 3^{i-k} 7^k$, modulo a polynomial factor. Adding the numbers over all the possible subsets of interior points gives

$$\sum_{k=0}^{i} \binom{i}{k} 4^v 3^{i-k} 7^k = 4^v 10^i.$$

Hence, a double circle (the case $i = v = n/2$) has $\sqrt{28}^n$ pointed pseudo-triangulations and $\sqrt{40}^n$ pseudo-triangulations in total, modulo a polynomial factor.

3. The single chain

As a step towards the study of the double chain, let us start with a single chain. By this we mean a point set A with three extremal vertices and a concave chain of l points next to an edge. Equivalently, a convex $l+2$-gon together with a point in its exterior and which sees all but one of its edges. We call this special point the *top* point and denote it by p. Let $p_0, \ldots, p_{l+1}$ be the rest of the points, numbered from left to right, so that the interior points are $p_1, \ldots, p_l$. We define $A_I = \{p_1, \ldots, p_l\}$.

We are particularly interested in the pointed pseudo-triangulations of the single chain. We classify them according to which interior points are joined to the top. For any subset $W \subset A_I$ we denote by $PPT_W(A)$ the set of pointed pseudo-triangulations of A in which p is joined to p_i if and only if $p_i \in W$. Clearly, $PPT_\emptyset(A)$ is in bijection to the set of triangulations of the convex $l+2$-gon, hence its cardinality is the Catalan number C_l.

Lemma 4 *For every W:*

$$|PT_W(A)| = \sum_{W' \subset W} |PPT_{W'}(A)|.$$

Hence, Conjecture 1 holds for A.

As a special case of this lemma, $|PT_\emptyset(A)| = |PPT_\emptyset(A)|$. That is to say, triangulations of A are in bijection to triangulations of the convex $l+2$-gon. Curiously enough, $PPT_{A_I}(A)$ (that is, the pointed pseudo-triangulations in which the top point p is joined to everything), have the cardinality of the next Catalan number C_{l+1}, and flips between them form the graph of the corresponding associahedron (see [13], Section 5.3 and the remark and picture on pp. 728–729). The following is a 1-dimensional analog of Conjecture 1.

Conjecture 5 *For every $W \subset A_I$ and $p \in A_I \backslash W$,*

$$|PPT_{W \cup \{v\}}(A)| \geq |PPT_W(A)|.$$

Unfortunately, we do not know how to compute the numbers $PPT_W(A)$, or even recursive formulae for them. But we can compute the sum of all the $PPT_W(A)$'s for each cardinality of W.

Theorem 6 *Let $a(l,i) := \sum_{|W|=i} |PPT_W(A)|$.*

(i) $a(l,0) = C_l$, and $a(l,1) = (l+1)C_l$.

(ii) For every $i \geq 2$,

$$a(l,i) = \binom{l+1}{i} C_l - a(l-1, i-2).$$

Part 2 of Theorem 6 allows to compute all the values of $a(l,i)$ recursively, starting from those stated in part 1. The following table shows the first few values:

$l \backslash i$	0	1	2	3	4	5	$\sum_{i=0}^{l} a(l,i)$
0	1						1
1	1	2					3
2	2	6	5				13
3	5	20	28	14			67
4	14	70	135	120	42		381
5	42	252	616	770	495	132	2307

The recursion also tells us that the array $a(l,i)$ coincides with the sequence with ID A062991 in [14]. There, we learn that the row sums, that is, the numbers $|PT_{A_I}(A)|$ of pointed pseudo-triangulations of these point sets, form the sequence A062992 and satisfy:

$$|PT_{A_I}(A)| = 2\sum_{j=0}^{l} (-1)^{l-j} C_j 2^j - (-1)^l.$$

Corollary 7 *The following inequalities hold for the number of pointed pseudo-triangulations of a single chain of l interior points, $l \geq 2$:*

$$2^l C_l < 2^{l+1} C_l - 2^l C_{l-1} < |PT_{A_I}(A)| < 2^{l+1} C_l.$$

In particular, the number is in $\Theta(8^l l^{-3/2})$.

4. The double chain

Let A be a double chain with l and m interior points in the two chains, resp. (so A has $l+m+4$ points in total). We call the $l+2$ and $m+2$ vertices in the two chains the "top" and "bottom" parts. We show how to count the number of pointed pseudo-triangulations of A.

Let us call B and C single chains with l and m interior points each. B can be considered the subset of A consisting of the top part plus a bottom vertex, and analogously for C. Every pseudo-triangulation T_A of A induces pseudo-triangulations T_B and T_C of B and C as follows: consider on the one hand all the pseudo-triangles of T_A that use at most one vertex of the bottom, and contract these vertices to a single one. Do the same for pseudo-triangles with at most one vertex in the top (see Fig. 3).

Conversely, given a pair of pseudo-triangulations of B and C, if i (resp. j) denotes the number of interior edges incident to the bottom (resp. top) point, there are exactly $\binom{i+j+2}{i+1}$ ways to recover a pseudo-triangulation of A from that data, by shuf-

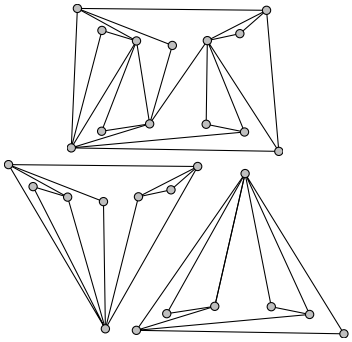


Fig. 3. Decomposing double chain pseudo-triangulations.

fling the $i+1$ pseudo-triangles of T_B incident to the bottom and the $j+1$ of T_C incident to the top.

Theorem 8 *Let V and W be subsets of the top and bottom interior points. For each $V' \subset V$ and $W' \subset W$ let $t^{V,W}_{V',W'} = \binom{l-|V\setminus V'|+m-|W\setminus W'|+2}{l-|V\setminus V'|+1}$. Then:*

$$|PT_{V\cup W}(A)| = \sum_{\substack{V'\subset V\\ W'\subset W}} t^{V,W}_{V',W'} |PPT_{V'}(B)||PPT_{W'}(C)|.$$

Corollary 9 *If Conjecture 5 holds, then the double chain satisfies Conjecture 1.*

For pointed pseudo-triangulations of the double chain, Theorem 8 says that:

$$|PT_{A_I}(A)| = \sum_{i=0}^{l}\sum_{j=0}^{m} \binom{i+j+2}{i+1} a(l,i)a(m,j),$$

where $a(\cdot,\cdot)$ is as in the previous section. The sequence for $l=m$ is

$$2, 38, 1476, 81310, 5495276, 424398044, \ldots$$

In order to analyze the asymptotics of this sequence we need the following lemma on the numbers $a(l,i)$:

Lemma 10

$$1 - \frac{i(i-1)}{(4l-2)(l-i+2)} \le \frac{a(l,i)}{\binom{l+1}{i}C_l} \le 1$$

Corollary 11 *Let A be a double chain with n points and with equal numbers on both sides (that is to say, $l=m=(n-4)/2$). Then:*

$$\Omega(12^n n^{-9/2}) \le |PT_{A_I}(A)| \le \mathrm{O}(12^n n^{-3/2}).$$

References

[1] P. K. Agarwal, J. Basch, L. J. Guibas, J. Hershberger, and L. Zhang. Deformable free space tilings for kinetic collision detection. In *Proc. 5th Workshop Algorithmic Found. Robotics* (A. K. Peters, 2001) 83–96.

[2] O. Aichholzer, F. Aurenhammer, H. Krasser, and B. Speckmann. Convexity Minimizes Pseudo-Triangulations. In *Proc. 14th Canad. Conf. Comp. Geom.* (2002) 158–161.

[3] O. Aichholzer, F. Aurenhammer, and H. Krasser. Adapting (pseudo)-triangulations with a near-linear number of edge flips. In *Lecture Notes in Computer Science 2748, Proc. 8th International Workshop on Algorithms and Data Structures (WADS), volume 2748* (2003) 12–24.

[4] O. Aichholzer, H. Krasser. The point-set order-type database: A collection of applications and results. In *Proc. 13th Canad. Conf. Comp. Geom.* (2001) 17–20.

[5] B. Chazelle, H. Edelsbrunner, M. Grigni, L. J. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica* **12** (1994) 54–68.

[6] M. Goodrich and R. Tamassia. Dynamic ray shooting and shortest paths in planar subdivision via balanced geodesic triangulations. *J. of Algorithms* **23** (1997) 51–73.

[7] F. Hurtado andM. Noy. Counting triangulations of almost-convex polygons. *Ars Combinatoria* **45** (1997) 169–179.

[8] D. Kirkpatrick, J. Snoeyink, and B. Speckmann. Kinetic collision detection for simple polygons. *Intern. Journal Comp. Geom. Appl.* **12** (2002) 3–27.

[9] D. Orden and F. Santos. The polytope of non-crossing graphs on a planar point set. Preprint 2003, *http://arxiv.org/abs/math.CO/0302126*, accepted in *Discrete Comput.Geom.*

[10] M. Pocchiola and G. Vegter. Minimal tangent visibility graphs. *Comput. Geom. Theory Appl.* **6** (1996) 303–314.

[11] M. Pocchiola and G. Vegter. Topologically sweeping visibility complexes via pseudo-triangulations. *Discrete Comp. Geom.* **16** (1996) 419–453.

[12] D. Randall, G. Rote, F. Santos and J. Snoeyink. Counting triangulations and pseudo-triangulations of wheels. In *Proc. 13th Canad. Conf. Comput. Geom.* (2001) 149–152.

[13] G. Rote, F. Santos, and I. Streinu. Expansive motions and the polytope of pointed pseudo-triangulations. *Discrete and Computational Geometry – The Goodman-Pollack Festschrift*, (B. Aronov, S. Basu, J. Pach, M. Sharir, eds), Algorithms and Combinatorics, Springer Verlag, Berlin, 699–736, 2003.

[14] N. J. A. Sloane. The On-Line Encyclopedia of Integer Sequences. Copyright 2003 AT&T, *http://www.research.att.com/~njas/sequences*

[15] B. Speckmann and C. D. Tóth. Allocating vertex π-guards in simple polygons via pseudo-triangulations. In *Proc. 14th Symp. Disc. Alg.* (2003) 109–118.

[16] I. Streinu. A combinatorial approach to planar non-colliding robot arm motion planning. In *Proc. 41st FOCS* (2000) 443–453.

Using Symmetry Evaluation to Improve Robotic Manipulation Performance

P. J. Sanz, R. Marín and S. Dabić

Department of Computer Science & Engineering. Univ. Jaume I. Campus Riu Sec (12071-Castellón)

Key words: Vision based Robotic Manipulation; 2D Grasp Determination; Geometric Reasoning.
PACS:

1. Introduction

Presently, the robotics domain is increasing its performance possibilities mainly by using all kind of recent sensor based technology, jointly with very efficient software and hardware to process the information in a suitable manner. In particular, new robotic applications in service context are starting to be available now (e.g. space and underwater activities, telesurgery, etc.), as can be observed in all the most important conferences around the world and in the real life scenarios. These emergent activities in robotics, outside the well structured and predictable industrial domains, would be impossible without the appropriate use of sensory information.

In our case, after some years of previous research in the robotic manipulation area, by using computer vision to guide the grasping actions of 2D objects [Sanz et al., 98], we have discovered the importance of implement some algorithms that make easier the underlying geometric reasoning necessary to improve the final robotic manipulation performance. In particular, the knowledge about symmetry of planar shapes (i.e. 2D images of the objects are available) has been successfully used by the authors [Sanz et al., 99], and some other researchers before [Blake, 95], within the grasping determination domain.

2. Problem Description and Previous Results

As it has been commented beforehand, in previous works the geometric reasoning necessary to determine suitable regions of one object, in order to be grasped, was supported by the symmetry knowledge associated to the contour of this object. With the aim to quantify this symmetry degree a new concept was introduced by the authors [Sanz et al., 99] the .normalized global symmetric deficiency.. In the following we clarify this concept. For a planar shape two privileged directions exist related to its mass distribution (inertia), these directions are I_{min} and I_{max}, and they represent the eigenvectors of the moment of inertia covariance matrix. If mirror symmetry exists, these directions are the first candidates for that. As a result of that the next objective is evaluate the symmetry degree associated with these two directions. From the curvature description K_{ki}, the symmetry degree is computed with respect to the I_{min} associated with the contour, using the intersection points $\{ \mathbf{C} \bigcap I_{min} \}$, computing Δ_i, such as:

$$\Delta_i = K_{k,P_3-i} - K_{k,P_3+i};\ i = 1, K,\ \tfrac{N}{2},$$

where $\mathrm{P}_3 \in \{\mathbf{C} \bigcap I_{min}\}$ traversing the contour clockwise from the initial point, PI, between P_1^0 and P_2^0. That is to say, Δ_i is computed as indicated, for each couple of points equidistant to P_3, covering all the contour. Using these quantities, we define the normalized global symmetric deficiency as:

$$\Phi = \tfrac{1}{N} \sum_{i=1}^{i=\frac{N}{2}} \Delta_i$$

where N is the total number of points in the contour. The same process is utilized to compute Φ for the Imax direction, but now, instead of P_3, we compute Δ_i with $P_1^0 \in \{ \mathbf{C} \bigcap I_{max} \}$, as introduced above. Note that $\Phi = 0$, i.e. perfect mirror symmetry with respect to the considered axis, exists only for an ideal mirror symmetric object. Some results in relation with the usefulness of Φ are shown in

Shape	I_{min}		I_{max}	
	μ_Φ	σ_Φ	μ_Φ	σ_Φ
Nut	1.86	0.74	1.99	0.69
Screwdriver	1.58	0.82	4.25	0.57
Scissors	1.59	0.92	4.85	0.32
Pliers	2.68	0.54	4.67	0.83
Pincers	4.42	0.61	5.60	0.87
Allen wrench	6.38	0.58	5.46	0.78

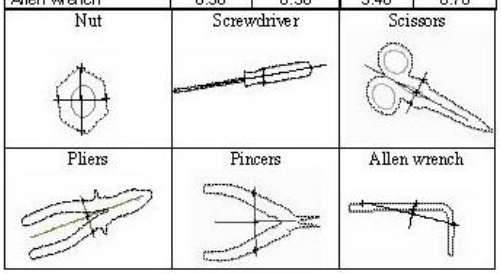


Table 1
Normalized global symmetric deficiency Φ computed for different images in both directions I_{min} and I_{max}. It shows the mean,ν_Φ, and the standard deviation, ρ_Φ, for each one.

Table 1.

The data shown in Table 1 are the mean and standard deviation computed from four digitizations for each object at different locations (position and orientation) over the work area. From that table some empirical results can be observed:

I_{min} direction. Looking at the table a gap between "pliers" and "pincers" is observed, $\Phi =\leq 3$. After many trials we have followed a mirror symmetry approach for those images that present $\Phi < 5$, named $\Phi_c = 5$ to this critical value.

I_{max} direction. In this case the gap appear just between "nut" and the rest of shapes. So a value $\Phi_c = 3$ represents a good critical value.

A primary classification of the objects present in Table 1 would be the following: "mirror symmetry for I_{min} and I_{max} directions" {nut}; "mirror symmetry in I_{min} direction" {nut..pincers}; and "without symmetry" {Allen wrench}. Note that only "nut" has mirror symmetry in both directions in correspondence with its inherent radial symmetry.

These results were applied successfully to the grasping determination domain [Sanz et al., 99], where the input were contours extracted from 2D images.

Nevertheless, as it has been remarked above, a problem was detected with some shapes, for instance, the pincers. The differences observed between the hopped (i.e. mirror symmetry along the Imin direction) and real results has been the starting point for the present research contribution. To

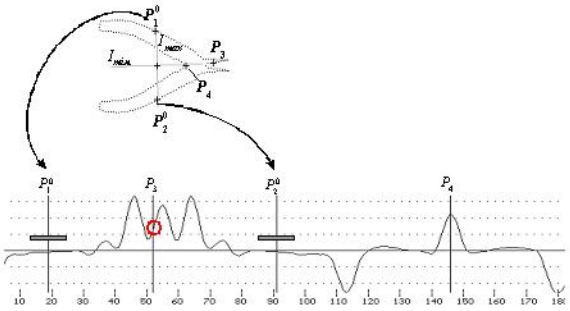


Fig. 1. The performance analysis in the case of pincers. Marked with a circle is observed the bad situation between the I_{min} axis and the external contour of this shape (i.e. P_3).

clarify this situation is convenient to observe the Fig 1, in which an image of pincers is processed. When the curvature-symmetry fusion [Sanz et al., 99] diagram is used to analyze the performance, we found that the intersection between one of the extremes of the I_{min} axis, and the external contour (i.e.P_3), is not well situated (i.e. see the circle in this Fig.1). And when the computation to evaluate Φ is carried out, the final result is that shown in Table1, namely a higher value that the theoretically hopped for this kind of shape, that as we can observe it is symmetric in that direction.

3. How to solve this problem?

Well, looking at literature we find the Leyton's Theorem [Leyton, 87], about "symmetry-curvature duality", were a local correspondence between a symmetry axis and a curvature point contour is established. Thus, if a symmetry axis exists in a shape, this axis intersect the shape always in a local extreme of curvature. Our present contribution has been to implement this Theorem in our algorithms in order to solve the initial problems found.

And, as work in progress, we are now testing the use of this new Φ, incorporating the Leyton.s Theorem, as a new descriptor in automatic object recognition.

Finally, it is noticeable that with this improvement we have got a very robust solution to evaluate the symmetry degree associated to a planar shape in a predefined direction, and in a very fast way,

making feasible real applications in the robotics domain.

References

[1] [Sanz et al., 98] Sanz PJ, del Pobil AP, Iesta JM, Recatal G. *Vision-Guided Grasping of Unknown Objects for Service Robots* . In IEEE Proc. on Robotics and Automation (ICRA.98), pp. 3018-3025. Leuven, Belgium. May 1998.

[2] [Sanz et al., 99] Sanz PJ, Iesta JM, del Pobil AP. *Planar Grasping Characterization Based on Curvature-Symmetry Fusion* . Applied Intelligence 10, pp. 25-36. Kluwer Academic Pub. 1999.

[3] [Blake, 95] Blake A. *A Symmetry Theory of Planar Grasp*. The International Journal of Robotics Research, Vol.14, No. 5, pp. 425-444. October, 1995.

[4] [Leyton, 87] Leyton M. Symmetry-Curvature Duality, Comput. Vision Graphics Image Process. 38, pp. 327-341. 1987

On geodesic and monophonic convexity [1]

Carmen Hernando [a] Mercè Mora [b] Ignacio M. Pelayo [c] Carlos Seara [b]

[a] *Dpt. Matemàtica Aplicada I, UPC, Barcelona, Spain, carmen.hernando@upc.es*
[b] *Dpt. Matemàtica Aplicada II, UPC, Barcelona, Spain, {merce.mora,carlos.seara}@upc.es*
[c] *Dpt. Matemàtica Aplicada III, UPC, Barcelona, Spain, ignacio.m.pelayo@upc.es*

Abstract

In this paper we deal with two types of graph convexities, which are the most natural *path convexities* in a graph and which are defined by a system $\mathcal{P}$ of paths in a connected graph G: the *geodesic convexity* (also called metric convexity) which arises when we consider shortest paths, and the *monophonic convexity* (also called minimal path convexity) when we consider chordless paths. First, we present a realization theorem proving, that there is no general relationship between monophonic and geodetic hull sets. Second, we study the contour of a graph, showing that the contour must be monophonic. Finally, we consider the so-called edge Steiner sets. We prove that every edge Steiner set is edge monophonic.

1. Introduction

A *convexity* on a finite set V is a family $\mathcal{C}$ of subsets of V, to be regarded as *convex sets*, which is closed under intersection and which contains both V and the empty set. The pair $(V, \mathcal{C})$ is called a *convexity space*. A finite *graph-convexity* space is a pair $(G, \mathcal{C})$, formed by a finite connected graph $G = (V, E)$ and a convexity $\mathcal{C}$ on V such that $(V, \mathcal{C})$ is a convexity space satisfying that every member of $\mathcal{C}$ induces a connected subgraph of G [4,5]. Thus, classical convexity can be extended to graphs in a natural way. We know that a set X of $\mathbb{R}^n$ is convex if every segment joining two points of X is entirely contained in it. Similarly, a vertex set W of a finite connected graph G is said to be a convex set of G if it contains all the vertices lying in a certain kind of path connecting vertices of W.

In this paper we deal with two types of graph convexities, which are the most natural *path convexities* in a graph and which are defined by a system $\mathcal{P}$ of paths in G: the *geodesic convexity* (also called metric convexity) [5,6,7,11] which arises when we consider shortest paths, and the *monophonic convexity* (also called minimal path convexity) [4,5] when we consider chordless paths.

[1] Partially supported by projects MCYT-FEDER TIC-2001-2171, MCYT-FEDER BFM 2002-0557, Gen Cat 2001SGR00224, MCYT BFM2000-1052-C02-01.

In what follows, $G = (V, E)$ denotes a finite connected graph with no loops or multiple edges. The *distance* $d(u, v)$ between two vertices u and v is the length of a shortest $u - v$ path in G. A *chord* of a path $u_0u_1 \dots u_h$ is an edge u_iu_j, with $j \geq i + 2$. A $u - v$ path ρ is called *monophonic* if it is a chordless path, and *geodesic* if it is a shortest $u - v$ path, that is, if $|E(\rho)| = d(u, v)$.

The *geodesic closed interval* $I[u, v]$ is the set of vertices of all $u - v$ geodesics. Similarly, the *monophonic closed interval* $J[u, v]$ is the set of vertices of all monophonic $u - v$ paths. For $W \subseteq V$, the *geodesic closure* $I[W]$ of W is defined as the union of all geodesic closed intervals $I[u, v]$ over all pairs $u, v \in W$. The *monophonic closure* $J[W]$ is the set formed by the union of all monophonic closed intervals $J[u, v]$.

A vertex set $W \subseteq V$ is called *geodesically convex* (or simply *g-convex*) if $I[W] = W$, while it is said to be *geodetic* if $I[W] = V$. Likewise, W is called *monophonically convex* (or simply *m-convex*) if $J[W] = W$, and is called *monophonic* if $J[W] = V$. The smallest g-convex set containing W is denoted $[W]_g$ and is called the *g-convex hull* of W. Similarly, the *m-convex hull* $[W]_m$ of W is defined as the minimum m-convex set containing W. Observe that $J[W] \subseteq [W]_m$, $I[W] \subseteq [W]_g$ and $[W]_g \subseteq [W]_m$. A *g-hull* (*m-hull*) set of G is a vertex set W satisfying $[W]_g = V$ ($[W]_m = V$).

For a nonempty set $W \subseteq V$, a connected subgraph of G with the minimum number of edges that contains all of W must be clearly a tree. Such a tree is called a *Steiner W-tree*. The *Steiner interval* $S(W)$ of W consists of all vertices that lie on some Steiner W-tree. If $S(W) = V$, then W is called a *Steiner set* for G [3].

The *monophonic* (*m-hull*, *geodetic*, *g-hull*, *Steiner*, respectively) number of G, denoted by $mn(G)$ ($mhn(G)$, $gn(G)$, $ghn(G)$, $st(G)$, respectively) is the minimum cardinality of a monophonic (m-hull, geodetic, g-hull, Steiner, respectively) set in G. Clearly, $ghn(G) \leq gn(G)$, since every geodetic set is a g-hull set. In [7] the authors showed that, apart from the previous one, no other general relationship among the parameters $ghn(G)$, $gn(G)$ and $st(G)$ exists. In Section 2, we approach the same problem by replacing the parameter $st(G)$ by both $mhn(G)$ and $mn(G)$.

In Section 3, we examine a number of monophonic convexity issues involving three types of vertices: contour, peripheral and extreme vertices [2]. We prove, among other facts, that the contour of a graph is monophonic. It is interesting to notice that this kind of results are closely related to the graph reconstruction problem, in the sense that we want to obtain all the vertices of a graph by considering a certain kind of paths joining vertices of a fixed set W.

In [3], it was shown that every Steiner set in G is also geodetic. Unfortunately, this particular result turned out to be wrong and was disproved by Pelayo [10]. In [7], the authors proved that every Steiner set is monophonic. As a consequence, they immediately derived that, in the class of distance-hereditary graphs (i.e., those graphs for which every monophonic path is a geodesic [8]), every Steiner set is geodetic. They also approached the problem of determining for which classes of chordal graphs (i.e., without induced cycles of length greater than 3) every Steiner set is geodetic, proving this statement to be true both for Ptolemaic graphs (i.e, distance-hereditary chordal graphs [5]) and interval graphs (i.e., chordal graphs without induced asteroid triples [9]). In Section 4, we focus our attention on the edges of geodesic and monophonic paths, approaching the same problems and obtaining similar results.

2. Monophonic and geodetic parameters

Let us review the main definitions involved in this section. A vertex set $W \subseteq V$ is a g-hull set if its g-convex hull $[W]_g$ covers all the graph, i.e., if $[W]_g = V$. Moreover, W is called geodetic if $I[W] = V$. The g-hull number $ghn(G)$ of G is defined as the minimum cardinality of a hull set. The geodetic number $gn(G)$ of G is the minimum cardinality of a geodetic set [6]. Certainly, $ghn(G) \leq gn(G)$.

Although it has been shown that determining the geodetic number and the hull number of a graph is a NP-hard problem [6], it is rather simple to obtain these two parameters for a wide range of classes of graphs as paths, cycles, trees, (bipartite) complete graphs, wheels and hypercubes.

A vertex set $W \subseteq V$ is a m-hull set if $[W]_m = V$. Moreover, W is called monophonic if $J[W] = V$. The m-hull number $mhn(G)$ of G is the minimum cardinality of a m-hull set. The monophonic number $mn(G)$ of G is the minimum cardinality of a monophonic set. Certainly, $mhn(G) \leq mn(G) \leq gn(G)$ and $mhn(G) \leq ghn(G)$, since every monophonic set is a m-hull set, every geodetic set is monophonic, and every g-hull set is a m-hull set. Nevertheless, it is not true that every g-hull set be monophonic. For example, if we consider the complete bipartite graph $K_{3,3}$, with $V_1 = \{a, b, c\}$ and $V_2 = \{e, f, g\}$, it is easy to see that the set $W = \{a, b\}$ satisfies $[W]_g = V$ and $J[W] = V \setminus \{c\}$.

At this point, what remains to be done is to ask the following question: *Is there any other general relationship among the parameters $mhn(G)$, $mn(G)$, $ghn(G)$ and $gn(G)$, apart from the previous known inequalities*?

The next realization theorem shows that, unless we restrict ourselves to a specific class of graphs, the answer is negative.

Theorem 1 *For any integers a, b, c, d such that $3 \leq a \leq b \leq c \leq d$, there exists a connected graph $G = (V, E)$ satisfying one of the following conditions:*

(i) $a = mhn(G)$, $b = mn(G)$, $c = ghn(G)$ *and* $d = gn(G)$,

(ii) $a = mnh(G)$, $b = ghn(G)$, $c = mn(G)$ *and* $d = gn(G)$.

3. Contour, peripheral and extreme vertices

Given a connected graph $G = (V, E)$, the *eccentricity* of a vertex $u \in V$ is defined as $ecc_G(u) = ecc(u) = \max\{d(u, v)|v \in V\}$. Hence, the diameter D of G can be defined as the maximum eccentricity of the vertices in G. The *periphery* of G, denoted $Per(G)$, is the set of vertices that have maximum eccentricity, i.e., the set of the so-called *peripheral* vertices. A vertex v is said to be *simplicial* in G if the subgraph induced by its neighborhood $N(v)$ is a clique. The *extreme set* of G, denoted $Ext(G)$, is the set of all its simplicial vertices. With the aim of generalizing these two definitions, the so-called *contour* of G was introduced in [2] as follows. A vertex $u \in W$ is said to be a *contour vertex* of W if $ecc_W(u) \geq ecc_W(v)$, for all $v \in N(u) \cap W$. The contour $Ct(W)$ of W is the set formed by all the contour vertices of W. If $W = V$, this set is called the contour of G and it is denoted $Ct(G)$. Notice that $Per(G) \cup Ext(G) \subseteq Ct(G)$.

Remark. We have examples of graphs showing that there is no general relationship between peripheral and extreme vertices.

Let $W \subseteq V$ be a m-convex (g-convex) set and let $F = \langle W \rangle_G$ be the subgraph of G induced by W. A vertex $v \in W$ is called an *m-extreme vertex* (*g-extreme vertex*) of W if $W \setminus \{v\}$ is a m-convex (g-convex) set. A vertex v of a m-convex (g-convex) set W is a m-extreme (g-extreme) vertex of W if and only if v is simplicial in F [5].

A convexity space $(V, \mathcal{C})$ is a *convex geometry* if it satisfies the so-called *Minkowsky-Krein-Milman* property: *Every convex set is the convex hull of its extreme vertices*. Notice that this condition allows us to rebuild every convex set from its extreme vertices, by using the convex hull operator. Farber and Jamison [5] proved that the monophonic (geodesic) convexity of a graph G is a convex geometry if and only if G is chordal (Ptolemaic). Cáceres et al. [2] obtained a similar property to the previous one, valid for every graph, by considering, instead of the extreme vertices, the contour vertices.

Theorem 2 *[2] Let $G = (V, E)$ be a connected graph and $W \subseteq V$ a g-convex set. Then, W is the g-convex hull of its contour vertices.*

As was pointed out in [2], the contour of a graph needs not to be geodetic. Nevertheless, we prove this assertion to be true in the following case.

Proposition 3 *If $Ct(G) = Per(G)$, then $Ct(G)$ is a geodetic set.*

PROOF. Let x be a vertex of $V(G) \setminus Ct(G)$. Since the eccentricities of two adjacent vertices differ by at most one unit, if x is not a contour vertex, then there exists a vertex $y \in V$, adjacent to x, such that its eccentricity satisfies $ecc(y) = ecc(x)+1$. This fact implies the existence of a path $\rho(x)= x_0x_1x_2 \dots x_r$, such that $x = x_0$, $x_i \notin Ct(G)$ for $i \in \{0, \dots, r-1\}$, $x_r \in Ct(G)$, and $ecc(x_i) = ecc(x_{i-1}) + 1 = l + i$ for $i \in \{1, \dots, r\}$, where $l = ecc(x)$. Moreover, $\rho(x)$ is a shortest $x - x_r$ path, since otherwise, the eccentricity of x_r would be less than $l+r$. But $x_r \in Ct(G) = Per(G)$ implies that $ecc(x_r) = D$ and $D = l + r$. Thus, there exists a vertex $z \in Per(G)$ such that $D = d(z, x_r) \leq d(z, x)+d(x, x_r) \leq ecc(x)+r = l+r = D$, that is, $d(z, x_r) = d(z, x)+d(x, x_r)$. Hence, x is on a shortest path between the vertices $z, x_r \in Per(G) = Ct(G)$.

As a consequence of this result, we have the following corollary.

Corollary 4 *If $Ct(G)$ has exactly two vertices, $Ct(G)$ is a geodetic set.*

Now, we approach the same issues by considering the monophonic convexity.

Theorem 5 *The contour of any connected graph G is a monophonic set.*

Corollary 6 *Let G be a connected graph and let $W \subseteq V$ be a m-convex set. Then, every vertex of W lies on a monophonic path joining contour vertices of W.*

PROOF. Let $F = \langle W \rangle_G$ be the subgraph of G induced by W, and let Ω be the set of contour vertices of W. Certainly, $\Omega = Ct(F)$. Hence, the previous statement is equivalent to saying that the contour of F is a monophonic set.

Corollary 7 *Let G be a connected graph and let $W \subseteq V$ be a m-convex set. Then, W is the m-convex hull of its contour vertices.*

Finally, we show another consequence of Theorem 5, which was directly proved in [2].

Corollary 8 *The contour of a distance-hereditary graph is a geodetic set.*

4. The edge Steiner problem

In this section, we focus our attention on the edges that lie in paths joining two vertices of $G = (V, E)$. We define the *edge intervals* of a graph as follows. The *edge geodetic closed interval* $I_e[u, v]$ is the set of edges of all $u - v$ geodesics. Similarly, the *edge monophonic closed interval* $J_e[u, v]$ is the set of vertices of all monophonic $u - v$ paths. For $W \subseteq V$, the *edge geodetic closure* $I_e[W]$ of W is the union of all edge closed intervals $I_e[u, v]$ over all pairs $u, v \in W$. The *edge monophonic closure*, $J_e[W]$, is defined as the union of all edge closed monophonic intervals over all pairs $u, v \in W$. In other words, we have

$$I_e[W] = \bigcup_{u,v \in W} I_e[u, v], \quad J_e[W] = \bigcup_{u,v \in W} J_e[u, v].$$

A vertex set W for which $J_e[W] = E$ is called an *edge monophonic set*. Similarly, W is an *edge geodetic set* if $I_e[W] = E$ [1]. A set $W \subseteq V$ is an *edge Steiner set* if the edges lying in some Steiner W-tree cover E. Notice that: (1) every edge Steiner set is a Steiner set, (2) every edge geodetic set is geodetic, (3) every edge monophonic set is monophonic, and (4) every edge geodetic set is an edge monophonic set. It is easy to find examples where the converses of these statements are not true. We have obtained the following results.

Theorem 9 *Every edge Steiner set of a connected graph is an edge monophonic set.*

Corollary 10 *In the class of connected distance-hereditary graph, every edge Steiner set is an edge geodetic set.*

Analogously to the vertex case, this last result also holds for interval graphs. First, we need to prove the following lemma.

Lemma 11 *(i) If P is a $x-y$ walk in G, any vertex of W is adjacent to at least a vertex of $V(P)$. (ii) If T_W is a Steiner W-tree, for any $u \in V(T_W) \setminus W$, u lies in the unique $x - y$ path of T_W. Moreover, there exists a Steiner W-tree, T_W^*, formed by the unique $x-y$ path of T_W and vertices of W adjacent to vertices of that path.*

Theorem 12 *In the class of connected interval graphs, every edge Steiner set is an edge geodetic set.*

In the preceding section we have seen that the contour of a graph is a monophonic set. Nevertheless, it is quite easy to find graphs whose contour is not an edge monophonic set.

It remains an open question the problem of characterizing those classes of chordal graphs for which *every edge Steiner set is edge geodetic*. We know this statement to be true for interval and Ptolemaic graphs, and false for split graphs (i.e., those chordal graphs whose complementary is also chordal).

References

[1] M. Atici and A. Vince, *Geodesics in graphs, an extremal set problem and perfect Hash families*, Graphs and Combinatorics, 18, (3), 2002, pp. 403–413.

[2] J. Cáceres, A. Márquez, O. R. Oellerman, M. L. Puertas, *Rebuilding convex sets in graphs*, 19th European Workshop on Computational Geometry, Bonn, 2003, submitted to Discrete Mathematics.

[3] G. Chartrand, P. Zhang, *The Steiner number of a graph*, Disc. Math., 242, 2002, pp. 41–54.

[4] P. Duchet, *Convex sets in graphs II. Minimal path convexity*, J. Comb. Theory ser. B, 44, 1988, pp. 307–316.

[5] M. Farber, R. E. Jamison, *Convexity in graphs and hypergraphs*, SIAM J. Alg. Disc. Math., 7 (3), 1986, pp. 433–444.

[6] F. Harary, E. Loukakis, C. Tsouros, *The geodetic number of a graph*, Math. Comput. Modelling, 17 (11), 1993, pp. 89–95.

[7] C. Hernando, T. Jiang, M. Mora, I. M. Pelayo, C. Seara, *On the Steiner, geodetic and hull numbers of graphs*, 19th British Combinatorial Conference, Bangor 2003; submitted to Discrete Mathematics.

[8] E. Howorka, *A characterization of distance-hereditary graphs*, Quart. J. Math. Oxford Ser. 2, 28 (112), 1977, pp. 417–420.

[9] C. G. Lekkerkerker, J. Ch. Boland, *Representation of a finite graph by a set of intervals on the real line*, Fund. Math., 51, 1962/1963, pp. 45–64.

[10] I. M. Pelayo, *Not every Steiner set is geodetic, a note on "The Steiner number of a graph"*, Disc. Math., accepted.

[11] V. P. Soltan, *Metric convexity in graphs*, Studia Univ. Babeş-Bolyai Math., 36, (4), 1991, pp. 3–43.

New Lower Bounds for the Number of Straight-Edge Triangulations of a Planar Point Set

Extended Abstract

Paul McCabe [a,1] and Raimund Seidel [b]

[a] *University of Toronto Dept. of Computer Science, 10 King's College Rd, Toronto Ontario, M5S 3G4 Canada*
[b] *Universität des Saarlandes, Im Stadtwald, 66123 Saarbrücken, Germany*

Abstract

We present new lower bounds on the number of straight-edge triangulations that every set of n points in plane must have. These bounds are better than previous bounds in case of sets with either many or few extreme points.

1. Introduction

For a finite set S of points in the plane let $T(S)$ denote the number of straight-edge triangulations of S. Let $t(h,m)$ and $T(h,m)$ denote the minimum and the maximum of $T(S)$ with S ranging over all sets with h extreme and m non-extreme ("interior") points. It is known that [4]

$$T(h,m) \leq \frac{59^m \cdot 7^h}{\binom{m+h+6}{6}}$$

and [1] that for $h+m \geq 1212$

$$t(h,m) \geq 0.092 \cdot 2.33^{h+m} = \Omega(2.33^{h+m}) .$$

We prove the following new bounds for $t(h,m)$:

Theorem 1 *For constant $c = 4829/116640 > 0.0414$ we have for all $h+m \geq 11$*

$$t(h,m) \geq c\Big(\frac{30}{11}\Big)^h \cdot \Big(\frac{11}{5}\Big)^m = c \cdot 2.\overline{7272}^h \cdot 2.2^m .$$

Theorem 2 *For every fixed h we have*

$$t(h,m) \geq \Omega(2.63^m) .$$

Email addresses: pmccabe@cs.toronto.edu (Paul McCabe), rseidel@cs.uni-sb.de (Raimund Seidel).

[1] Part of this research was done while the first author was with the International-Max-Planck-Research-School in Saarbrücken.

2. Proof Outline for Theorem 1

Let S be a finite planar set (in non-degenerate position), let p be an extreme point of S, and let $S_p = S \setminus \{p\}$. We call p of type (k,c) if the following holds: 1) The chain C_p of convex hull edges of S_p that are visible from p consists of exactly $k+1$ edges. 2) The chain C_p admits at most c simultaneous non-intersecting chords.

What is a chord? This is an edge e connecting vertices of C_p so that the polygon bounded by e and the relevant portion of C_p contains no point of S_p in its interior.

Note that if p is of type (k,c) then the difference of the number of extreme points of $S-p$ and the number of extreme points of S is $k-1$.

Theorem 1 is a consequence of the following two Lemmas:

Lemma 3 *Assume p is of type (k,c) and let I_p be the k interior vertices of the chain C_p. The following can be done c times, starting with $U = S_p$:*

Find a vertex $u \in I_p \cap U$ that is of type $(0,0)$ with respect to U and remove it from U.

Lemma 4 *Let p be an extreme point of S of type (k,c) so that $S_p \setminus C_p$ contains at least 4 points:*

$$T(S) \geq \Phi(k,c) \cdot T(S_p) ,$$

where $\Phi(0,0) = 3$, $\Phi(1,0) = 11/5$, and for all other pairs $c \leq k$

$$\Phi(k,c) = \Big(\frac{2^c}{2^{c+1}-1}\Big)\beta(k,c) + 1$$

$$\beta(k,c) = \begin{cases} \dfrac{2C_{k+1}-1}{2C_{k+1}-2} & \text{if } c = k \\ \dfrac{2C_{k+2}-1}{2C_{k+2}-2} & \text{if } c < k, \end{cases}$$

and $C_j = \binom{2j}{j}/(j+1)$ *is the* j*-the Catalan number.*

Chasing through the definitions in Lemma 4 one finds that extreme points of type $(k,0)$ yield a nice increase in triangulation count from S_p to S, with the worst being type $(1,0)$. If on the other hand p has type (k,c) with $c > 0$, which does not lead to such a large triangulation count increase, then Lemma 3 guarantees the existence of c other nice vertices of type $(0,0)$ that provide sufficient increase if S is built up by adding extreme points.

3. Proof Outline for Theorem 2

This proofs expands an initial idea of Francisco Santos [3]. It suffices to consider the case of $t(3,m)$, in other words the case of m points inside a triangle Δ.

Consider each of the m interior points in turn. Connect it to the three corners of Δ. This partitions Δ into three triangles, each of which can be triangulated recursively. This leads to the following recursive relation:

$$t(3,m) \geq m \cdot \min_{m_1+m_2+m_3=m-1} \{t(3,m_1)\cdot t(3,m_2)\cdot t(3,m_2)\}$$

Now we would like to set $t(3,m) \geq 2^{cm-a}$ for appropriate constants a and c and use induction based on the above recursive relation. The problem now is to make sure that this induction has a good base. For this purpose we construct using various computational methods explicit lower bounds $b(m)$ for $t(3,m)$ for m from 0 to some N (we used $N = 300$). Then we choose a and c so that $2^{cm-a} \leq b(m)$ for all $m \leq N$ and so that with $t(3,m) \geq 2^{cm-a}$ the above recursive relation is satisfied for all $m > N$. This amounts to the constraint that $2^{c+2a} \leq N+1$.

Values for c and a can then be found using linear programming, since the constraints for c and a after taking logarithms are linear. By optimizing c we then got that

$$t(3,m) \geq 0.093 \cdot 2.63^m .$$

4. Acknowledgments

We would like to thank Oswin Aichholzer for providing us with a preprint of the improved version of [1] and for providing the exact values of $t(h,m)$ for $h+m \leq 11$.

References

[1] O. Aichholzer, F. Hurtado and M. Noy, On the Number of Triangulations Every Planar Point Set Must Have, in *Proc. 13th Annual Canadian Conference on Computational Geometry CCCG 2001*, Waterloo, Canada, 2001, pp. 13–16. An improved version is to appear in CGTA.

[2] P. McCabe, Lower bounding the number of triangulations of straight-edge triangulations of planar point sets. *M.Sc. Thesis*, FR Informatik, Saarland University, 2003.

[3] F. Santos Private communication.

[4] F. Santos and R. Seidel, A better upper bound on the number of triangulations of a planar point set. *Journal of Combinatorial Theory, Series A 102(1)*, 2003, pp. 186–193.

A Simple and Less Slow Method for Counting Triangulations and for Related Problems

Extended Abstract

Saurabh Ray [a,1] and Raimund Seidel [b]

[a] *Max-Planck-Institut für Informatik Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany*
[b] *Universität des Saarlandes, Im Stadtwald, 66123 Saarbrücken, Germany*

Abstract

We present a simple dynamic programming based method for counting straight-edge triangulations of planar point sets. This method can be adapted to solve related problems such as finding the best triangulation of a point set according to certain optimality criteria, or generating a triangulation of a point set uniformly at random.

We have implemented our counting method. It appears to be substantially less slow than previous methods: instances with 20 points, which used to take minutes, can now be handled in less than a second, and instances with 30 points, which used to be solvable only by employing several workstations in parallel over a substantial amount of time, can now be solved in about one minute on a single standard workstation.

1. Introduction

In recent years there has been some interest in studying the set $\mathcal{T}(S)$ of straight-edge triangulations associated with a finite planar point set S. Typical problems are *counting,* i.e. given S determine $|\mathcal{T}(S)|$, *random generation,* i.e. given S randomly generate a triangulation in $\mathcal{T}(S)$ with uniform probability, or *optimization*, i.e. given S find the triangulation in $\mathcal{T}(S)$ that satisfies some optimality criterion.

The counting problem is at this point not known to be in P. It is known that $|\mathcal{T}(S)|$ is exponential in $n = |S|$ with the best currently known lower [4] and upper [14] bounds of roughly 2.33^n and 59^n. Some previous work on the counting problem addressed cases where S has some special structure [13,10,11,6]. An algorithm for the general case was given by Avis and Fukuda [5] which was based on their reverse-search paradigm and hence counts via enumeration. For another enumerative method see Rambau's TOPCOM page [12]. The so far best algorithm was given by Aichholzer [2]. It is a divide-and-conquer type algorithm based on the notion of so-called paths of a triangulation. Erkinger [8] implemented a prototype of this algorithm and subsequently Gimpl [9] wrote a "production type" implementation that even includes parallelization in the sense of distributing work to several machines. For usage see *"The Triangulation Homepage"* [3].

The random generation and the optimization problem can clearly be solved, though expensively, using the enumeration based method of Avis and Fukuda [5]. Aichholzer's method can as well be adapted to solve the random generation problem and also certain versions of the optimization problem, namely those that are "decomposable" in the following sense: Suppose one wishes to find the optimum triangulation of a polygon P with interior points and subject to the constraint that some set E of edges must be included in the triangulation. Suppose that E is such that it induces a partition of P into two polygons P_1 and P_2. Then the optimum triangulation of P subject ot E must be computable from the optimum triangulations of P_i subject to $E \cap P_i$, with $i = 1, 2$. Examples of optimization problems that satisfy this decomposability condition are MinMax-Area [7][p. 142] or minimum total edge-length (a.k.a. minimum weight).

Email addresses: saurabh@mpi-sb.mpg.de (Saurabh Ray), rseidel@cs.uni-sb.de (Raimund Seidel).

[1] The first author is supported by the International-Max-Planck-Research-School in Saarbrücken.

We present a simple dynamic-programming based algorithm for the counting problem. It can be viewed as a divide-and-conquer algorithm that in addition stores computed results of subproblems for later reuse. The algorithm can be naturally adapted to solve the random generation problem and to solve optimization problems that are decomposable in the sense described above. We report on a prototype implementation of the counting algorithm.

2. The Algorithm

We consider the slightly more general problem of computing the number $T(P)$ of triangulations of a simple polygon P that contains the point set S and whose corners are all in S. In the original problem polygon P is just the convex hull of S.

In courses on algorithm design the case that P has no "interior points" (i.e. the corners of P are precisely S) is a common homework problem for practicing the application of dynamic programming: Choose an arbitrary edge $e = [a, b]$ of P as "base edge." Let $C(e)$ be the set of all "candidate" vertices c of P so that the triangle spanned by a, b, c is contained in P. Consider some $c \in C(e)$. The diagonals $[a, c]$ and $[b, c]$ partition polygon P into three parts: the trianlge (a, b, c) and two subpolygons P_{ac} and P_{cb} (which may be trivial, i.e. an edge). The number of triangulations of P that include triangle (a, b, c) is then given by the product $T(P_{ac})T(P_{cb})$. Since in any triangulation of P the base edge $[a, b]$ has to be part of some triangle we get

$$T(P) = \sum_{c \in C(e)} T(P_{ac})T(P_{cb}) .$$

Of course the subproblems like $T(P_{ac})$ are solved recursively using the same method and using $[a, c]$ as base edge. This choice of base edge ensures that all subproblems ever considered only involve subpolygons of P formed by cutting P along just one diagonal. Since there are only $O(n^2)$ such subpolygons, only $O(n^2)$ many subproblems need to be solved in total and computed results can be stored and reused. This leads to an $O(n^3)$ "time" and $O(n^2)$ "space" algorithm, where the quotation marks are to remind the reader that the stated bounds only hold in a model where the cost of storing of and operating on arbitrarily large integers is constant. In our case we will be dealing with integers not larger than 59^n, i.e. representable by $O(n)$ many bits. This leads to an $O(n^4 \log n)$ time bound and an $O(n^3)$ space bound in the more appropriate word model of computation.

Let us now consider the general case where there are "interior point," i.e. S consists of more than just the corners of P. We will proceed as in the method outlined above: Choose some base edge $e = [a, b]$ from the boundary of P. The candidate set $C(e)$ now must contain all points $c \in S$ that together with e can span a triangle in some triangulation of P. Thus $C(e)$ consists of all $c \in S$ so that the triangle D_c spanned by a, b, c is contained in P and no point of S lies in the interior of triangle D_c. (In this abstract we assume nondegeneracy.) If a candidate point c is a corner of P we can proceed exactly as in the simple case outlined above and compute the number of triangulations of P containing the triangle D_c by solving two recursive subproblems. If c lies in the interior of P the number of triangulations of P containing D_c is given by the number of triangulations of the polygon $P_c = P \setminus D_c$, i.e. the polygon P with edge $[a, b]$ replaced by the chain of two edges $[a, c], [c, b]$. Thus we only need to solve one recursive subproblem, namely finding the number of triangulations of P_c. Note that this problem is easier than the original problem in the sense that there are one fewer interior points (and if there are no interior points we know how to solve the problem). Thus our algorithm proceeds as follows:

(i) Choose some base edge $e = [a, b]$.
(ii) Determine the candidate set $C(e)$.
(iii) For each $c \in C(e)$ that is not a corner of P compute $T(P_c)$ recursively.
(iv) For each $c \in C(e)$ that is a corner of P solve two recursive subproblems and compute $T(P_{ac})T(P_{cb})$.
(v) Return the sum of the computed numbers.

In this procedure some subpolygon Q of P may be considered many times. In the usual top-down dynamic programming fashion we will compute $T(Q)$ only once and store the result for later reuse. Unfortunately the arising subpolygons do not have in general such a nice form as they do in the simple case without interior points. Thus we cannot use an array for storage but need to resort to a hash table with a canonic sequence of vertex names around polygon Q serving as the key for Q.

n	h	m	avg. #triangulations	avg. time(seconds) Dynamic Programming	avg. time(seconds) Path Method
13	3	10	117017.2	0.009	0.250
18	3	15	434650561.2	0.064	43.512
23	3	20	2175541362109.0	0.982	*
28	3	25	17295702671778911.6	24.973	*
33	3	30	9064438879955990031.2	537.890	*
18	15	3	61156327.0	0.030	1.570
23	15	8	148363536731.6	0.178	262.232
28	15	13	596631344845165.6	2.569	*
33	15	18	2615696070967273559.2	55.623	*
23	20	3	49106130174.0	0.077	69.634
28	20	8	124263097179506.8	0.480	*
33	20	13	1303793620633224385.4	10.793	*

Table 1: Results for $n = h + m$ points, with m points randomly chosen in a convex h-gon; average taken over 5 runs. An asterisk indicates that the program did not complete a single instance of that size within 90 minutes.

3. Heuristics

The algorithm outlined above makes no restricions on the choice of the base edge. We have found the following two heuristics profitable for this choice.

If P has an edge e with $|C(e)| \leq 2$, then choose e as base edge.

Otherwise fix a line ℓ that has approximately one half of the points of S on each side and as long as interior points are considered as candidate points choose as base edge always a boundary edge of the current polygon that intersects ℓ. For dividing lines of subproblems always choose lines parallel to the initial ℓ.

The reasonability of the first heuristic is obvious. The second heuristic is reminiscent of the path method of Aichholzer [2]. It aims to steer the algorithm towards a rapid breakup of the polygons considered.

4. Preliminary Experimental Results

We have implemented our dynamic programming algorithm in C++ using the g++ Compiler and STL libraries. Oswin Aichholzer [1] kindly provided us with Gimpl's C-implentation [9] of his path-based method. Thus direct runtime comparisons were possible.

Our experiments were run on a single machine with a 2.40 GHz Intel Pentium 4 Processor, with 512 MB memory and 512 KB Cache, running under Linux 2.4.21.4.p4. We did not attempt to use the parallel feature of Gimpl's program that allows to spread work over several machines.

We compared the two implemtations in a set of experiments where we considered m points distributed uniformly at random in a convex h-gon. We report here the results for pairs (h, m) with $h \in \{3, 15, 20\}$ and $h+m \in \{13, 18, 23, 28, 33\}$. For each pair of parameters Table 1 reports the average over 5 runs of the computed number of triangulations and the average time (in seconds) taken for the computation by each program. An asterisk indicates that the program did not solve a single instance of that size within 90 minutes.

We also tried our method on the largest examples that Gimpl reported on, namely a set of 32 points representing European capitals and a set of 30 random points [3]. Gimpl ran these examples on clusters of machines with a 1GHz Athlon Thunderbird Processor and with 256 MB memory running under Linux. He did not report any explicit running times for these example but stated that these was the largest he could solve in "reasonable time" employing parallelism and several machines. We have been told that "reasonable time" in this context is to mean "several weeks."

Our implementation solved the 32 point problem in 70.6 seconds and the 30 point problem in 42.9 seconds.

The largest example we tried was a set of 35 points, with 32 points distributed randomly in a triangle. On our standard machine this example led to excessive paging due to the large size of the required hash table. However on a SPARC compute server we could complete this example in 15 minutes 32 seconds using one processor and 4 GB of main memory. Adjusting hash table size to available main memory may lead to a more graceful degradation of performance of our algorithm when the input size increases.

5. Discussion

We will not discuss in this abstract how our method can be applied to the random generation problem and to the optimization problem. The adaptions that need to be made are fairly standard.

Our method does not parallelize as easily as Aichholzer's method since reusing already computed results may require non-trivial communication between processors.

We are in the process of analyzing the running time of our method. Empirically we noticed that the number of recursive calls when computing $T(S)$ is always about $\sqrt{T(S)}$. So far we have been unable to prove this or any other non-trivial statement about the running time.

Of course determining the true computational complexity of plane triangulation counting still remains an open problem.

6. Acknowledgments

We would like to thank Oswin Aichholzer for providing valuable information about the existing implementations and also for kindly providing the implementations themselves.

References

[1] O. Aicholzer, Private Communication.

[2] O. Aicholzer, The path of a triangulation, in *Proc. 15th Symp. on Comp. Geometry 1999*, pp. 14–23.

[3] `http://www.cis.tugraz.at/igi/oaich/triangulations/counting/counting.html`

[4] O. Aichholzer, F. Hurtado and M. Noy, On the Number of Triangulations Every Planar Point Set Must Have, in *Proc. 13th Annual Canadian Conference on Computational Geometry CCCG 2001*, Waterloo, Canada, 2001, pp. 13–16. An improved version is to appear in CGTA.

[5] D. Avis and K. Fukuda, Reverse Search for Enumeration. *Discrete Appl. Math. 65*, 1996, pp. 21–46.

[6] R. Bacher, Counting triangulations of configurations. *Manunscript*, `http://arxiv.org/abs/math.CO/0310206`.

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

[8] B. Erkinger, Struktureigenschaften von Triangulierungen. *Master's Thesis*, TU-Graz, 1998.

[9] J. Gimpl, Enumeration von Triangulierungen. *Master's Thesis*, TU-Graz, 2002.

[10] F. Hurtado and M. Noy, Counting triangulations of almost convex polygons. *Ars Combinatorica 45*, 1997, pp. 169–179.

[11] V. Kaibel and G.M. Ziegler, Counting Lattice Triangulations. To appear in: *"British Combinatorial Surveys"* (C. D. Wensley, ed.), Cambridge University Press.

[12] `http://www.zib.de/rambau/TOPCOM/`

[13] D. Randall, G. Rote, F. Santos, and J. Snoeyink, Counting triangulations and pseudo-triangulations of wheels, in *Proc. 13th Annual Canadian Conference on Computational Geometry CCCG 2001*, Waterloo, Canada, 2001, pp. 149–152.

[14] F. Santos and R. Seidel, A better upper bound on the number of triangulations of a planar point set. *Journal of Combinatorial Theory, Series A 102(1)*, 2003, pp. 186–193.

Finding a widest empty 1-corner corridor

J.M. Díaz-Báñez [a], M. A. López [b] and J.A. Sellarès [c,*]

[a] *Universidad de Sevilla, Spain*
[b] *University of Denver, USA*
[c] *Universitat de Girona, Spain*

Abstract

Given a set of n points in the plane, we consider the problem of computing a widest empty 1-corner corridor. We star giving a characterization of the 1-corner corridors that we call *locally widest*. Our approach to finding a widest empty 1-corner corridor consists of identifying a set of 1-corner corridors locally widest, that is guaranteed to contain a solution. We describe an algorithm that solves the problem in $O(n^4 \log n)$ time and $O(n)$ space.

1. Introduction

A *corridor* $c = (\ell, \ell')$ is the open region of the plane bounded by two parallel straight lines ℓ and ℓ'. The width of c is the Euclidean distance $d(\ell, \ell')$ between its two parallel bounding lines. A *link* L is an open region defined by two parallel rays $r(L) = p + \boldsymbol{v}t$ and $r'(L) = p' + \boldsymbol{v}t$, and a line segment $s(L) = \overline{pp'}$, forming an unbounded trapezoid. We consider the points of $s(L)$, but not those of $r(L)$ and $r'(L)$, as part of the link. The width of a link L, denoted by $\omega(L)$, is the Euclidean distance $d(r(L), r'(L))$ between its bounding parallel rays.

A 1-corner corridor $C = (L, L')$ is the union of two links L and L' sharing only the segment $s(L) = s(L')$. Thus, C is an open region bounded by an *outer boundary* that contains a convex corner with respect to the interior of the corridor, and an *inner boundary* that contains a concave corner. Each boundary consists of two rays which we call the *boundary legs*. We adopt the convention of using $r(L)$ and $r(L')$ (resp. $r'(L)$ and $r'(L')$) to denote the legs of the outer (resp. inner) boundary. The *width* of a 1-corner corridor C, denoted by $\omega(C)$, is the smaller of the widths of its two links. The *angle* $\alpha(C), 0 < \alpha(C) \leq \pi$, of the 1-corner corridor $C = (L, L')$ is the angle determined by the rays $r(L)$ and $r(L')$.

Let S be a set of n points in the Euclidean plane. A corridor c intersecting the convex hull, $CH(S)$, of S is *empty* if it does not contain any points of S. Note that an empty corridor must intersect $CH(S)$, as otherwise the widest empty corridor is not well-defined. A 1-corner corridor C is *empty* if it does not contain any points of S and its removal partitions the plane into two unbounded regions, each containing at least one point of S. Note that, as suggested in [Che96] (for the case $\alpha(C) = \pi/2$), it no longer suffices to require that candidate corridors intersect $CH(S)$, as this would allow such corridors to "scratch the exterior" of S without really "passing through" it. Widest empty corridor problems have applications in robot manipulation and spatial planning.

The problem of computing a widest empty corridor can be solved in $O(n^2)$ time and $O(n)$ space [HM88,JP94]. Cheng shows how to compute a widest empty 1-corner corridor for the case of fixed $\alpha(C) = \pi/2$, in $O(n^3)$ time and $O(n^3)$ space [Che96]. In this paper we allow $\alpha(C)$ to assume arbitrary values and describe an algorithm to compute a widest empty 1-corridor in $O(n^4 \log n)$ time and $O(n)$ space. It is clear that the solution to this problem might be not unique. In this paper we just look for one optimal corridor.

In the sequel, unless otherwise specified, whenever we talk about a 1-corner corridor, we assume that this corridor is empty.

Due to space constraint, in this extend abstract most proofs have been omitted.

* Corresponding author
Email addresses: dbanez@us.es (J.M. Díaz-Báñez), mlopez@cs.du.edu (M. A. López), sellares@ima.udg.es (J.A. Sellarès).

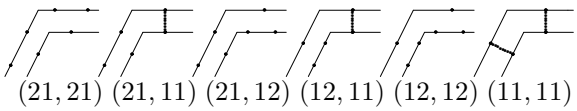


Fig. 1. Six types of locally widest corridors. The interior segments are perpendicular to the incident boundary legs.

2. Locally Widest Corridors

Since an empty corridor C can always be considered a 1-corner corridor with $\alpha(C) = \pi$, our optimal 1-corner corridor is at least as wide as the widest empty corridor. Thus, we can dismiss this case in $O(n^2)$ time and concentrate our attention on 1-corner corridors with $\alpha(C) < \pi$.

We begin with the obvious observation that there exists an optimal solution C that contains at least one point of S in each leg, otherwise the width of one or both links of C can be increased.

We say that a 1-corner corridor is *locally widest* if each leg contains at least one point of S and it is not possible to increase the width of either link by performing rotations of the legs around the points from S incident on the leg boundaries.

Lemma 1 *Each link L of a locally widest corridor satisfies one of the following conditions:*

21: There are points p_1 and p_2 of S that lie on the outer leg $r(L)$ and a point p' of S that lies on the inner leg $r'(L)$, such that both $\angle p'p_1p_2$ and $\angle p'p_2p_1$ are acute.

12: There are points p'_1 and p'_2 of S that lie on the inner leg $r'(L)$ and a point p of S that lies on the outer leg $r(L)$, such that both $\angle pp'_1p'_2$ and $\angle pp'_2p'_1$ are acute.

11: There are points p and p' of S that lie on the outer and inner legs of L, respectively, such that $\overline{pp'}$ is orthogonal to both $r(L)$ and $r'(L)$.

From the characterization given in Lemma 1 it results six classes of locally widest corridors, which we label $(21, 21)$, $(21, 11)$, $(21, 12)$, $(12, 11)$, $(12, 12)$, $(11, 11)$, depending on the types of the participating links. The six types of corridors are illustrated in Figure 1.

Lemma 2 *There always exists an optimal 1-corner corridor that is locally widest.*

Our approach to finding a widest empty 1-corner corridor consists of identifying a set $\mathcal{C}$ of locally widest 1-corner corridors that, by Lemma 2, is guaranteed to contain a solution. Our algorithm operates by systematically generating $\mathcal{C}$. Since there are members of $\mathcal{C}$ with six points on the boundary, a brute force algorithm would run in $O(n^7)$ time. Instead, our algorithm generates one boundary for each link, and computes the remaining boundaries by translating and appropriately adjusting a copy of a boundary ray until a point of S is encountered. This allows us to compute the optimal 1-corner corridor in $O(n^4 \log n)$ time.

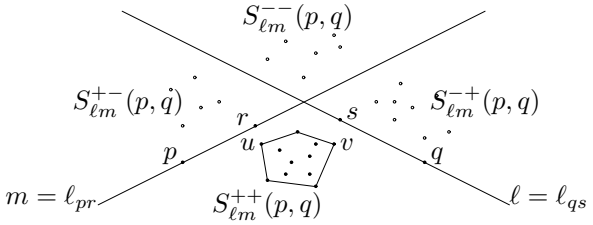


Fig. 2. Partition of the plane by boundary legs. The points u and v of $S^{++}_{\ell m}(p,q)$, closest to the outer boundary legs, help define the links of the corridor.

3. Preliminaries

For any two points p and q we denote by ℓ_{pq} the line through p and q and for any point t and line ℓ we denote by $H^+_\ell(t)$ (resp. $H^-_\ell(t)$) the open halfplane bounded by ℓ that contains (resp. does not contain) t. When $\ell = \ell_{pq}$, we simply write $H^+_{pq}(t)$ (resp. $H^-_{pq}(t)$). Also for any two non-parallel lines ℓ and m and points p and q, we let $S^{++}_{\ell m}(p,q)$ denote the subset of S in the interior of $H^+_\ell(p) \cap H^+_m(q)$. Again, when $\ell = \ell_{qs}$ and $m = \ell_{pr}$, we may write $S^{++}_{qspr}(p,q)$ to denote the same set of points. The other three regions determined by ℓ and m are labelled $S^{--}_{\ell m}(p,q)$, $S^{-+}_{\ell m}(p,q)$, and $S^{+-}_{\ell m}(p,q)$, as illustrated in Figure 2.

Our algorithms depend on finding efficiently a point of a subset P of S closest to a line ℓ that bounds a halfplane containing P. To this end, we take advantage of the following observation.

Observation 1 *Let H be an open halfplane bounded by a line ℓ and let P be a set of m points in H.*

- *The point of P closest to ℓ is a vertex of the convex hull $CH(P)$ of P.*
- *Once $CH(P)$ has been computed, the point of P closest to ℓ can be computed in $O(\log m)$ time.*

In the example of Figure 2, if P is the set of points in $S^{++}_{\ell m}(p,q)$, then u and v are the points of P closest to ℓ_{pr} and ℓ_{qs}, respectively. Points u, p and r (resp. v, q and s), define the boundaries of one of the links of a 1-corner corridor.

4. The algorithm

We describe an algorithm to compute a widest 1-corner corridor by processing each of the six cases described in Section 2. To simplify the description we assume that no three points of S are collinear. In practice, collinear degeneracies can be coped by using the simulation of simplicity technique [EM90].

Case (21,21).

Consider first an optimal corridor of type $(21, 21)$. Such a corridor contains points (p_1, p_2) on one outer leg and points (q_1, q_2) on the other. We do not discard the possibility that $q_2 = p_2$, which means that the convex corner of the corridor is a point from S that lies on both outer legs. We will compute all candidate corridors of type $(21, 21)$ for each triple (q_1, p_1, p_2) of points from S.

For each point $q_1 \in S$, we start by computing the radial ordering of $S \setminus \{q_1\}$ as a line ℓ through q_1 rotates around q_1. The initial orientation of the rotating line is arbitrary and rotation angles in the range $[0, \pi)$ suffice so that each input point is visited exactly once by one of the two rays making up the rotating line. In practice, and depending on the type of corridor sought, only a sublist of the entire sorted list will be needed, but this sublist depends of the choice of p_1 and p_2.

For each ordered pair (p_1, p_2) of points from $S \setminus \{q_1\}$, let $m = \ell_{p_1p_2}$ and $S' = S \cap H^+_m(q_1)$. The idea is to consider all corridors of type $(21, 21)$ that contain p_1 and p_2 on one outer leg, and q_1 on the other, and keep track of the widest. The points of S' are examined in radial order and each allowed to take on the role of q_2, i.e., the other point on the same outer leg as q_1. Initially, we set $q_2 = p_2$, as $\ell_{q_1p_2}$ is the position of the rotating line ℓ at the start of the sweep. The sweep then proceeds in the direction that causes the intersection of $\ell_{p_1p_2}$ and ℓ to move farther away from p_1. (This may be clockwise or counter-clockwise, depending on the relative position of q_1, p_1, p_2.) See Figure 3 for an example. The numeric labels indicate the order in which the points of S' are visited by the sweep.

As the sweep proceeds, starting from $q_2 = p_2$, and up to angle $\angle q_1p_2p_1$ from this position, the set $P = S^{++}_{\ell m}(p_1, q_1)$ changes dynamically. We keep track of $CH(P)$, and update it via insertions or deletions, depending on whether the points of S' enter or exit $H^+_\ell(p_1)$. For each point visited, we compute a tentative inner boundary by finding

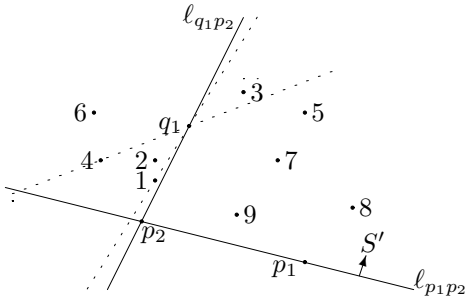


Fig. 3. Computing corridors of type $(21, 21)$ based on points q_1, p_1 and p_2. Solid line $\ell_{q_1p_2}$ defines the start of the sweep. Dashed linesillustrate different events of the sweep. Numeric labels indicate the order in which the points of S' are visited by the sweep.

the points p' and q' of P nearest to $m = \ell_{p_1p_2}$ and $\ell = \ell_{q_1q_2}$, respectively. This can be done in $O(\log n)$ time, as indicated in Observation 1. Points p' and q' define the inner boundary of a locally widest corridor if $\angle p'p_1, p_2$, $\angle p'p_2p_1$, $\angle q'q_1q_2$, and $\angle q'q_2q_1$ are all acute. In the example of Figure 3, when the sweepline reaches 1, i.e., when $q_2 = 1$, $CH(P) = \langle 3, 5, 8, 9\rangle$, $p' = 9$, and $q' = 3$. Since $\angle 3q_11$ is obtuse, these points do not define a locally widest corridor (a small counter-clockwise rotation around q_1 and 3, increases the width of the link through q_1). When the sweepline reaches 4, $CH(P) = \langle 1, 2, 5, 8, 9\rangle$, $p' = 9$, and $q' = 2$ and we have a valid 1-corner corridor of type $(21, 21)$.

The time to perform a radial sort is $O(n \log n)$. Since $CH(P)$ can be updated dynamically at an amortized cost of $O(\log n)$ per insertion or deletion [BJ02], the cost of one radial sweep is $O(n \log n)$. Since, there are $O(n^3)$ triples (q_1, p_1, p_2), the best candidate of type $(21, 21)$ can be found in $O(n^4 \log n)$ time and $O(n)$ space.

The remaining five cases can be solved in a similar way. We briefly describe how to process.

Case (21,11).

This case is handled by a simple modification to the ideas discussed above. As before, p_1 and p_2 lie on one outer leg, and q_1 lies on the other. Let ℓ denote the rotating line, and $\ell^\perp$ the line through q_1, perpendicular to ℓ. We handle an event every time ℓ goes through a point of S'. Unlike the previous case, these events serve only the purpose of keeping $CH(P)$ up to date and no candidate corridors are generated. Additionally, we handle an event every time $\ell^\perp$ goes through a point q. When this happens, the points p' and q' of P closest to $\ell_{p_1p_2}$ and ℓ

are computed, and a corridor of type (12,11) is generated if $q' = q$ and angles $\angle p'p_1p_2$ and $\angle p'p_2p_1$ are both acute. The complexity of both time and space remains the same as before.

Case (11,11).

This case is similar to $(21, 11)$ except for the interpretation of p_1 and p_2. Without lost of generality we could suppose that points p_1 and q_1 lie on the outer legs of the 1-corner corridor. For each pair p_1 and p_2 of points from S, if p_2 is the point of S' closest to the line m orthogonal to $\overline{p_1p_2}$ through p_1 then a sweep similar to that used for $(21, 11)$ is performed. The complexity remains the same.

Case (21,12).

The rotating line starts out parallel to $m = \ell_{p_1p_2}$. We keep track of two convex hulls, one built on $P = S^{++}_{\ell m}(p_1, q_1)$, and another built on $P' = S^{-+}_{\ell m}(p_1, q_1)$. For every point q_2 of S visited by the rotating ℓ, we find the point p' of P closest to m, and the point q' of P' closest to ℓ. The corridor is valid if the line parallel to ℓ through q' intersects with line m in such a way that points p_1, p_2 lie on the outer leg determined by m, and if angles $\angle p'p_1p_2$, $\angle p'p_2p_1$, $\angle q'q_1q_2$ and $\angle q'q_2q_1$ are all acute. Again the complexity remains the same.

Case (12,12).

We keep track of three convex hulls, built on $P = S^{++}_{\ell m}(p_1, q_1)$, $P' = S^{-+}_{\ell m}(p_1, q_1)$ and $P'' = S^{--}_{\ell m}(p_1, q_1)$. Find the point p' of $S^{+-}_{\ell m}(p_1, q_1)$ closest to m and the point q' of $S^{-+}_{\ell m}(p_1, q_1)$ closest to ℓ. Let m' be the line through p' parallel to m, ℓ' be the line through q' parallel to ℓ and denote $R = H^+_{m'}(p_1) \cap H^+_{\ell'}(q_1)$. If no point of $S^{--}_{\ell m}(p_1, q_1)$ lies inside R (condition that can be tested in $O(\log n)$ time by checking that ℓ' and m' do not intersect $CH(P'')$ and that any vertex v of $CH(P'')$ is outside R) then we are done. Otherwise we want to find the points of $S^{--}_{\ell m}(p_1, q_1)$ interiors to R closest to m and to ℓ, respectively. Find the points p'' and q'' of $S^{--}_{\ell m}(p_1, q_1)$ closest to m and ℓ, respectively. If q'' (p'') lies outside the region R, then find the edge s (t) of $CH(P'')$ intersected by ℓ' (m') that is interior in part to R and, abusing of language, denote by q'' (p'') the endpoint of s (t) that lies inside R. Observe that point q'' (p'') can be computed in $O(\log n)$ time. Compute a smaller corridor from the points p', p'', q', q'' found. If we denote p^* and q^* the points that define such a corridor, then the corridor is valid if the line parallel to ℓ through q^* intersects with line m in such a way that points p_1, p_2 lie on the outer leg determined by m, and if angles $\angle p^*p_1p_2$, $\angle p^*p_2p_1$, $\angle q^*q_1q_2$ and $\angle q^*q_2q_1$ are all acute. The best candidate of type (12,12) can be found in $O(n^4 \log n)$ time and $O(n)$ space.

Case (12,11).

This case may be handled by combining the ideas used to solve cases (21,11) and (12,12). Now $\overline{q_1q^*}$ is orthogonal to ℓ. Again the complexity remains the same.

In summary, we have established the following.

Theorem 1 *Let S be a set of n points. A widest 1-square corridor through S can be computed in $O(n^4 \log n)$ time and $O(n)$ space.*

5. Future work

We are presently working on other variants of the problem, which include finding a widest empty k-dense 1-corner corridor, dynamic updates, and approximation algorithms.

6. Acknowledgments

This problem was posed and partially solved during the *First Spanish Workshop on Geometric Optimization*, held in the Summer of 2002. The first author was supported in part by grant BFM2000-1052.The second author was supported in part by the National Science Foundation under grant DMS-0107628. The The third author was supported in part by grants TIC2001-2392-C03-01 and Gen. Catalunya DURSI-2001SGR-00296.

References

[BJ02] G. S. Brodal and R. Jacob, "Dynamic planar convex hull", *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pp 617-626, 2002.

[Che96] Siu-Wing Cheng, "Widest empty L-shaped corridor", *Information Processing Letters*, 58, pp 277-283, 1996.

[EM90] H. Edelsbrunner, E. P. Müecke, "Simulation of Simplicity", *ACM Transactions on Graphics*, 9, pp 66-104, 1990.

[HM88] M. Houle, A. Maciel, "Finding the widest empty corridor through a set of points", In G.T. Toussaint, editor *Snapshots of computational and discrete geometry*, pp 210-213 , 1988.

[JP94] R. Janardan, F.P. Preparata, "Widest-corridor problems", *Nordic Journal of Computing*, 1, pp 231-245, 1994.

Two Optimization Problems with Floodlights

Andreas Spillner
Institut für Informatik, Friedrich-Schiller-Universität Jena, 07743 Jena, Germany

Key words: illumination, floodlight

1. Introduction

The area of visibility and illumination problems is a very attractive field inside computational geometry. We refer the interested reader to [8] for an overview of recent results. In this paper we study two illumination problems involving floodlights. Loosely spoken a floodlight is a light source that illuminates a region in the plane which is bounded by two rays emanating from a common point, the apex of the floodlight. The size of a floodlight is the angle between its bounding rays.

In the first problem we are given a segment S of a straight line L in the plane and a finite set P of n points. P is contained in one of the open halfplanes bounded by L. We want to find a set $\mathfrak{F}$ of floodlights such that each point of S is contained in at least one floodlight, (ie the floodlights in $\mathfrak{F}$ illuminate S), such that the apex of each floodlight in $\mathfrak{F}$ is a point from P, and such that the sum of the sizes of the floodlights in $\mathfrak{F}$ is minimum. Furthermore we require that no two floodlights in $\mathfrak{F}$ share a common apex.

Variants of this problem have been studied before. In [1] the authors formulate a related decision problem and outline the difficulties in finding an efficient algorithm for it: We are given a segment S, called the *stage*, a set of points P and $|P|$ angles. Can floodlights whose sizes equal the given angles be placed at the given points to illuminate S? Please note that here it is not forbidden to place two or more floodlights at the same point.

A partial answer to the question about the computational complexity of the last mentioned variant was given in [6]. The variant considered there is stated as follows: We are given a segment S, a set of points P and a set of angles A. Every angle $\alpha \in A$ is mapped to a point $p(\alpha) \in P$. Can we find floodlights F_α, $\alpha \in A$, that together illuminate S, such that the apex of F_α is placed at $p(\alpha)$ and the size of F_α is α? It is shown that this problem is NP-complete.

The problem of illuminating a given segment S from a given set P of points such that the sum of the sizes of the floodlights used is minimized was studied in [4] and [3]. But there it was not required that the apices of the floodlights have to be pairwise distinct. The authors give an $O(n\log n)$ time and $O(n)$ space algorithm to solve this problem. Since it may happen that more than one floodlight is placed at a point from P, it could be interesting to study the problem with such placement explicitly ruled out. This was also posed as an open question in [8] and [2].

In the second problem that we consider in this paper we are given a convex polygon P with n vertices and a positive integer k. We want to illuminate P with k floodlights such that the sum of their sizes is minimum. Such a set of k floodlights is called optimal. The apices of the floodlights must be located in P.

For $k = 1$ the problem is easy: We have to find a vertex v of the convex polygon P such that the interior angle at v is minimum. This can trivially be done in $O(n)$ time. For $k = 2$ in [5] an algorithm is given that solves this problem in $O(n^2)$ time using $O(n)$ space. The algorithm exploits the fact that in an optimal set with two floodlights the apices of these two floodlights must be located at vertices of P. For $k \geq 3$ the problem was open [7]. We consider the case $k = 3$.

This paper is structured as follows. After this introduction we give some formal definitions. In the third section we present an $O(n\log n)$ time and $O(n)$ space algorithm for the first problem (illu-

Email address: spillner@minet.uni-jena.de (Andreas Spillner).

mination of a stage). In the fourth section we give an $O(n^2)$ time and $O(n)$ space algorithm for the second problem (illumination of a convex polygon with three floodlights). We end with some concluding remarks.

2. Preliminaries

Our setting will be the plane $\mathbb{R}^2$. For $M \subseteq \mathbb{R}^2$ we denote by $int(M)$ the interior of M, by $relint(M)$ the relative interior of M, by $bd(M)$ the boundary of M, and by $cl(M)$ the closure of M.

Definition 1 *Let p and q be points in the plane. By $\overline{pq}$ we denote the straight line segment with endpoints p and q.*

Definition 2 *Let $M \subseteq \mathbb{R}^2$. M is called convex iff for every $p, q \in M$ the straight line segment $\overline{pq}$ is contained in M.*

Definition 3 *Let R_l and R_r be two distinct rays with common starting point p. Let R be a ray rotating around p in clockwise direction starting at the position where $R = R_l$ and stopping at the position where $R = R_r$. Every point touched by R during its rotation belongs to the floodlight F with bounding rays $l(F) = R_l$ and $r(F) = R_r$. The point p is called the apex of F and denoted by $a(F)$. The angle from R_l to R_r in clockwise direction is called the size of F and denoted by $s(F)$.*

Definition 4 *Let $v_1, \ldots, v_n$ be pairwise distinct points in the plane and $n \in \mathbb{N}$, $n \geq 3$. The set of points $S = \overline{v_n v_1} \cup \bigcup_{i=1}^{n-1} \overline{v_i v_{i+1}}$ is called a simple closed polygonal curve iff it is a homeomorph of the boundary of an open disk and no two consecutive line segments on S are collinear. The points in $v(S) = \{v_1, \ldots, v_n\}$ are called the vertices of S. The line segments in $e(S) = \{\overline{v_n v_1}, \overline{v_1 v_2}, \ldots, \overline{v_{n-1} v_n}\}$ are called the edges of S.*

Definition 5 *A set P of points is called a simple polygon iff P is closed, bounded, and connected, $int(P) \neq \emptyset$ and $bd(P)$ is a simple closed polygonal curve. We set $v(P) = v(bd(P))$ and $e(P) = e(bd(P))$. A simple polygon P is called a convex polygon iff P is convex.*

Definition 6 *Let P be a simple polygon, p and q points, F a floodlight. We say that p and q can see each other and write this $p \leftrightarrow q$ iff $\overline{pq} \subseteq P$. The region visible from point p is $vis(p, P) = \{x \in \mathbb{R}^2 : p \leftrightarrow x\}$. The region illuminated by F is $ill(F, P) = F \cap vis(a(F), P)$.*

3. Illumination of a stage

Now we can state our *stage illumination problem* or *SIP* for short more formally: An instance (S, P) consists of a segment S of a straight line L and a finite set of points P such that P is contained in one of the open halfplanes bounded by L. We want to find a set $\mathfrak{F}$ of floodlights with the following properties:

(i) $S \subseteq \cup_{F \in \mathfrak{F}} F$
(ii) $\forall F \in \mathfrak{F}\ (a(F) \in P)$
(iii) $\forall F_1, F_2 \in \mathfrak{F}\ (F_1 \neq F_2 \ \Rightarrow\ a(F_1) \neq a(F_2))$
(iv) $s(\mathfrak{F}) = \sum_{F \in \mathfrak{F}} s(F)$ is minimum

Now it is clear that without loss of generality we can assume that L is the x-axis and all points in P have a positive y-coordinate, ie the points in P lie above the x-axis.

Definition 7 *Let (S, P) be an instance, $p \in P$ and z be a point on L. By $C_p(z)$ we denote the circle through the point p such that L is tangent to $C_p(z)$ at the point z.*

If we drop property (iii) in SIP we will obtain the problem considered in [4] and denote it by SIP*. From [4] we know that in a solution $\mathfrak{F}^*$ to an instance (S, P) for SIP* a point $z \in S$ is contained in a floodlight $F \in \mathfrak{F}^*$ if and only if no point of P lies outside $C_{a(F)}(z)$. Hence for SIP* to every instance there exists a unique solution. Please note that SIP* is meaningful even in the limiting case, ie if $S = L$.

Observation 8 *Consider the solution to an instance (S, P) for SIP*.*

(i) There are at most two floodlights placed at each point of P.
(ii) There is at most one point $p \in P$ such that there are two floodlights placed at p.
(iii) If there are two floodlights placed at $p \in P$ then p has a smaller y-coordinate than every other point in P.

If we have an instance (S, P) and solutions $\mathfrak{F}$ and $\mathfrak{F}^*$ to this instance for SIP and SIP* respectively then it is clear that $s(\mathfrak{F}) \geq s(\mathfrak{F}^*)$.

Observation 9 *Consider a solution $\mathfrak{F}$ to an instance (S, P) for SIP.*

(i) $\forall F \in \mathfrak{F}\ (F \cap L = F \cap S)$
(ii) $\forall F_1, F_2 \in \mathfrak{F}\ (F_1 \neq F_2 \ \Rightarrow\ relint(F_1 \cap S) \cap relint(F_2 \cap S) = \emptyset)$.

Definition 10 *Let p and q be two points on the x-axis. Then $[p, q] = \overline{pq}$ and the x-coordinate of q is not smaller than the x-coordinate of p.*

Definition 11 *Let $\mathfrak{F}$ be a solution to an instance (S, P) for SIP and $k = |\mathfrak{F}|$. According to observation 9 the segment S is partitioned into k internally disjoined subsegments, each of which is the intersection of a floodlight from $\mathfrak{F}$ with S.*

(i) Let $S_1, S_2, \ldots, S_k$ denote these subsegments in order they appear on S from left to right.

(ii) Let z_{i-1} and z_i denote the endpoints of S_i, that is $S_i = [z_{i-1}, z_i]$, $i \in \{1, \ldots, k\}$.

(iii) Let F_i denote the floodlight from $\mathfrak{F}$ that contains S_i, $i \in \{1, \ldots, k\}$.

The next three lemmas we give without proof.

Lemma 12 *Let $\mathfrak{F}$ be a solution to an instance (S, P) for SIP, $k = |\mathfrak{F}|$ and $i \in \{1, \ldots, k-1\}$. Then the point $a_{i+1} = a(F_{i+1})$ lies in clockwise direction between z_i and $a_i = a(F_i)$ on the circle $C = C_{a_i}(z_i)$.*

Lemma 13 *Let $I = (S, P)$ be an instance such that in the solution $\mathfrak{F}^*$ to I for SIP* there are two floodlights placed at the point $p \in P$. Then in every solution $\mathfrak{F}$ to I for SIP there exists a floodlight $F \in \mathfrak{F}$ such that $a(F) = p$.*

Lemma 14 *Let $I = (S, P)$ be an instance such that in the solution $\mathfrak{F}^*$ to I for SIP* there are two floodlights placed at the point $p \in P$. Let $\mathfrak{F}$ be a solution to I for SIP and $k = |\mathfrak{F}|$. Then it is $a(F_1) = p$ or $a(F_k) = p$ again using the notation from definition 11.*

Now it is not hard to see that the problem SIP can be solved in polynomial time employing the *dynamic programming* technique. We found an algorithm running in $O(n\log n)$ time and $O(n)$ space which first sorts the points in P according to their y-coordinates and then processes these points in the order obtained from sorting. We omit the details and summarize our results.

Theorem 15 *Let $I = ([a, b], P)$ be an instance and $n = |P|$. A solution to instance I for SIP can be computed in $O(n\log n)$ time using $O(n)$ space in the real RAM model of computation. In the worst case we need $\Omega(n\log n)$ time to find a solution.*

Remark 16 *The lower bound follows from the reduction of SORTING to SIP* presented in [4] and [2].*

4. Illumination of a convex polygon

First we state our illumination problem more formally. Fix a positive integer k. Given a convex polygon P we want to find a set of floodlights $\mathfrak{F}$ such that:

(i) $P = \bigcup_{F \in \mathfrak{F}} ill(F, P)$
(ii) $|\mathfrak{F}| = k$
(iii) $s(\mathfrak{F}) = \sum_{F \in \mathfrak{F}} s(F)$ is minimum
(iv) $\forall k^* \in \{1, \ldots, k-1\} \forall \mathfrak{F}^*$ ($\mathfrak{F}^*$ is a solution to P of $\Pi_{k^*} \Rightarrow s(\mathfrak{F}^*) > s(\mathfrak{F})$)

We will refer to this problem as Π_k and call a set $\mathfrak{F}$ satisfying (i)-(iv) a solution to instance P of Π_k.

Remark 17 *An instance P of Π_k need not have a solution (consider for example an instance P of Π_2 such that P is a triangle). But for such an instance P there exists $k^* < k$ such that P has a solution for problem Π_{k^*} and using more than k^* (but at most k) floodlights does not help to decrease the total size.*

Observation 18 *Let P be an instance of Π_k and $\mathfrak{F}$ a solution to P. Then for every $F \in \mathfrak{F}$ holds $a(F) \in bd(P)$.*

But are there polygons where three floodlights are better than two? The answer is yes. Consider for example the convex polygon with vertices (0,0), (0,2), (10,2), (15,1), (10,0).

Definition 19 *Let P be a convex polygon and $p, q \in bd(P)$ such that $p \neq q$. By $[p, q]$ we denote the closed segment of the boundary of P that consists of those points in $bd(P)$ which we meet by traversing the boundary of P from p to q in clockwise direction. Furthermore we introduce $]p, q[= [p, q] \setminus \{p, q\}$.*

Definition 20 *Let P be a convex polygon and $\mathfrak{F}$ a finite set of floodlights such that $\forall F \in \mathfrak{F}$ ($a(F) \in bd(P)$). Then $cl(P \setminus (\cup_{F \in \mathfrak{F}} F)) = \cup_{Q \in \mathfrak{Q}} Q$ where $\mathfrak{Q}$ is a finite set of convex polygons such that $\forall Q_1, Q_2 \in \mathfrak{Q}$ ($Q_1 \neq Q_2 \Rightarrow Q_1 \cap Q_2 \subseteq bd(P)$). We set $v(cl(P \setminus (\cup_{F \in \mathfrak{F}} F))) = \cup_{Q \in \mathfrak{Q}} v(Q)$.*

The next lemmas tell us something about the structure of solutions. We give them without proof.

Lemma 21 *Let P be a convex polygon, $\mathfrak{F} = \{F_1, F_2, F_3\}$ a solution to P of Π_3, R_i, $i \in \{1, 2, 3\}$, a bounding ray of F_i such that R_1, R_2, R_3 are pairwise nonparallel. Then there is no point $s \in int(P)$ sucht that $s \in R_1 \cap R_2 \cap R_3$.*

Lemma 22 *Let P be a convex polygon and $\mathfrak{F}$ a solution to P of Π_3. Then $v(cl(P \setminus (\cup_{F \in \mathfrak{F}} F))) \cap int(P) = \emptyset$.*

Lemma 23 *Let P be a convex polygon and $\mathfrak{F}$ a solution to P of Π_3. Then $\forall F_1, F_2 \in \mathfrak{F}$ ($F_1 \neq F_2 \Rightarrow int(F_1) \cap int(F_2) = \emptyset$).*

Lemma 24 *Let P be a convex polygon and $\mathfrak{F}$ a solution to P of Π_3. Then we can number the elements of $\mathfrak{F}$ with elements of $\{1,2,3\}$ such that $a(F_1) \in l(F_2)$, $a(F_2) \in l(F_1)$, $a(F_2) \in r(F_3)$ and $a(F_3) \in r(F_2)$.*

Lemma 25 *Let P be a convex polygon and $\mathfrak{F}$ a solution to P of Π_3. Then $\{a(F) : F \in \mathfrak{F}\} \subseteq v(P)$.*

Lemma 26 *Let P be a convex polygon and $\mathfrak{F} = \{F_1, F_2, F_3\}$ a solution to P of Π_3. Let the elements of $\mathfrak{F}$ be numbered as in lemma 24 and let D denote the disk bounded by the circle through the points $a(F_1)$, $a(F_2)$ and $a(F_3)$. Then $[a(F_3), a(F_1)] \subseteq D$.*

Now we are ready to sketch an algorithm for problem Π_3. Given a convex polygon P if there exists a solution $\{F_1, F_2, F_3\}$ to P of Π_3 then from lemma 24 we know the way the floodlights in this solution are arranged. In connection with lemma 26 this suggests the following strategy: For every pair of distinct vertices u and w of P try to find a vertex $v \in]u, w[$ such that $[u, w]$ is contained in the disk bounded by the circle through the points u, v and w. That is we fix a position for $a(F_3)$ (vertex u) and a position for $a(F_1)$ (vertex w) and try to find a position for $a(F_2)$ (vertex v). It should be clear that by pursueing this strategy if there is a solution to P of Π_3 then we will find it. Since we may suppose that solutions to P of Π_1 and Π_2 are available (or we know that there is no solution to P of Π_2), it is also easy to detect those polygons that admit no solution of Π_3.

The question that remains is about an efficient implementation of the above sketched algorithm. We can achieve a running time in $O(n^2)$. But we omit the details and summarize our results.

Theorem 27 *Given a convex polygon P with n vertices we can compute in $O(n^2)$ time and $O(n)$ space a solution to P of Π_3 or find out that there is no such solution using the real RAM model of computation.*

5. Concluding remarks

In our view there are at least two interesting open questions regarding the illumination problem of convex polygons: Are there algorithms for problems Π_2 and Π_3 with running time in $o(n^2)$? Can our results be extended to problems Π_k with $k \geq 4$?

References

[1] P. Bose, L. J. Guibas, A. Lubiw, M. H. Overmars, D. L. Souvaine, and J. Urrutia. The floodlight problem. *International Journal of Computational Geometry & Applications*, 7:153–163, 1997.

[2] F. Contreras. Cutting polygons and a problem on illumination of stages. Master's thesis, University of Ottawa, 1998.

[3] F. Contreras, J. Czyzowicz, E. Rivera-Campo, and J. Urrutia. Optimal floodlight illumination of stages. In *Proc. 14th Annual Symposium on Computational Geometry*, pages 409–410, 1998.

[4] J. Czyzowicz, E. Rivera-Campo, and J. Urrutia. Optimal floodlight illumination of stages. In *Proc. 5th Canadian Conference on Computational Geometry*, pages 393–398, 1993.

[5] V. Estivill-Castro and J. Urrutia. Two-floodlight illumination of convex polygons. In *Proceedings of the Fourth Workshop on Algorithms and Data Structures*, volume 955 of *LNCS*, pages 62–73, 1995.

[6] H. Ito, H. Uehara, and M. Yokoyama. NP-completeness of stage illumination problems. In *Proc. Japan Conference Discrete Computational Geometry*, pages 88–92, 1998.

[7] J. O'Rourke. Open problems in the combinatorics of visibility and illumination. In *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pages 237–243, South Hadley, MA, 1996.

[8] J. Urrutia. Art gallery and illumination problems. In J. R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 973–1027. North Holland, 2000.

Approximate Range Searching Using Binary Space Partitions

Mark de Berg [a] and Micha Streppel [a,1],
[a] *Department of Computer Science, TU Eindhoven, P.O.Box 513, 5600 MB Eindhoven, the Netherlands.*

1. Introduction

Multi-functional data structures and BSP trees. In computational geometry, efficient data structures have been developed for a variety of geometric query problems: range searching, point location, nearest-neighbor searching, etc. The theoretical performance of these structures is often close to the theoretical lower bounds. In order to achieve close to optimal performance, most structures are dedicated to very specific settings. It would be preferable, however, to have a single *multi-functional geometric data structure*: a data structure that can store different types of data and answer various types of queries. Indeed, this is what is often done in practice.

Another potential problem with the structures developed in computational geometry is that they are sometimes rather involved, and that it is unclear how large the constant factors in the query time and storage costs are. Moreover, they may fail to take advantage of the cases where the input objects and/or the query have some nice properties. Hence, the ideal would be to have a multi-functional data structure that is simple and takes advantage of any favorable properties of the input and/or query.

A *binary space partition tree*, or *BSP tree*, is a space-partitioning structure where the subdivision of the underlying space is done in a hierarchical fashion using hyperplanes (that is, lines in case the space is 2D, planes in 3D, etc.) The hierarchical subdivision process usually continues until each cell contains at most one (or maybe a small number of) input object(s). BSP trees are used for many purposes; among these are range searching [1] and hidden surface removal with the painter's algorithm [11].

In some applications—hidden-surface removal is a typical example—the efficiency is determined by the size of the BSP tree. Hence, several researchers have proposed algorithms to construct small BSP trees in various different settings [2,5,13,15]. In this paper we focus on the query complexity of BSP trees.

Approximate range searching. Developing a multi-functional geometric data structure—one that can store any type of object and can do range searching with any type of query range—that provably has good performance seems quite hard, if not impossible. As it turns out, however, such results can be achieved if one is willing to settle for ε*-approximate range searching*, as introduced by Arya and Mount [3].

Here one considers, for a parameter $\varepsilon > 0$, the ε-extended query range Q_ε, which is the set of points lying at distance at most $\epsilon \cdot \mathrm{diam}(Q)$ from Q, where $\mathrm{diam}(Q)$ is the diameter of Q. Objects intersecting Q must be reported, while objects intersecting Q_ε (but not Q) may or may not be reported; objects outside Q_ε are not allowed to be reported. In practice, one would expect that for small values of ε, not too many extra objects are reported.

Our results. In this paper we show that it is possible to construct BSP trees for sets of disjoint segments in the plane, and for low-density scenes in any dimension, whose query time for approximate range searching is as good, or almost as good, as the best known bounds for point data [3,8,9]. More precisely, our results are as follows.

In Section 3 we study BSP trees for a set S of n disjoint line segments in the plane. We give a general technique to convert a BSP tree T_p for a set of points to a BSP tree T_S for S, such that the size of T_S is $O(n \cdot \mathrm{depth}(T_p))$, and the time for range searching remains almost the same.

In Section 4 we then consider low-density scenes.

Email addresses: m.t.d.berg@tue.nl (Mark de Berg), m.w.a.streppel@tue.nl (Micha Streppel).

[1] MS is supported by the Netherlands Organisation for Scientific Research (N.W.O.) under project no. 612.065.203.

We prove that any scene of constant density in $\mathbb{R}^d$ admits a BSP of linear size, such that range-searching queries with arbitrary convex ranges can be answered in $O(\log n + \min_{\varepsilon>0}\{(1/\varepsilon^{d-1}) + k_\varepsilon\})$, where k_ε is the number of objects intersecting Q_ε.

2. Preliminaries

In this section we briefly introduce some terminology and notation that we will use throughout the paper.

A BSP tree for a set S of n objects in $\mathbb{R}^d$ is a binary tree T with the following properties.

- Every (internal or leaf) node ν corresponds to a subset region(ν) of $\mathbb{R}^d$, which we call the *region* of ν. These regions are not stored with the nodes. When ν is a leaf node, we sometimes refer to region(ν) as a *cell*. The root node root(T) corresponds to $\mathbb{R}^d$.
- Every internal node ν stores a hyperplane $h(\nu)$. The left child of ν then corresponds to region(ν) $\cap\, h(\nu)^-$, where $h(\nu)^-$ denotes the half-space below $h(\nu)$, and the right child corresponds to region(ν) $\cap\, h(\nu)^+$, where $h(\nu)^+$ is the half-space above $h(\nu)$.

  A node ν stores, besides the splitting hyperplane $h(\nu)$, a list $L(\nu)$ with all objects contained in $h(\nu)$ that intersect region(ν).
- Every leaf node μ stores a list $L(\mu)$ of all objects in S intersecting the interior of region(μ). In our case the lists have a constant length.

The *size* of a BSP tree is defined as the total number of nodes plus the total size of the lists $L(\nu)$ over all nodes ν in T. Finally, for a node ν in a tree T, we use $T(\nu)$ to denote the subtree of T rooted at ν, and we use depth(T) to denote the depth of T.

3. BSPs for segments in the plane

Let S be a set of n disjoint line segments in the plane. In this section we describe a general technique to construct a BSP for S, based on a BSP on the endpoints of S. The technique uses a segment-tree like approach similar to, but more general than, the deterministic BSP construction of Paterson and Yao [13]. The range-searching structure of Overmars et al.[12] uses similar ideas, except that they store so-called long segments—see below—in an associated structure, so they do not construct a BSP for the segments. The main extra complication we face is that we must ensure that we only work with the relevant portions of the given tree during the recursive construction, and prune away irrelevant portions. The pruning has to be done carefully, however, because too much pruning can have a negative effect on the query time. Next we describe the construction in more detail.

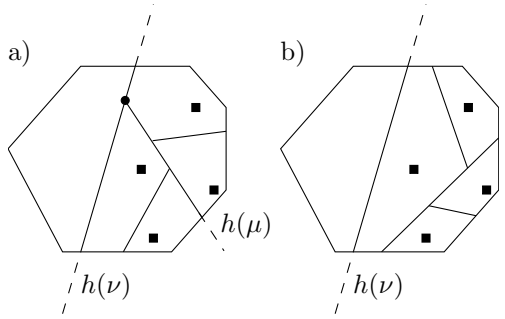


Figure 1. Illustration of the pruning strategy. The black squares indicate input segments. a) There is a T-junction on $h(\nu)$: a splitting line in the subtree ends on $h(\nu)$. Pruning $h(\nu)$ would partition the empty part of the region, which might have a negative effect on the query time. b) There is no T-junction on $h(\nu)$ and $h(\nu)$ can be pruned.

Let P be the set of $2n$ endpoints of the segments in S, and let T_P be a BSP tree for P. We assume that T_P has size $O(n)$, and that the leaves of T_P store at most one point from P. Below we describe the global construction of the BSP tree for S. Some details of the construction will be omitted here.

The BSP tree T_S for S is constructed recursively from T_P, as follows. Let ν be a node in T_P. We call a segment $s \in S$ *short* at ν if region(ν) contains an endpoint of s. A segment s is called *long* at ν if (i) s intersects the interior of region(ν), and (ii) s is short at parent(ν) but not at ν. In a recursive call there are two parameters: a node $\nu \in T_P$ and a subset $S^* \subset S$, clipped to lie within region(ν). The recursive call will construct a BSP tree T_{S^*} for S^* based on $T_P(\nu)$. Initially, $\nu = \text{root}(T_P)$ and $S^* = S$. The recursion stops when S^* is empty, in which case T_{S^*} is a single leaf.

Let $L \subset S^*$ be the set of segments from S^* that are long at ν. The recursive call is handled as follows.

(i) If L is empty, we compute $S_l = S^* \cap h(\nu)^-$ and $S_r = S^* \cap h(\nu)^+$.

  If both S_l and S_r are non-empty, we create a root node for T_{S^*} which stores $h(\nu)$ as

its splitting line. We then recurse on the left child of ν with S_l and on the right child of ν with S_r to construct respectively the left and the right subtree of the root.

If one of S_l and S_r is empty, it seems the splitting line $h(\nu)$ is useless in T_{S^*}. We have to be careful, however, that we do not increase the query time: the removal of $h(\nu)$ can cause other splitting lines, which used to end on $h(\nu)$, to extend further. Hence, we proceed as follows. Define a *T-junction*, see Fig. 1, to be a vertex of the original BSP subdivision induced by T_P . To decide whether or not to use $h(\nu)$, we check if $h(\nu) \cap R$ contains a T-junction in its interior, where R is the region that corresponds to the root of T_{S^*}. If this is the case, we do not prune.

(ii) The second case is when L is not empty. Now the long segments partition $region(\nu)$ into $m := |L| + 1$ regions, $R_1, \ldots, R_m$. We take the following steps.

(i) We split $S^* \setminus L$ into m subsets $S_1^*, \ldots, S_m^*$, where S_i^* contains the segments from S^* lying inside R_i.

(ii) We construct a binary tree T with $m-1$ internal nodes whose splitting lines are the lines containing the long segments. The leaves of T correspond to the regions R_i, and will become the roots of the subtrees to be created for the sets S_i^*. T is balanced by the sizes of the sets S_i^*, as in the trapezoid method for point location [14].

The tree T_{S^*} then consists of the tree T, with, for every $1 \leqslant i \leqslant m$, the leaf of T corresponding to R_i replaced by a subtree for S_i^*. More precisely, each subtree T_i is constructed using a recursive call with node ν and S_i^* as parameters.

The following theorem states the performance of the BSP. Its proof is omitted in this extended abstract.

Theorem 1 *Let R be a family of constant-complexity query ranges in $\mathbb{R}^2$. Suppose that for any set P of n points in $\mathbb{R}^2$, there is a BSP tree T_P of linear size, where each leaf stores at most one point from P, with the following property: any query with a range from R intersects at most $v(n,k)$ cells in the BSP subdivision, where k is the number of points in the query range. Then for any set S of n disjoint segments in $\mathbb{R}^2$, there is a BSP tree T_S such that*

(i) the depth of T_S is $O(depth(T_P))$

(ii) the size of T_S is $O(n \cdot depth(T_P))$

(iii) any query with a range from R visits at most $O((v(n,k)+k) \cdot depth(T_P))$ nodes from T_S, where k is the number of segments intersecting the range.

The BSP tree T_S can be constructed in $O(n \cdot (depth(T_P))^2)$ time.

Several of the known data structures for range searching in point sets are actually BSP trees. Examples are ham-sandwich trees [10], kd-trees [7], and BAR-trees [8,9]. Here we focus on the application using BAR-trees, as it gives good bounds for approximate range searching for line segments in the plane.

One can construct BAR-trees with logarithmic depth, such that the number of leaves visited by a query with a convex query range Q is bounded by $O((1/\varepsilon) + k_\varepsilon)$, where k_ε is the number of points inside the extended query range Q_ε.

Combining this with theorem 1 (and a specialized construction algorithm that speeds up the construction time by a logarithmic factor) we get the following results.

Corollary 2 *Let S be a set of n disjoint segments in $\mathbb{R}^2$. In $O(n \log n)$ time one can construct a BSP tree for S of size $O(n \log n)$ and depth $O(\log n)$ such that a range query with a constant-complexity convex range can be answered in time $O(\min_{\varepsilon>0}\{(1/\varepsilon)\log n + k_\varepsilon \log n\})$, where k_ε is the number of segments intersecting Q_ε.*

4. BSPs for low-density scenes

Let S be a set of n objects $\mathbb{R}^d$. For an object o, we use $\rho(o)$ to denote the radius of the smallest enclosing ball of o. The *density* of a set S is the smallest number λ such that the following holds: any ball B is intersected by at most λ objects $o \in S$ with $\rho(o) \geqslant \rho(B)$ [6]. If S has density λ, we call S a λ-low-density scene. In this section we show how to construct a BSP tree for S that has linear size and very good performance for approximate range searching if the density of S is constant. Our method combines ideas from de Berg [5] with the BAR-tree of Duncan et al.[9] Our overall strategy, also used by de Berg [5], is to compute a suitable

set of points that will guide the construction of the BSP tree. Unlike in [5], however, we cannot use the bounding-box vertices of the objects in S for this, because that does not work in combination with a BAR-tree. What we need is a set G of points with the following property: any cell in a BAR-tree that does not contain a point from G in its interior is intersected by at most κ objects from S, for some constant κ. We call such a set G a *κ-guarding set* [4] against BAR-tree cells, and we call the points in G *guards*.

The algorithm is as follows.

(i) Construct a κ-guarding set G for S, as explained below. The construction of the guarding set is done by generating $O(1)$ guards for each object $o \in S$, so that the guards created for any subset of the objects will form a κ-guarding set for that subset. Using the results of [4] it can be shown that there exists a guarding set for λ-low-density scenes against BAR-tree cells. Using special properties of (corner-cut) BAR-tree cells a guarding set of size $12n$ can be given for the planar case. We will use object(g) to denote the object for which a guard $g \in G$ was created.

(ii) Create a BAR-tree T on the set G using the algorithm of Duncan et al. [9], with the following adaptation: whenever a recursively call is made with a subset $G^* \subset G$ in a region R, we delete all guards g from G^* for which object(g) does not intersect R. This pruning step, which was not needed in [5], is essential to guarantee a bound on the query time. This leads to a BSP tree whose cells can only contain guards whose corresponding objects do not intersect the cell.

(iii) Search with each object $o \in S$ in T to determine which leaf cells are intersected by o. Store with each leaf the set of all intersected objects. Let T_S be the resulting BSP tree.

The following theorem states the performance of the BSP. Its proof is omitted in this extended abstract.

Theorem 3 *Let S be a λ-low-density scene consisting of n objects in $\mathbb{R}^d$. There exists a BSP tree T_S for S such that*

(i) the depth of T_S is $O(\log n)$

(ii) the size is $O(\lambda n)$

(iii) a query range with a convex range Q visits takes $O(\log n + \lambda \cdot \min_{\varepsilon>0}\{(1/\varepsilon) + k_\varepsilon\})$ time, where k_ε is the number of objects intersecting the extended query range Q_ε.

The BSP tree can be constructed in $O(\lambda n \log n)$ time.

References

[1] P.K. Agarwal and J. Erickson. Geometric range searching and its relatives. In: B. Chazelle, J. Goodman, and R. Pollack (Eds.), *Advances in Discrete and Computational Geometry*, Vol. 223 of *Contemporary Mathematics*, pages 1–56, American Mathematical Society, 1998.

[2] P.K. Agarwal, E. Grove, T.M. Murali and J.S. Vitter. Binary space partitions for fat rectangles. *SIAM J. Comput.* 29:1422-1448, 2000.

[3] A. Arya, D. Mount, Approximate range searching, Comput. Geom. Theory Appl. 17 (2000) 135–152.

[4] M. de Berg, H. David, M. J. Katz, M. Overmars, A. F. van der Stappen, and J. Vleugels. Guarding scenes against invasive hypercubes. *Comput. Geom.*, 26:99–117, 2003.

[5] M. de Berg. Linear size binary space partitions for uncluttered scenes. *Algorithmica* 28:353–366, 2000.

[6] M. de Berg, M.J. Katz, A. F. van der Stappen, and J. Vleugels. Realistic input models for geometric algorithms. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 294–303, 1997.

[7] M. de Berg, M. van Kreveld, M. Overmars, and O. Cheong. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 1997.

[8] C.A. Duncan, Balanced Aspect Ratio Trees, Ph.D. Thesis, John Hopkins University, 1999.

[9] C.A. Duncan, M.T. Goodrich, S.G. Kobourov, Balanced aspect ratio trees: Combining the advantages of k-d trees and octrees, In *Proc. 10th Ann. ACM-SIAM Sympos. Discrete Algorithms*, pages 300–309, 1999.

[10] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, 1987.

[11] H. Fuchs, Z. M. Kedem, and B. Naylor. On visible surface generation by a priori tree structures. *Comput. Graph.*, 14(3):124–133, 1980. Proc. SIGGRAPH '80.

[12] M.H. Overmars, H. Schipper, and M. Sharir. Storing line segments in partition trees. *BIT*, 30:385–403, 1990

[13] M. S. Paterson and F. F. Yao. Efficient binary space partitions for hidden-surface removal and solid modeling. *Discrete Comput. Geom.*, 5:485–503, 1990.

[14] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.

[15] C.D. Tóth. Binary Space Partitions for Line Segments with a Limited Number of Directions. *SIAM J. Comput.* 32:307–325, 2003.

Straight Line Skeleton in Linear Time, Topologically Equivalent to the Medial Axis

Mirela Tănase Remco C. Veltkamp
Institute of Information and Computing Sciences, Utrecht University, The Netherlands
{mirela, Remco.Veltkamp}@cs.uu.nl

1. Introduction

Skeletons have long been recognized as an important tool in computer graphics, computer vision and medical imaging. The most known and widely used skeleton is the *medial axis* [1]. However, the presence of parabolic arcs may be a disadvantage for the medial axis. The straight line skeleton [2], on the other hand, is composed only of line segments. In [3] a polygon decomposition based on the straight line skeleton was presented. The results obtained from the implementation indicated that this technique provides plausible decompositions for a variety of shapes. However, sharp reflex angles have a big impact on the form of the straight line skeleton, which in turn has a large effect on the decomposition (see figure 1).

This paper presents the *linear axis*, a new skeleton for polygons. It is related to the medial axis and the straight line skeleton, being the result of a wavefront propagation process. The wavefront is linear and propagates by translating edges at constant speed. The initial wavefront is an altered version of the original polygon: zero-length edges are added at reflex vertices. The linear axis is a subset of the straight line skeleton of the altered polygon. In this way, the counter-intuitive effects in the straight line skeleton caused by sharp reflex vertices are alleviated. We introduce the notion of ε-equivalence between two skeletons, and give an algorithm that computes the number of hidden edges for each reflex vertex which yield a linear axis ε-equivalent to the medial axis. This linear axis and thus the straight line skeleton can then be computed from the medial axis in linear time for non-degenerate polygons, i.e. polygons with a constant number of "nearly co-circular" sites.

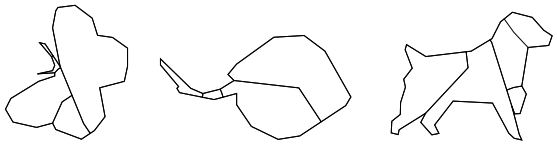


Fig. 1. A decomposition [3] based on split events of the straight line skeleton gives counter-intuitive results if the polygon contains sharp reflex vertices.

2. Linear axis

Let P be a simple, closed polygon. The set of sites defining the Voronoi diagram $VD(P)$ of P is the set of line segments and reflex vertices of P. The set of points $\mathcal{U}(d)$ inside P, having some fixed distance d to the polygon is called a *uniform wavefront*. As the distance d increases, the wavefront points move at equal, constant velocity along the normal direction. This process is called *uniform wavefront propagation*. During the propagation, the breakpoints between the line segments and circular arcs in $\mathcal{U}(d)$ trace the Voronoi diagram $VD(P)$. The medial axis $M(P)$ is a subset of $VD(P)$; the Voronoi edges incident to the reflex vertices are not part of the medial axis (see figure 2(a)).

The straight line skeleton is also defined as the trace of adjacent components of a propagating wavefront. The wavefront is linear and is obtained by translating the edges of the polygon in a self-parallel manner, keeping sharp corners at reflex vertices (edges incident to a reflex vertex will grow in length, see figure 2(b)). This also means that points in the wavefront move at different speeds: wavefront points originating from reflex vertices move faster than points originating from the edges incident to those vertices. In fact, the speed of the wavefront points originating from a reflex vertex gets arbitrary large when the internal angle of the

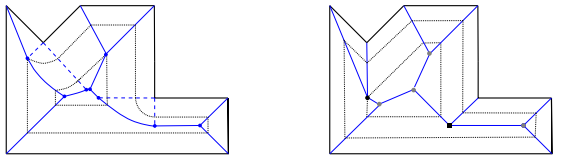


Fig. 2. Medial Axis (a) vs. Straight Line Skeleton (b). Instances of the propagating wavefront generating the skeletons are drawn with dotted line style in both cases. In (a) the Voronoi edges that are not part of the medial axis are drawn with dashed line style.

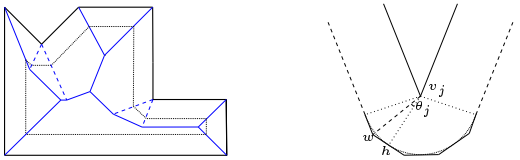


Fig. 3. (a) The linear axis in the case when one hidden edge is inserted at each reflex vertex. A linear wavefront is drawn in dotted line style; the dashed lines are the bisectors that are not part of the linear axis. (b) The linear offset (solid line) of a reflex vertex with 3 associated edges.

vertex gets arbitrary close to 2π. And that is why sharp reflex vertices have such a big impact on the form of the straight line skeleton. The straight line skeleton $SLS(P)$ of P is the trace in the propagation of the vertices of the wavefront.

Let $\{v_1, v_2, \ldots, v_n\}$ denote the vertices of a simple polygon P and let $\kappa = (k_1, k_2, \ldots, k_n)$ be a sequence of natural numbers. If v_i is a convex vertex of P, $k_i = 0$, and if it is reflex vertex, $k_i \geq 0$. Let $\mathcal{P}^\kappa(0)$ be the polygon obtained from P by replacing each reflex vertex v_i with $k_i + 1$ identical vertices. We assume that these vertices are the endpoints of k_i zero-length edges, which will be referred to as the *hidden edges* associated with v_i. The directions of the hidden edges are chosen such that the reflex vertex v_i of P is replaced in $\mathcal{P}^\kappa(0)$ by $k_i + 1$ "reflex vertices" of equal internal angle. The polygon $\mathcal{P}^\kappa(t)$ represents the linear wavefront, corresponding to a sequence κ of hidden edges, at moment t. The propagation consists of translating edges at constant unit speed, in a self-parallel manner.

Definition 1 *The **linear axis** $L^\kappa(P)$ of P, corresponding to a sequence κ of hidden edges, is the trace of the convex vertices of the linear wavefront $\mathcal{P}^\kappa$ in the above propagation process.*

Obviously, $L^\kappa(P)$ is a subset of $SLS(\mathcal{P}^\kappa(0))$; we only have to remove the bisectors traced by the reflex vertices of the wavefront (see figure 2 (a)).

Lemma 1 *If any reflex vertex v_j of internal angle $\alpha_j \geq 3\pi/2$ has at least one associated hidden edge, then no new reflex vertices are introduced to the wavefront during the propagation of $\mathcal{P}^\kappa$.*

For the rest of this paper, we will assume that each selection κ of hidden edges that is considered, satisfies the condition in lemma 1. This ensures that $L^\kappa(P)$ is a connected graph.

A *site* of P is a line segment or a reflex vertex. If S is an arbitrary site of P, we denote by $\mathcal{P}^\kappa_S(t)$ the points in $\mathcal{P}^\kappa(t)$ originating from S. We call $\mathcal{P}^\kappa_S(t)$ the *(linear) offset of S at moment t.*

Individual points in $\mathcal{P}^\kappa$ move at different speeds. The fastest moving points in $\mathcal{P}^\kappa$ are its reflex vertices. The slowest moving points have unit speed. Let v_j be a reflex vertex of P, of internal angle α_j, and with k_j associated hidden edges. Let $\mathcal{P}^\kappa_{v_j}(t)$ be the offset of v_j at some moment t.

Lemma 2 *Points in $\mathcal{P}^\kappa_{v_j}(t)$ move at speed at most*

$$s_j = 1/\cos\left((\alpha_j - \pi)/(2 * (k_j + 1))\right).$$

Each convex vertex of $\mathcal{P}^\kappa(t)$ is a breakpoint between two linear offsets. Lemmas 3 and 4 describe the trace of the intersection of two adjacent offsets in the linear wavefront propagation. If v is a vertex and e is an edge non-incident with v, we denote by $P(v, e)$ the parabola with focus v and directrix the line supporting e. For any real value $r > 1$, we denote by $H^r(v, e)$ the locus of points whose distances to v and the line supporting e are in constant ratio r. This locus is a hyperbola branch. If u and v are reflex vertices, we denote by $C^r(u, v)$ the Apollonius circle associated with u, v and ratio $r \neq 1$. $C^r(u, v)$ is the locus of points whose distances to u and v, respectively, are in constant ratio r.

Let e be an edge and v_j a reflex vertex of P. The points in the offset $\mathcal{P}^\kappa_e$ move at unit speed, the points in the offset $\mathcal{P}^\kappa_{v_j}$ move at speeds varying in the interval $[1, s_j]$, where s_j is given by lemma 2.

Lemma 3 *If the linear offsets of v_j and e become adjacent, the trace of their intersection is a polygonal chain that satisfies:*
1) it lies in the region between $P(v_j, e)$ and $H^{s_j}(v_j, e)$;
2) the lines supporting its segments are tangent to $P(v_j, e)$; the tangent points lie on the traces of the unit speed points in $\mathcal{P}^\kappa_{v_j}$;
3) $H^{s_j}(v_j, e)$ passes through those vertices which lie on the trace of a reflex vertex of $\mathcal{P}^\kappa_{v_j}$.

Let v_i and v_j be reflex vertices of P. The points

in the offset $\mathcal{P}^{\kappa}_{v_i}$ move at speeds varying in the interval $[1, s_i]$ while the points in the offset $\mathcal{P}^{\kappa}_{v_j}$ move at speeds varying in the interval $[1, s_j]$.

Lemma 4 *If the linear offsets of v_i and v_j become adjacent, the trace of their intersection is a polygonal chain that lies outside the Apollonius circles $C^{s_i}(v_i, v_j)$ and $C^{s_j}(v_j, v_i)$.*

3. Linear Axis vs. Medial Axis

Obviously, the larger the number of hidden edges associated with the reflex vertices, the closer the corresponding linear axis approximates the medial axis. For many applications of the medial axis, an approximation that preserves the main visual cues of the shape, though non-isomorphic with the medial axis, is perfectly adequate. The way we now define the ε-equivalence between the medial axis and the linear axis, will allow us to compute a linear axis that closely approximates the medial axis using only a small number of hidden edges.

Let $\varepsilon \geq 0$; an ε-edge is a Voronoi edge generated by four almost co-circular sites. Let b_ib_j be a Voronoi edge, with b_i equally distanced to sites S_k, S_i, S_l, and b_j equally distanced to S_k, S_j, S_l.

Definition 2 *The edge b_ib_j is an **ε-edge** if $d(b_i, S_j) < (1+\varepsilon)d(b_i, S_i)$ or $d(b_j, S_i) < (1+\varepsilon)d(b_j, S_j)$.*

A path between two nodes of $M(P)$ is an *ε-path* if it is made only of ε-edges. For any node b of M, a node b' such that the path between b and b' is an ε-path, is called an *ε-neighbour* of b. Let $N_\varepsilon(b)$ be the set of ε-neighbours of b. The set $\{b\} \cup N_\varepsilon(b)$ is called an *ε-cluster*.

Let (V_M, E_M) be the graph induced by $M(P)$ on the set of vertices V_M composed of the convex vertices of P and the nodes of degree 3 of $M(P)$. Let $(V_{L^\kappa}, E_{L^\kappa})$ be the graph induced by $L^\kappa(P)$ on the set of vertices V_{L^κ} composed of the convex vertices of P and the nodes of degree 3 of $L^\kappa(P)$.

Definition 3 *$M(P)$ and $L^\kappa(P)$ are **ε-equivalent** if there exists a surjection $f : V_M \to V_{L^\kappa}$ so that:*
i) $f(p) = p$, for all convex p of P;
ii) $\forall\, b_i, b_j \in V_M$ with $b_j \notin N_\varepsilon(b_i)$, $\exists$ an arc in E_M connecting b_i and $b_j \Leftrightarrow \exists$ an arc in E_{L^κ} connecting $f(b'_i)$ and $f(b'_j)$ where $b'_i \in \{b_i\} \cup N_\varepsilon(b_i)$ and $b'_j \in \{b_j\} \cup N_\varepsilon(b_j)$.

The following lemma gives a sufficient condition for the ε-equivalence of the two skeletons. The path between two disjoint Voronoi cells $VC(S_i)$ and $VC(S_j)$ is the shortest path in $M(P)$ between a point of $VC(S_i) \cap M(P)$ and a point of $VC(S_j) \cap M(P)$.

Lemma 5 *If the only sites whose linear offsets become adjacent in the propagation, are sites with adjacent Voronoi cells or sites whose path between their Voronoi cells is an ε-path, then the linear axis and the medial axis are ε-equivalent.*

We now present an algorithm for computing a sequence κ of hidden edges such that the resulting linear axis is ε-equivalent to the medial axis. The algorithm handles pairs of sites whose linear offsets must be at any moment disjoint in order to ensure the ε-equivalence of the two skeletons. These are sites with disjoint Voronoi cells and whose path between these Voronoi cells is not an ε-path (see lemma 5). However, we do not have to consider each such pair. The algorithm actually handles the pairs of *conflicting sites*, where two sites S_i and S_j (at least one being a reflex vertex) are said to be conflicting if the path between their Voronoi cells contains exactly one non-ε edge. When handling the pair S_i, S_j we check and, if necessary, adjust the maximal speeds s_i and s_j of the offsets of S_i and S_j, respectively, so that these offsets remain disjoint in the propagation. This is done in the subroutine HandleConflictingPair by looking locally at the configuration of the uniform wavefront and using the localization constraints for the edges of the linear axis given by lemmas 3 and 4.

The algorithm for computing a sequence κ of hidden edges can be summarized as follows.

Algorithm ComputeHiddenEdges
Input A simple polygon P and its medial axis.
Output A number of hidden edges for each reflex vertex.

1. For each reflex vertex S_j of P
   if $\alpha_j \geq 3\pi/2$ **then** $s_j \leftarrow 1/\cos((\alpha_j - \pi)/4)$
   else $s_j \leftarrow 1/\cos((\alpha_j - \pi)/2)$
2. Compute all pairs of conflicting sites
3. For each pair of conflicting sites S_i, S_j
   HandleConflictingPair(S_i,S_j)
4. For each reflex vertex S_j of P
   $k_j \leftarrow \lceil (\alpha_j - \pi)/(2\cos^{-1}(1/s_j)) \rceil$.

Theorem 6 *Algorithm ComputeHiddenEdges computes a sequence of hidden edges that leads to a linear axis ε-equivalent to the medial axis.*

The performance of this algorithm depends on the number of conflicting pairs. This in turn depends on the number of nodes in the ε-clusters of $M(P)$. If any ε-cluster of M has a constant number of nodes, there are only a linear number of con-

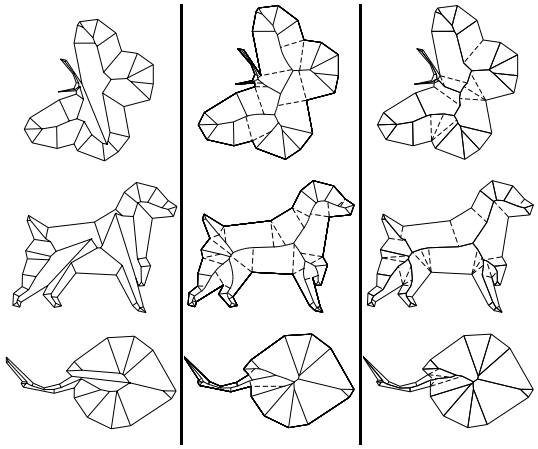


Fig. 4. A comparison of the linear axis (right) with the medial axis (middle) and the straight line skeleton (left). The skeletons are drawn in solid line style.

flicting pairs. Each conflicting pair is handled in constant time [4], thus in this case, ComputeHiddenEdges computes the sequence κ in linear time. There is only a limited class of shapes with a constant number of clusters, such that each has a linear number of nodes.

Once we have a sequence κ that ensures the ε-equivalence between the corresponding linear axis and the medial axis, we can construct this linear axis. The medial axis can be computed in linear time [5]. Despite its similarity to the medial axis, the fastest known algorithms [6] [7] for the straight line skeleton are slower. Any of these algorithms can be used to compute the straight line skeleton of $\mathcal{P}^\kappa(0)$. The linear axis $L^\kappa(P)$ corresponding to the sequence κ is then obtained from $SLS(\mathcal{P}^\kappa(0))$ by removing the bisectors incident to the reflex vertices of P.

However, if M has only ε-clusters of constant size, $L^\kappa(P)$ can be computed from the medial axis in linear time by adjusting the medial axis. In computing the linear axis, we adjust each non-ε-edge of the medial axis to its counterpart in the linear axis. When adjusting an edge b_ib_j we first adjust the location of its endpoints to the location of the endpoints of its counterpart. If node b_i is part of an ε-cluster, we compute first the counterparts of the nodes in this cluster based on a local reconstruction of the linear wavefront. The adjustment of a node's location is done in constant time, if its ε-cluster has constant size, i.e. when the polygon is non-degenerate. Finally, we use lemmas 3-4 to replace the parabolic arc or the perpendicular bisector with the corresponding chain of segments.

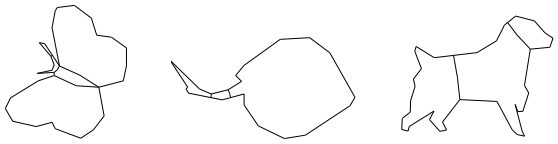


Fig. 5. A decomposition based on split events of the linear axis gives natural results even if the polygon contains sharp reflex vertices.

Theorem 7 *For a non-degenerate polygon P, the straight line skeleton of $\mathcal{P}^\kappa(0)$ can be computed in linear time.*

We have implemented the algorithm ComputeHiddenEdges and the algorithm that constructs the linear axis from the medial axis. Figure 4 illustrates the straight line skeleton (left column), medial axis (middle column) and the linear axis (right column) of three contours. We also used the linear axis instead of the straight line skeleton to decompose the contours in figure 1 based on wavefront split events [3]. The results are presented in figure 5; we see that the unwanted effects of the sharp reflex vertices are eliminated and the results of this first step in the decomposition look more natural.

References

[1] H. Blum, A Transformation for Extracting New Descriptors of Shape, *Symposium Models for Speech and Visual Form.* (ed: W. Wathen-Dunn. MIT Press, 1967) 362–381.

[2] O. Aichholzer, F. Aurenhammer, Straight Skeletons for General Polygonal Figures in the Plane, in: *Proc. 2nd International Computing and Combinatorics Conference COCOON '96.* 117–126.

[3] M. Tănase, R.C. Veltkamp, Polygon Decomposition based on the Straight Line Skeleton, in: *Proc. 19th ACM Symposium on Computational Geometry.* (2003) 58–67.

[4] M. Tănase, R.C. Veltkamp, Straight Line Skeleton in Linear Time, Topologically Equivalent to the Medial Axis, TR (2004).

[5] F. Chin, J. Snoeyink, C.-A. Wang, Finding the Medial Axis of a Simple Polygon in Linear Time. *Discrete Computational Geometry* **21(3)** (1999) 405–420.

[6] D. Eppstein, J. Erickson, Raising Roofs, Crashing Cycles, and Playing Pool: Applications of a Data Structure for Finding Pairwise Interactions, *Discrete and Computational Geometry* **22(4)** (1999) 569–592.

[7] S.-W. Cheng, A. Vigneron, Motorcycle Graphs and Straight Skeletons, in: *Proc. 13th ACM-SIAM Symp. Discrete Algorithms* (2002) 156–165.

Partitioning Orthogonal Polygons by Extension of All Edges Incident to Reflex Vertices: lower and upper bounds on the number of pieces ⋆

António Leslie Bajuelos [a], Ana Paula Tomás [*,b] and Fábio Marques [c]

[a] *Department of Mathematics & CEOC - Center for Research in Optimization and Control, University of Aveiro, Portugal*
[b] *DCC-FC & LIACC, University of Porto, Portugal*
[c] *School of Technology and Management, University of Aveiro, Portugal*

Abstract

Given an orthogonal polygon P, let $|\Pi(P)|$ be the number of rectangles that result when we partition P by extending the edges incident to reflex vertices towards $\mathrm{INT}(P)$. In [4] we showed that $|\Pi(P)| \leq 1 + r + r^2$, where r is the number of reflex vertices of P. We shall now give sharper bounds both for $\max_P |\Pi(P)|$ and $\min_P |\Pi(P)|$. Moreover, we characterize the structure of orthogonal polygons in general position for which these new bounds are exact.

Key words: Orthogonal Polygons, Decomposition, Rectilinear Cut, Square Grid.

1. Introduction

We shall call *simple polygon* P a region of a plane enclosed by a finite collection of straight line segments forming a simple cycle. This paper deals only with simple polygons, so that we call them just polygons, in the sequel. We will denote the interior of the polygon P by $\mathrm{INT}(P)$ and the boundary by $\mathrm{BND}(P)$. The boundary shall be considered part of the polygon, that is $P = \mathrm{INT}(P) \cup \mathrm{BND}(P)$. A vertex is called *convex* if the interior angle between its two incident edges is at most π; otherwise it is called *reflex* (or *concave*). We use r to represent the number of reflex vertices of P. A polygon is called *orthogonal* (or *rectilinear*) iff its edges meet at right angles. O'Rourke [2] has shown that $n = 2r + 4$ for every n-vertex orthogonal polygon (n-*ogon*, for short).

⋆ (Extended abstract) Partially funded by LIACC through *Programa de Financiamento Plurianual, Fundação para a Ciência e Tecnologia (FCT)* and *Programa POSI*, and by CEOC (Univ. of Aveiro) through *Programa POCTI, FCT, co-financed by EC fund FEDER.*

* Corresponding author

Email addresses: leslie@mat.ua.pt (António Leslie Bajuelos), apt@ncc.up.pt (Ana Paula Tomás), fabio@estga.ua.pt (Fábio Marques).

Definition 1 *A rectilinear cut of an n-ogon P is obtained by extending each edge incident to a reflex vertex of P towards* $\mathrm{INT}(P)$ *until it hits* $\mathrm{BND}(P)$. *We denote this partition by* $\Pi(P)$ *and the number of its elements (pieces) by* $|\Pi(P)|$. *Each piece is a rectangle, so that we call it* r-piece.

Generic n-ogons may be obtained from a particular kind of n-ogons – the so-called *grid orthogonal polygons* [3], as illustrated in Fig. 1 (The reader may skip Definition 2 and Lemmas 3 and 4 if he/she has already read [3].)

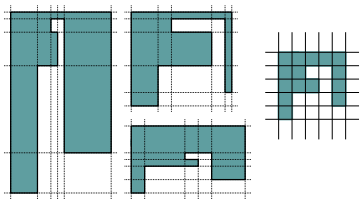


Fig. 1. Three 12-ogons mapped to the same grid 12-ogon.

Definition 2 *An n-ogon P is* in general position *iff P has no collinear edges. We call "grid n-ogon" each n-ogon in general position defined in a $\frac{n}{2} \times \frac{n}{2}$ square grid.*

Lemma 3 follows immediately from this definition.

Lemma 3 *Each grid n-ogon has exactly one edge in every line of the grid.*

Each n-ogon not in general position may be mapped to an n-ogon in general position by ϵ-perturbations, for a sufficiently small constant $\epsilon > 0$. Consequently, we shall first address n-ogons in general position.

Lemma 4 *Each n-ogon in general position is mapped to a unique grid n-ogon through top-to-bottom and left-to-right sweeping. And, reciprocally, given a grid n-ogon we may create an n-ogon that is an instance of its class by randomly spacing the grid lines in such a way that their relative order is kept.*

The number of classes may be further reduced if we group grid n-ogons that are symmetrically equivalent. In this way, the grid n-ogons in Fig. 2 represent the same class.

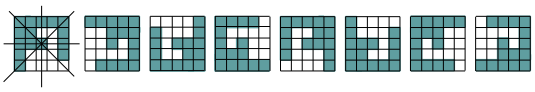


Fig. 2. Eight grid n-ogons that are symmetrically equivalent. From left to right, we see images by clockwise rotations of 90°, 180° and 270°, by flips wrt horizontal and vertical axes and flips wrt positive and negative diagonals.

Definition 5 *Given an n-ogon P in general position, GRID(P) denotes any grid n-ogon in the class that contains the grid n-ogon to which P is mapped by the sweep procedure described in Lemma 4*

The following result is a trivial consequence of the definition of GRID(P).

Lemma 6 *For all n-ogons P in general position, $|\Pi(P)| = |\Pi(GRID(P))|$.*

2. Lower and Upper bounds on $|\Pi(P)|$

In [4] we showed that $\Pi(P)$ has at most $1+r+r^2$ pieces. Later we noted that this upper bound is not sufficiently tightened. Actually, for small values of r, namely $r = 3, 4, 5, 6, 7$, we experimentally found that the difference between $1 + r + r^2$ and $\max|\Pi(P)|$ was 1, 2, 4, 6 and 9, respectively.

Definition 7 *A grid n-ogon Q is called* FAT *iff $|\Pi(Q)| \geq |\Pi(P)|$, for all grid n-ogons P. Similarly, a grid n-ogon Q is called* THIN *iff $|\Pi(Q)| \leq |\Pi(P)|$, for all grid n-ogons P.*

The experimental results supported our conjecture that there was a single FAT n-ogon (except for symmetries of the grid) and that it had the form illustrated in Fig. 3.

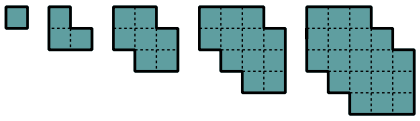


Fig. 3. The unique FAT n-ogons (symmetries excluded), for $n = 4, 6, 8, 10, 12$.

Clearly, each piece r-piece is defined by four vertices. Each vertex is either in INT(P) (*internal vertex*) or is in BND(P) (*boundary vertex*). Similar definitions hold for edges. An edge e of r-piece R is called an *internal edge* if $e \cap \text{INT}(P) \neq \emptyset$, and it is called *boundary edge* otherwise.

Lemma 8 *The total number $|V_i|$ of internal vertices in $\Pi(P)$, when the grid n-ogon P is as illustrated in Fig. 3 is given by (1)*

$$|V_i| = \begin{cases} \dfrac{3r^2 - 2r}{4}, & \text{for } r \text{ even} \\ \dfrac{(3r+1)(r-1)}{4}, & \text{for } r \text{ odd} \end{cases} \quad (1)$$

where r is the number of reflex vertices of P.

PROOF. In case r is even,

$$|V_i| = 2\sum_{k=1}^{\frac{r}{2}}(r-k)$$

and in case r is odd

$$|V_i| = (r - \frac{r+1}{2}) + 2\sum_{k=1}^{\frac{r-1}{2}}(r-k)$$

□

Proposition 9 *Every n-vertex orthogonal polygon P such that the number of internal vertices of $\Pi(P)$ is given by (1) has at most a single reflex vertex in each horizontal and vertical line.*

PROOF. We shall suppose first that P is a grid n-ogon. Then, let $v_{L_1} = (x_{L_1}, y_{L_1})$ and $v_{R_1} = (x_{R_1}, y_{R_1})$ be leftmost and rightmost reflex vertices of P, respectively. The horizontal chord with origin at v_{L_1} can intersect at most $x_{R_1} - x_{L_1}$ vertical chords, since we shall not count the intersection with the vertical chord defined by v_{L_1}. The same may be said about the the horizontal chord with origin at v_{R_1}. There are exactly r vertical and r horizontal chords, and thus $x_{R_1} - x_{L_1} \leq r - 1$. If there were c vertical edges such that both extreme points are reflex vertices then $x_{R_1} - x_{L_1} \leq r - 1 - c$.

This would imply that the number of internal vertices of $\Pi(P)$ would be strictly smaller than the value defined by (1). Indeed, we could proceed to consider the second leftmost vertex (for $x > x_{L_1}$), say v_{L_2}, then the second rightmost vertex (with $x < x_{R_1}$) and so forth. The horizontal chord that v_{L_2} defines either intersects only the vertical chord defined by v_{L_1} or it does not intersect it at all. So, it intersects at most $r-2-c$ vertical chords. In sum, c should be null, and by symmetry, we would conclude that there is exactly a reflex vertex in each vertical grid line (for $x > 1$ and $x < \frac{n}{2} = r + 2$).

Now, if P is not a grid n-ogon but is in general position, then $\Pi(P)$ has the same combinatorial structure as $\Pi(GRID(P))$, so that we do not have to prove anything more.

If P is not in general position, then let we render it in general position by a sufficiently small ϵ-perturbation, so that the partition of this latter polygon would not have less internal vertices than $\Pi(P)$. □

Corollary 10 *For all grid n-ogons P, the number of internal vertices of $\Pi(P)$ is less than or equal to the value established by (1).*

PROOF. It results from the proof of Proposition 9. □

Theorem 11 *Let P be a grid n-ogon, $r = \frac{n-4}{2}$ the number of its reflex vertices. If P is* FAT *then*

$$|\Pi(P)| = \begin{cases} \dfrac{3r^2+6r+4}{4}, & \textit{for } r \textit{ even} \\ \dfrac{3(r+1)^2}{4}, & \textit{for } r \textit{ odd} \end{cases}$$

and if P is THIN then $|\Pi(P)| = 2r + 1$.

PROOF. Suppose that P is a grid n-ogon. Let V, E and F be the sets of all vertices, edges and faces of $\Pi(P)$, respectively. Let us denote by V_i and V_b the sets of all internal and boundary vertices of the pieces of $\Pi(P)$. Similarly, E_i and E_b represent the sets of all internal and boundary edges of such pieces. Then, $V = V_i \cup V_b$ and $E = E_i \cup E_b$. Being P in general position, each chord we draw to form $\Pi(P)$ hits BND(P) in the interior of an edge and no two chords hit BND(P) in the same point. Hence, using O'Rourke's formula [2] we obtain $|E_b| = |V_b| = (2r + 4) + 2r = 4r + 4$. It is easily seen that to obtain a FAT n-ogon we must maximize the number of internal vertices.

By Corollary 10,

$$\max_P |V_i| = \begin{cases} \dfrac{3r^2-2r}{4}, & \text{for } r \text{ even} \\ \dfrac{(3r+1)(r-1)}{4}, & \text{for } r \text{ odd} \end{cases}$$

and, therefore, $\max_P |V| = \max_P(|V_i| + |V_b|)$ is given by

$$\max_P |V| = \begin{cases} \dfrac{3r^2+14r+16}{4}, & \text{for } r \text{ even} \\ \dfrac{3r^2+14r+15}{4}, & \text{for } r \text{ odd} \end{cases}$$

From Graph Theory [1] we know that the sum of the degrees of vertices in a graph is twice the number of its edges, that is, $\sum_{v\in V} \delta(v) = 2|E|$. Using the definitions of grid n-ogon and of $\Pi(P)$, we may partition V as

$$V = V_c \cup V_r \cup (V_b \setminus (V_c \cup V_r)) \cup V_i$$

V_r and V_c representing the sets of reflex and of convex vertices of P, respectively. Moreover, we may conclude that $\delta(v) = 4$ for all $v \in V_r \cup V_i$, $\delta(v) = 3$ for all $v \in V_b \setminus (V_c \cup V_r)$ and $\delta(v) = 2$ for all $v \in V_c$. Hence,

$$\begin{aligned} 2|E| &= \sum_{v\in V_r\cup V_i} \delta(v) + \sum_{v\in V_c} \delta(v) + \sum_{v\in V_b\setminus(V_c\cup V_r)} \delta(v) \\ &= 4|V_i| + 4|V_r| + 2|V_c| + 3(|V_b| - |V_r| - |V_c|) \\ &= 4|V_i| + 12r + 8 \end{aligned}$$

and, consequently, $|E| = 2|V_i| + 6r + 4$.

Similarly, to obtain THIN n-ogons we must minimize the number of internal vertices of the arrangement. For all n, there are grid n-ogons such that $|V_i| = 0$. Thus, for THIN n-ogons $|V| = 4r + 4$.

Finally, to conclude the proof, we have to deduce the expression of the upper and lower bound of the number of faces of $\Pi(P)$, that is of $|\Pi(P)|$. Using Euler's formula $|F| = 1 + |E| - |V|$, and the expressions deduced above, we have $\max_P |F| = 1 + 2(\max_P |V_i|) + 6r + 4 - \max_P |V|$. That is, $\max_P |F| = \max_P |V_i| + 6r + 5$, so that

$$\max_P |F| = \begin{cases} \dfrac{3r^2+6r+4}{4}, & \text{for } r \text{ even} \\ \dfrac{3(r+1)^2}{4}, & \text{for } r \text{ odd} \end{cases}$$

and

$$\begin{aligned}\min_P |F| &= 1 + 2(\min_P |V_i|) + 6r + 4 - \min_P |V| \\ &= 1 + 6r + 4 - 4r - 4 = 2r + 1\end{aligned}$$

The existence of FAT grid n-ogons and THIN grid n-ogons for all n (such that n is even and $n \geq 4$) follows from Lemma 8 and the construction indicated in Fig. 5, respectively. □

Figs. 4 and 5 show some THIN n-ogons.

Fig. 4. Some grid n-gons with $|V_i| = 0$.

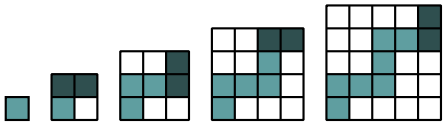
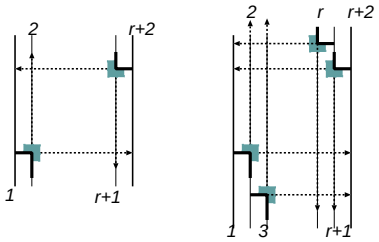


Fig. 5. Constructing the grid ogons of the smallest area, for $r = 0, 1, 2, 3, 4, \ldots$. The area is $2r + 1$.

Based on the proof of Proposition 9, we may prove the uniqueness of FATs and fully characterize them.

Proposition 12 *There is a single FAT n-ogon (except for symmetries of the grid) and its form is illustrated in Fig. 3.*

PROOF. We saw that FAT n-ogons must have a single reflex vertex in each vertical grid-line, for $x > 1$ and $x < \frac{n}{2}$. Also, the horizontal chords with origins at the reflex vertices that have $x = 2$ and $x = \frac{n}{2} - 1 = r + 1$, determine $2(r-1)$ internal points (by intersections with vertical chords). To achieve this value, they must be positioned as illustrated below on the left.



Moreover, the reflex vertices on the vertical grid-lines $x = 3$ and $x = r$ add $2(r - 2)$ internal points. To achieve that, we may conclude by some simple case reasoning, that v_{L_2} must be below v_{L_1} and v_{R_2} must be above v_{R_1}, as shown above on the right. And, so forth... □

The area $A(P)$ of a grid n-ogon P is the number of grid cells in its interior. FATs are not the grid n-ogons that have the largest area, except for small values of n, as we may see in Fig 6.

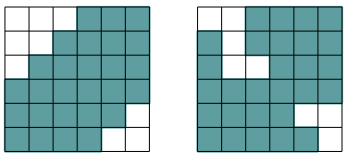


Fig. 6. On the left we see the FAT grid 14-ogon. It has area 27, whereas the grid 14-ogon on the right has area 28, which is the maximum.

Proposition 13 *Let P be any grid n-ogon with $n \geq 8$ and r reflex vertices ($r = \frac{n-4}{2}$, for all P). Then*

$$2r + 1 \leq A(P) \leq r^2 + 3$$

and there exist grid n-ogons having area $2r+1$ (indeed, a single one except for symmetries) and grid n-ogons having area $r^2 + 3$.

PROOF. Our proof is strongly based on the INFLATE-PASTE method for generating grid ogons, that will be presented also at this workshop [3]. □

References

[1] Bondy, J., Murty, U.: Graph Theory with Applications. Elseiver Science, New York, (1976).

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

[3] Tomás, A. P., Bajuelos, A. L.: Quadratic-time linear-space algorithms for generating orthogonal polygons with a given number of vertices. (Extended abstract) In Proceedings European Workshop on Computational Geometry (2004). (This book)

[4] Tomás, A. P., Bajuelos, A. L., Marques, F.: Approximation algorithms to minimum vertex cover problems on polygons and terrains. In P.M.A Sloot et al. (Eds): Proc. of ICCS 2003, LNCS 2657, Springer-Verlag (2003) 869-878.

Quadratic-Time Linear-Space Algorithms for Generating Orthogonal Polygons with a Given Number of Vertices ⋆

Ana Paula Tomás *,[a] and António Leslie Bajuelos [b]

[a] *DCC-FC & LIACC, University of Porto, Portugal*

[b] *Department of Mathematics & CEOC - Center for Research in Optimization and Control, University of Aveiro, Portugal*

Key words: Orthogonal Polygons, Generation, Decomposition, Dual Graphs, Mouths.

1. Introduction

This work focus on simple polygons without holes. Therefore, we call them just polygons. Moreover, "polygon" will sometimes mean a polygon together with its interior. P denotes a polygon and r the number of its reflex vertices. A polygon is *orthogonal* (or *rectilinear*) if its edges meet at right angles. O'Rourke [4] has shown that $n = 2r + 4$ for every n-vertex orthogonal polygon (n-*ogon*, for short). Generic n-ogons may be obtained from a particular kind of n-ogons, that we called grid orthogonal polygons, as illustrated in Fig. 1.

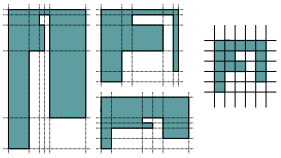


Fig. 1. Three 12-ogons mapped to the same grid 12-ogon.

Definition 1 *An n-ogon P is* in general position *iff every horizontal and vertical line contains at most one edge of P, i.e., iff P has no collinear edges. We call "grid n-ogon" each n-ogon in general position defined in a $\frac{n}{2} \times \frac{n}{2}$ square grid.*

We assume that the grid is defined by horizontal lines $y = 1, \ldots, y = \frac{n}{2}$ and vertical lines $x = 1, \ldots, x = \frac{n}{2}$ and that its northwest corner has coordinates (1,1). Each grid n-ogon has exactly one edge in every line of the grid.

Each n-ogon not in general position may be mapped to an n-ogon in general position by ϵ-perturbations, for a sufficiently small constant $\epsilon > 0$. Hence, we restrict generation to n-ogons in general position. Each n-ogon in general position is mapped to a unique grid n-ogon through top-to-bottom and left-to-right sweeping. And, reciprocally, given a grid n-ogon we may create an n-ogon that is an instance of its class by randomly spacing the grid lines in such a way that their relative order is kept.

1.1. *The paper's contribution*

We propose two methods that generate grid n-ogons in polynomial time – Inflate-Cut and Inflate-Paste. The former was published in [7], where we also gave implementation details, showing that it requires linear space in n and runs in quadratic time in average. Two programs for generating random orthogonal polygons, by O'Rourke (developed for the evaluation of [5]) and by Filgueiras [1] are mentioned there. The main idea of O'Rourke is to construct such a polygon via growth from a seed *cell* (i.e., unit square) in a board, gluing together a given number of cells that are selected randomly using some heuristics. Filgueiras' method shares a similar idea though it glues rectangles of larger areas and allows them to overlap. Neither of these methods allows to control the final number of vertices of the polygon. A major idea in Inflate-Paste is also to glue

⋆ (Extended abstract) Partially funded by LIACC through *Programa de Financiamento Plurianual, Fundação para a Ciência e Tecnologia (FCT)* and *Programa POSI*, and by CEOC (Univ. of Aveiro) through *Programa POCTI, FCT, co-financed by EC fund FEDER.*

* Corresponding author

Email addresses: apt@ncc.up.pt (Ana Paula Tomás), leslie@mat.ua.pt (António Leslie Bajuelos).

[1] personal communication, DCC-LIACC, 2003.

rectangles. Nevertheless, it restricts the positions where rectangles may be glued, which renders the algorithm simpler and provides control on the final number of vertices. For the latter purpose, the INFLATE transformation is crucial. It is possible to implement INFLATE-PASTE so that it requires quadratic-time in the worst-case and linear-space. Our methods may be also adapted to generate simple orthogonal polygons with holes. Indeed, each hole is an orthogonal polygon without holes.

2. Inflate, Cut and Paste transformations

Let $v_i = (x_i, y_i)$, for $i = 1, \ldots, n$, be the vertices of a grid n-ogon P, in CCW order.

INFLATE takes a grid n-ogon P and a pair of integers (p, q) with $p, q \in [0, \frac{n}{2}]$, and yields a new n-vertex orthogonal polygon $\tilde{P}$ with vertices $\tilde{v}_i = (\tilde{x}_i, \tilde{y}_i)$ given by $\tilde{x}_i = x_i$ if $x_i \leq p$ and $\tilde{x}_i = x_i + 1$ if $x_i > p$, and $\tilde{y}_i = y_i$ if $y_i \leq q$ and $\tilde{y}_i = y_i + 1$ if $y_i > q$, for $i = 1, \ldots, n$. Thus, it augments the grid, creating two free lines, $x = p + 1$ and $y = q + 1$.

INFLATE-CUT: Let C be a unit cell in the interior of P, with center c and northwest vertex (p, q). When we apply INFLATE to P using (p, q), c is mapped to $\tilde{c} = (p + 1, q + 1)$, that is the center of inflated C. The goal of CUT is to introduce $\tilde{c}$ as reflex vertex of the polygon. To do that, it cuts one rectangle (defined by $\tilde{c}$ and a vertex $\tilde{v}_m$ belonging to one of the four edges shot by the horizontal and vertical rays that emanate from $\tilde{c}$). We allow such a rectangle to be cut iff it contains no vertex of $\tilde{P}$ except $\tilde{v}_m$. If no rectangle may be cut, we say that CUT fails for C.

So, suppose that $\tilde{s}$ is the point where one of these rays first intersects the boundary of $\tilde{P}$, that $\tilde{v}_m$ is one of the two vertices on the edge of $\tilde{P}$ that contains $\tilde{s}$ and that the rectangle defined by $\tilde{c}$ and $\tilde{v}_m$ may be cut. CUT cuts this rectangle from $\tilde{P}$ replacing $\tilde{v}_m$ by $\tilde{s}$, $\tilde{c}$, $\tilde{s}'$ if this sequence is in CCW order (or $\tilde{s}'$, $\tilde{c}$, $\tilde{s}$, otherwise), with $\tilde{s}' = \tilde{c} + (\tilde{v}_m - \tilde{s})$. We may conclude that $\tilde{s}$, $\tilde{c}$, $\tilde{s}'$ is in CCW order iff $\tilde{s}$ belongs to the edge $\overline{\tilde{v}_{m-1}\tilde{v}_m}$ and in CW order iff it belongs to $\overline{\tilde{v}_m\tilde{v}_{m+1}}$. CUT always removes a single vertex of the grid ogon and introduces three new ones. Fig. 2 illustrates this technique. Because CUT never fails if C has an edge that is part of an edge of P, INFLATE-CUT may be always applied to P.

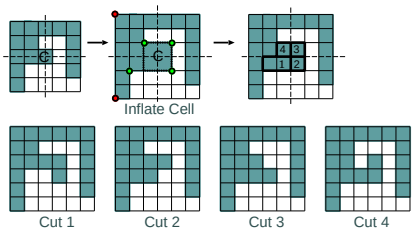


Fig. 2. The two rectangles defined by the center of C and the vertices of the leftmost vertical edge $((1, 1), (1, 7))$ cannot be cut. There remain the four possibilities shown.

INFLATE-PASTE: We first imagine the grid n-ogon merged in a $(\frac{n}{2} + 2) \times (\frac{n}{2} + 2)$ square grid, with the top, bottom, leftmost and rightmost grid lines free. The top line is $x = 0$ and the leftmost one $y = 0$, so that $(0, 0)$ is now the northwest corner of this extended grid. Let $e_H(v_i)$ represent the horizontal edge of P to which v_i belongs.

Definition 2 *Given a grid n-ogon P merged into a $(\frac{n}{2} + 2) \times (\frac{n}{2} + 2)$ square grid, and a convex vertex v_i of P, the* free staircase neighbourhood of v_i, *denoted by FSN(v_i), is the largest staircase polygon in this grid that has v_i as vertex, does not intersect the interior of P and its base edge contains $e_H(v_i)$.*

An example is given in Fig. 3.

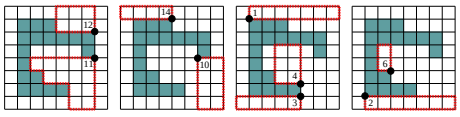


Fig. 3. A grid n-ogon merged into a $(\frac{n}{2} + 2) \times (\frac{n}{2} + 2)$ square grid and the free staircase neighbourhood for each of its convex vertices, with $n = 14$.

Now, to transform P by INFLATE-PASTE we first take a convex vertex v_i of P, select a cell C in FSN(v_i), and apply INFLATE to P using the nortwest corner (p, q) of C. As before, the center of cell C is mapped to $\tilde{c} = (p + 1, q + 1)$, which will now be a convex vertex of the new polygon. PASTE glues the rectangle defined by $\tilde{v}_i$ and $\tilde{c}$ to $\tilde{P}$, increasing the number of vertices by two. If $e_H(v_i) \equiv \overline{v_i v_{i+1}}$ then PASTE removes $\tilde{v}_i = (\tilde{x}_i, \tilde{y}_i)$ and inserts the chain $(\tilde{x}_i, q + 1)$, $\tilde{c}$, $(p + 1, \tilde{y}_i)$ in its place. If $e_H(v_i) \equiv \overline{v_{i-1} v_i}$, PASTE replaces $\tilde{v}_i$ by the sequence $(p + 1, \tilde{y}_i)$, $\tilde{c}$, $(\tilde{x}_i, q + 1)$. Fig. 4 illustrates this transformation. Clearly, PASTE never fails, in contrast to CUT.

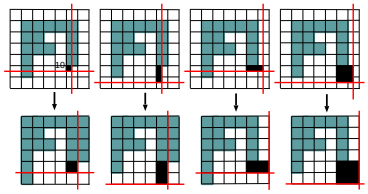


Fig. 4. The four grid 14-ogons that we may construct if we apply INFLATE-PASTE to the given 12-ogon, to extend the vertical edge that ends in vertex 10.

3. Inflate-Cut and Inflate-Paste Methods

We showed in [7] that every grid n-ogon may be created from a unit square (i.e., the grid 4-ogon) by applying r INFLATE-CUT transformations. Now, we may show the same result for INFLATE-PASTE. At iteration k, both methods construct a grid $(2k+4)$-ogon from the grid $(2(k-1)+4)$-ogon obtained in the previous iteration, for $1 \leq k \leq r$. The INFLATE-CUT method yields a random grid n-ogon, if cells and rectangles are chosen at random. This is also true for INFLATE-PASTE, though now for the selections of v_i and of C in FSN(v_i). It is not difficult to see that both INFLATE-CUT and INFLATE-PASTE yield grid ogons. In contrast, the proof of their completeness is not immediate, as suggested by the example given in Fig. 5.

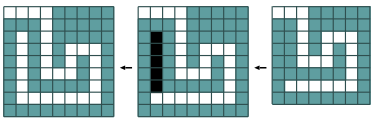


Fig. 5. The rightmost polygon is the unique grid 16-ogon that gives rise to this 18-ogon, if we apply INFLATE-CUT.

Before we go through the proof, we need to introduce some definitions and results.

Definition 3 *Given a simple orthogonal polygon P without holes, let $\Pi_{\mathrm{H}}(P)$ be the horizontal decomposition of P into rectangles obtained by extending the horizontal edges incident to reflex vertices towards the interior of P until they hit its boundary.*

Each *chord* (i.e., edge extension) separates exactly two adjacent pieces (faces), since it makes an *horizontal cut* (see e.g. [9]). The dual graph of $\Pi_{\mathrm{H}}(P)$ captures the adjacency relation between pieces of $\Pi_{\mathrm{H}}(P)$. Its nodes are the pieces of $\Pi_{\mathrm{H}}(P)$ and its non-oriented edges connect adjacent pieces.

Lemma 4 *The dual graph of $\Pi_{\mathrm{H}}(P)$ is a tree for all simple orthogonal polygons P without holes.*

PROOF. This result follows from the well-known Jordan Curve Theorem. Suppose the graph contains a simple cycle $F_0, F_1, \ldots, F_d, F_0$, with $d \geq 2$. Let $\gamma = (\gamma_{0,1}\gamma_{1,2}\ldots\gamma_{d,0})$ be a simple closed curve in the interior of P that links the centroids of the faces $F_0, F_1, \ldots, F_d$. Denote by v the reflex vertex that defines the chord $\overline{v\,s_v}$ that separates F_0 from F_1. Here, s_v is the point where this edge's extension intersects the boundary of P. Either v or s_v would be in the interior of γ, because γ needs to cross the horizontal line supporting $\overline{v\,s_v}$ at least twice and just $\gamma_{0,1}$ crosses $\overline{v\,s_v}$. But the interior of γ is contained in the interior of P, and there exist points in the exterior of P in the neighbourhood of v and of s_v, so that we achieve a contradiction. □

It is worth noting that the vertical decomposition $\Pi_{\mathrm{V}}(P)$ of P would have identical properties. We shall now prove Proposition 5 that asserts the completeness of INFLATE-PASTE.

Proposition 5 *For each grid $(n+2)$-ogon, with $n \geq 4$, there is a grid n-ogon that yields it by* INFLATE-PASTE.

PROOF. Given a grid $(n+2)$-ogon P, we use Lemma 4 to conclude that the dual graph of $\Pi_{\mathrm{H}}(P)$ is a tree. Each leaf of this tree corresponds to a rectangle that could have been glued by PASTE to yield P. Indeed, suppose that $\overline{v\,s_v}$ is the chord that separates a leaf F from the rest of P. Because grid ogons are in general position, s_v is not a vertex of P. It belongs to the relative interior of an edge of P. The vertex of rectangle F that is not adjacent to s_v would be $\tilde{c}$ in INFLATE-PASTE. If we cut F, we would obtain an inflated n-ogon, that we may deflate to get a grid n-ogon that yields P. The two grid lines $y = y_{\tilde{c}}$ and $x = x_{\tilde{c}}$ are free. Clearly s_v is the vertex we called v_i in the description of INFLATE-PASTE (more accurately, s_v is $\tilde{v}_i$) and $c = (x_{\tilde{c}} - 1, y_{\tilde{c}} - 1) \in FSN(v_i)$. □

For this paper to be self-contained, we recall now a proof of the completeness of INFLATE-CUT, already sketched in [7]. It was inspired by work about convexification of simple polygons [2,6,8], in particular, by a recent paper by O. Aichholzer et al. [1]. It also shares ideas of a proof of Meisters' *Two-Ears Theorem* [3] by O'Rourke, though we were not aware of this when we wrote it. Fig. 6 illustrates the fundamental ideas.

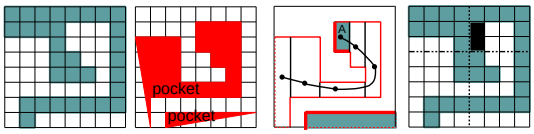


Fig. 6. The two leftmost grids show a grid 18-ogon and its pockets. The shaded rectangle A is a leaf of the tree associated to the vertical partitioning of the largest pocket. The rightmost polygon is an inflated grid 16-ogon that yields the represented grid 18-ogon, if CUT removes rectangle A.

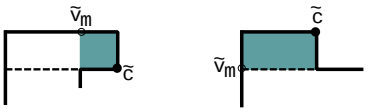
We need some additional definitions and results.

Definition 6 *A* pocket *of a nonconvex polygon P is a maximal sequence of edges of P disjoint from its convex hull except at the endpoints. The* lid *is the line segment joining its two endpoints.*

Any nonconvex polygon P has at least one pocket. Each pocket of an n-ogon, together with its lid, defines a simple polygon without holes, that is almost orthogonal except for an edge (lid). It is possible to slightly transform it to obtain an orthogonal polygon, as illustrated in Fig. 6. We shall refer to this polygon as an *orthogonalized pocket*. For every orthogonalized pocket Q, it is easy to see that the pocket's lid is contained in a single rectangle of either $\Pi_{\mathrm{H}}(Q)$ or $\Pi_{\mathrm{V}}(Q)$. Let $\Pi(Q)$ represent the one where the lid is contained in a single piece.

Proposition 7 *For each grid $(n+2)$-ogon, there is a grid n-ogon that yields it by* INFLATE-CUT.

PROOF. Given a grid $(n+2)$-ogon P, let Q be an orthogonalized pocket of P. Necessarily, Q is in general position. By Lemma 4 the dual graph of $\Pi(Q)$ is a tree. We claim that at least one of its leaves contains or is itself a rectangle that might have been removed by CUT to yield P. Indeed, the leaves are of the two following forms.



The shaded rectangles are the ones that might have been cut. We have also represented the points that would be $\tilde{v}_m$ and $\tilde{c}$ in INFLATE-CUT. Here, we must be careful about the leaf that has the pocket's lid. Only if the tree consists of a single node (c.f. the smallest pocket in Fig. 6), may this leaf be filled. But, every non-degenerated tree has at least two leaves. Then, in this case the tree has a leaf other than the one that contains the lid. □

The concept of *mouth* [8] was crucial to reach the current formulation of CUT. Actually, INFLATE-CUT is somehow doing the reverse of an algorithm given by Toussaint in [8] that computes the convex hull of a polygon globbing-up mouths to successively remove its concavities. For orthogonal polygons, we would rather define rectangular mouths.

Definition 8 *A reflex vertex v_i of an ogon P is a rectangular mouth of P iff the interior of the rectangle defined by v_{i-1} and v_{i+1} is contained in the exterior of P and neither this rectangle nor its interior contain vertices of P, except v_{i-1}, v_i and v_{i+1}.*

To justify the correction of our technique, we observe that when we apply CUT to obtain a grid $(n+2)$-ogon, the vertex $\tilde{c}$ is always a rectangular mouth of the resulting $(n+2)$-ogon. In sum, the proof of Proposition 7 given above justifies Corollary 9, which rephrases the *One-Mouth Theorem* by Toussaint.

Corollary 9 *Each grid n-ogon has at least one rectangular mouth, for $n \geq 6$.*

References

[1] Aichholzer, O., Cortés, C., Demaine, E. D., Dujmovic, V., Erickson, J., Meijer, H., Overmars, M., Palop, B., Ramaswawi, S., Toussaint, G. T.: Flipturning polygons. Discrete & Comput. Geometry **28** (2002), 231–253.

[2] Erdös, P.: Problem number 3763. American Mathematical Monthly **42** (1935) 627.

[3] Meisters, G. H.: Polygons have ears. American Mathematical Monthly **82** (1975) 648-651.

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

[5] O'Rourke, J., Pashchenko, I., Tewari, G.: Partitioning orthogonal polygons into fat rectangles. In Proc. 13th Canadian Conference on Computational Geometry (CCCG'01) (2001) 133-136.

[6] Sz.-Nagy, B.: Solution of problem 3763. American Mathematical Monthly **46** (1939) 176–177.

[7] Tomás, A. P., Bajuelos, A. L.: Generating Random Orthogonal Polygons. To appear in *Post-conference Proceedings of CAEPIA-TTIA'2003*, LNAI, Springer-Verlag (2004).

[8] Toussaint, G. T.: Polygons are anthropomorphic. American Mathematical Monthly **122** (1991) 31–35.

[9] Urrutia, J.: Art gallery and illumination problems. In J.-R. Sack and J. Urrutia, editors, Handbook on Computational Geometry. Elsevier (2000).

Warping Cubes: Better Triangles from Marching Cubes

LeeAnn Tzeng [a,1]

[a] *Dartmouth College, 6211 Sudikoff Laboratory, Hanover, NH 03755*

Key words: triangulation, isosurface reconstruction, Marching Cubes

1. Introduction

The Marching Cubes algorithm [6] for extracting a triangulation of an isosurface of a function defined over a three-dimensional space is famous throughout graphics. However, the triangles created by Marching Cubes are often quite skinny. These thin triangles can create artifacts in computer rendering, and make the surface triangulation unsuitable for volume rendering. To avoid these artifacts in rendering, thin triangles need to be eliminated from the triangulation created by the Marching Cubes algorithm.

The most basic form of Marching Cubes places a grid over the region containing the function whose isosurface we wish to extract. The function is evaluated at the vertices of this grid (I will call these the "corners" of the cubes to avoid confusion with the vertices of the resulting triangulation), and each corner is labelled as being positive or negative. Based on the pattern of labels, each cube is triangulated via a lookup table (for speed). The result is a triangulation that has vertices on the edges of the cubes where the isosurface intersects the edge. [8]

2. Warping and Collapsing

My algorithm modifies Marching Cubes by adding a "warping" phase to the original algorithm. This phase might better be called the "collapsing" phase, as will become clear shortly (though it could be described as "warping" the triangulation). The warping here should not be confused with Oct-tree Warping, which is visually similar but actually very different [7]. This process of warping or collapsing is based on the observation that skinny triangles take one of two forms. The first kind of skinny triangle is very tall, with a tiny base. This kind has one sharp angle at the top. The other kind of skinny triangle has an extremely short altitude and a very long base. (Fig. 1) My approach attacks each of these problems directly.

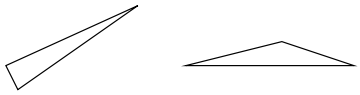


Fig. 1. Two kinds of skinny triangles.

2.1. *Short edges*

A short edge occurs when the isosurface intersects a cube very close to the cube's corner. Each endpoint of the short edge lies on one of the cube's edges close to that corner. A naive first thought is to pull the corner away from the short edge, creating a distorted cube that lengthens the short edge. However, even in 2D, it is easy to see how this might cause additional problems in adjacent cubes. (Fig. 2)

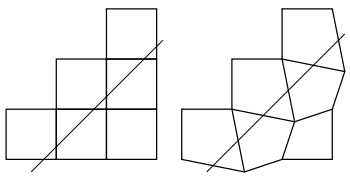


Fig. 2. In 2D, pulling the corners can create more short edges in neighboring cubes.

Instead, I choose to move the corner *onto* the short edge. This removes the short edge altogether

Email address: ltzeng@cs.dartmouth.edu (LeeAnn Tzeng).

[1] Supported on a National Science Foundation Graduate Fellowship.

by replacing its two endpoints, each a vertex in the triangulation, with a single vertex. This *warps* the cubes in a more unilaterally beneficial way: I eliminate a short edge but do not shorten any other edges. In fact, more often than not, the remaining edges are actually lengthened. One can easily see how this technique might also be seen as *collapsing* the short edge into a single vertex. (Fig. 3 and 4)

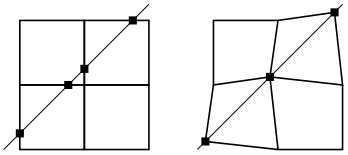


Fig. 3. Warping onto the short edge in 2D. This can also be seen as collapsing the short edge into a vertex.

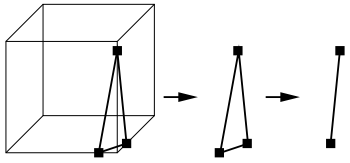


Fig. 4. Collapsing the short edge in 3D. The two long edges merge together into one edge. The top endpoint stays the same, and the bottom endpoint is the new collapsed vertex.

A quick mental experiment shows that the best place to which to move the corner is the midpoint of the short edge. Ideally, the warped triangulation stays as close to the known isosurface as possible. The midpoint of the short edge avoids ever getting too far away from the known isosurface, since it minimizes how far each endpoint has to move to achieve a collapsed edge. (Fig. 5)

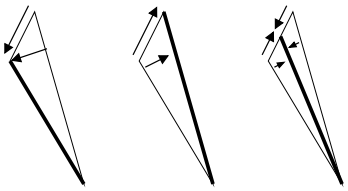


Fig. 5. Collapsing to the short edge's midpoint minimizes the distance from the known points of the isosurface.

One obvious question is what to do if there are two or more short edges at the same corner. If there are two short edges at the same corner, but not a third, then the two short edges must be in neighboring cubes. I choose to collapse the two edges into their shared vertex. This is better than collapsing first one and then the other short edge both because it keeps the resulting vertex on the known isosurface, and because it minimizes how far each of the neighboring non-short edges has to move. (Fig. 6)

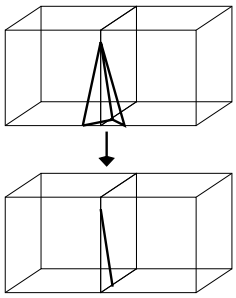


Fig. 6. Collapsing two adjacent short edges into their common vertex.

If there are three short edges at the same corner, then we have a small triangle that is likely to be well-shaped (by Delaunay standards). In this case, all three edges are collapsed into a single vertex. This looks like shrinking the small triangle into a vertex. Since the triangle is so small, any point within the triangle is a reasonable candidate for the final vertex, so I am currently using the incenter simply because it is guaranteed to be inside the triangle. (Fig. 7)

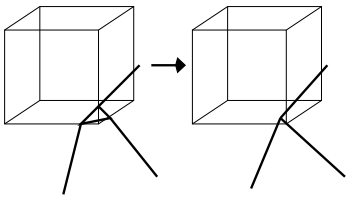


Fig. 7. Collapsing a small triangle into its incenter.

This concept is extended as edges are added. There can be a total of twelve short edges at one corner, which implies a "bubble" in the triangulation. If this bubble is completely disconnected from the rest of the triangulation, then all of its triangles will be nicely shaped, so there will not be any rendering artifacts. If the bubble is internal, then for the purposes of rendering, we can ignore the bubble completely. If the bubble has other edges extending from its vertices, one can collapse all of its edges collectively into the incenter of the octahedron defined by the twelve edges. This leaves a vertex with any outside edges now coming into it.

For any given number of short edges at one corner, I always choose to collapse the centermost

piece at that corner first. Four edges forces the fifth, which is two adjacent triangles sharing a common edge. I collapse the common edge first, leaving two short edges meeting at the new collapes vertex. I then treat it as a two-edge adjacent pair. A sixth edge forces eight edges, and the centermost piece is a vertex at the "peak" of the resulting pyramid. Since the centermost piece is already a vertex, I collapse all of the edges directly into the peak vertex. A ninth edge forces a total of twelve edges which has already been mentioned above.

2.2. *Short altitudes*

The other kind of thin triangle is in some ways much more insidious. The wide-base, short-altitude triangle does not necessarily have an obvious short edge to collapse, and the collapsing procedure has more potentially dangerous consequences. In isolation, it seems rather innocuous. The triangle can simply be collapsed along its short altitude, resulting in an edge. (Fig. 8) This altitude is the shortest distance that this triangle can be flattened, thus again minimizing how far the collapsed triangulation strays from the known isosurface.

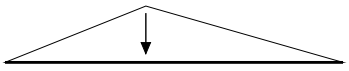


Fig. 8. The triangle is flattened onto its longest edge.

Within a triangulation, however, this collapse creates an extra vertex and requires the addition of a new edge to the triangulation. (Fig. 9) Whereas the short-edge collapse avoids ever shortening a remaining edge, the short-altitude collapse cuts a longer edge in two. This new edge has the potential to create a new thin triangle where there was not one before.

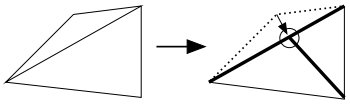


Fig. 9. The collapsed altitude results in a new vertex along the long edge and a new edge on the opposite side.

It is worth noting that there is only one way to get a short-altitude triangle from the original Marching Cubes. In particular, every triangle generated by Marching Cubes is contained within a cube, so the possible triangle configurations come from the Marching Cubes list. To get a short-altitude triangle, the two shorter edges must lie "across an edge of the cube", meaning that each edge lies in a face of the cube, and the two faces have an edge in common. (See Fig. 10) The vertex of the triangle that is an endpoint of the short altitude lies on the edge that is shared by the two faces of the cube. The two shorter edges of the triangle each extend to an adjacent edge of the cube respectively, and the distance along each of the cube edges where the triangle's edge ends is very small.

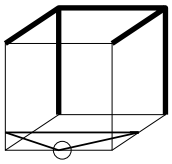


Fig. 10. A short-altitude triangle and the five edges where the opposite vertx may lie.

This means that the triangle on the other side of the short-altitude triangle's long edge can only take on a specific range of shapes. The opposite vertex can be in one of two relative locations. Consider the two faces across which the two shorter edges of the triangle lie. All seven of the edges bounding those two faces are not possible locations for the opposite vertex. This leaves five edges on which the opposite vertex may lie. Of these five, four give very similar triangle possibilities, and then there is the fifth. The fifth edge is the edge that shares both endpoints with other edges in this collection. All five of these possible edges yield the same worst-case triangle. This triangle *can* yield one short-altitude triangle, but that triangle can then be resolved by collapsing that triangle's shortest edge in the manner described in the previous section. This works because the only way to get another bad short-altitude triangle is to have a very short edge resulting from the first altitude-collapse. Further, the collapse of short edges can not generate a short-altitude triangle, so this ends the process.

There is one situation from the Marching Cubes 33 set of possible triangulations that allows for the short-altitude triangle to lie completely within one face of the cube [3]. In the cases where this occurs, there is always a fourth vertex on the remaining edge of that cube face, and this fourth vertex completes the triangle on the other side of the short-altitude triangle's longest edge. This opposite-side

triangle is, at worst, better than the worst-case triangle from the regular Marching Cubes triangulation. This triangle can thus be handled the same way as described above.

2.3. *Short edges before short altitudes*

It is important to collapse the short edges first, before collapsing the altitudes. Once the altitudes have been collapsed, another (much smaller) pass of short-edge-collapsing removes any new short edges left behind by the altitude collapses. The reason for this order is to avoid creating an unnecessary new short edge when the short altitude is collapsed onto the longest edge. The longest edge is broken into two, and a short edge in the starting triangle will cause the long edge to be broken into one very short edge and one reasonably long edge. This new short edge is unwanted. If the short edges are not collapsed first, this kind of short-edge creation has the potential to turn into a series of such short edges appearing. By collapsing the short edges first, the probability of these series arising during the altitude collapse is greatly reduced, and the few newly created short edges can be handled quickly afterwards.

3. Results and Conclusions

Without loss of generality, I treat the grid size as 1, and define ε to be the minimum acceptable edge length as a fraction of the grid size. I define η to be the minimum acceptable altitude, also as a fraction of the grid size. On the first pass, edges that are shorter than ε are collapsed. Once the short edges have been addressed, I find all triangles with altitude smaller than η and collapse those. Another quick pass through gets rid of any newly created short edges. Let B be the circumradius-to-shortest-edge ratio for a triangle. Then if I choose $\varepsilon = 0.4$ and $\eta = 0.35$, then $B \leq 1.5$.

The fourteen cube configurations defined by the Marching Cubes algorithm can be reduced to six based on combinations, and each of these has a worst-case triangle. For about half of these cases, it is sufficient to have $\eta = 0.25$, but there are a few cases that have particularly unwieldy potential triangles, and these require that $\eta = 0.35$. Since the shortest edge length is set by ε, the resulting bound on B directly implies a nice upper bound on the circumradius of the triangles.

Quality bounds on triangulations in isosurface extraction are still relatively rare. While a great deal of quality analysis has been done on mesh-generation and triangulation algorithms for known surfaces and point-sets, there has been remarkably little analysis on the quality of the triangulations generated by isosurface-extraction algorithms, where the surface is not known *a priori*. Attali and Lachaud give an algorithm that locally constructs an isosurface that is Delaunay conforming [2] [1], thus giving some sense of triangulation quality, but even Delaunay triangulations often still contain the skinny triangles that can cause havoc in rendering contexts. I have presented an isosurface extraction algorithm that respects a guarantee of quality in terms of circumradius-to-shortest-edge ratio, an increasingly popular measure of mesh and triangulation quality.

References

[1] D. Attali, J.-O. Lachaud, Constructing Iso-Surfaces Satisfying the Delaunay Constraint. Application to the Skeleton Computation., *Proc. 10th Intl. Conf. on Image Analysis and Processing* (Venice, Italy, 1999) 382–387.

[2] D. Attali, J.-O. Lachaud, Delaunay Conforming Iso-surface, Skeleton Extraction and Noise Removal, *Comp. Geometry: Theory and Applications* **19** (2001) 175–189.

[3] E.V. Chernyaev, Marching Cubes 33: Construction of Topologically Correct Isosurfaces, *CERN Report* **CN-95-17** (1995).

[4] P. Crossno, E. Angel, Isosurface Extraction Using Particle Systems, *IEEE Visualization* (1997) 495–498.

[5] K. Hormann, U. Labsik, M. Meister, G. Greiner, Hierarchical Extraction of Iso-Surfaces with Semi-Regular Meshes, *Proc. 7th ACM Symposium on Solid Modeling and Applications* (Saarbrücken, Germany, 2002) 53–58.

[6] W. E. Lorensen, H. E. Cline, Marching Cubes: A High Resolution 3D Surface Construction Algorithm, *Computer Graphics* **21:4** (July 1987) 163–169.

[7] S. A. Mitchell, S. A. Vavasis, Quality Mesh Generation in Higher Dimensions, *SIAM Journal on Computing* **29:4** (2000) 1334–1370.

[8] J. Sharman, The Marching Cubes Algorithm, *http://www.exaflop.org/docs/marchcubes/ind.html.* (2003).

[9] G. Varadhan, S. Krishnan, Y. J. Kim, D. Manocha, Feature-Sensitive Subdivision and Iso-Surface Reconstruction, to appear in *Proc. IEEE Visualization* (2003).

The siphon problem

J.M. Díaz-Báñez *[,a,1], C. Seara [b,2] and I. Ventura [a,1]

[a] *Universidad de Sevilla, Spain*
[b] *Universitat Politècnica de Catalunya, Spain*
[c] *Universidad de Huelva, Spain*

Abstract

An α-siphon is the locus of points in the plane that are at the same distance ϵ from a polygonal chain consisting of two half-lines emanating from a common point such that α is the interior angle of the half-lines. Given a set S of n points in the plane and a fixed angle α, we want to compute an α-siphon of largest width ϵ such that no points of S lies in its interior. We present an efficient $O(n^2)$-time algorithm for computing an orthogonal siphon. The approach can be handled to solve the problem of the oriented α-siphon for which the orientation of a half-line is known. We also propose an $O(n^3 \log n)$-time algorithm for the arbitrarily oriented version.

1. Introduction

A corridor through a planar point set S is the open region of the plane that is bounded by two parallel lines intersecting the convex hull of S, $CH(S)$. A corridor is empty if it does not contain any point of S. The problem of computing the widest empty corridor through a set S of n points in the plane has been solved by Houle and Maciel [3] in $O(n^2)$ time (Figure 1a).

One of the possible motivations of the widest empty corridor problem is to find a collision-free route to transport objects through a set of point obstacles. However, even the widest empty corridor may not be wide enough sometimes. This motivates to consider allowing right-angle turns. Chen in [1] studied this generalization considering an *L-shaped corridor*, which is the concatenation of two perpendicular links (a link is composed by two parallel rays and one line segment forming an unbounded trapezoid).

In this paper we consider a kind of corridor problem which we call the *siphon problem*. More precisely, we define a *siphon* as the locus of points in the plane that are at the same distance ϵ from a polygonal chain P consisting of two half-lines emanating from a common point (a 1-corner polygonal chain). Let α be the interior angle of the half-lines. An α-siphon is defined similarly but with the constraint that the interior angle of the two half-lines emanating from a common point is α.

An α-siphon is determined by P and ϵ, where ϵ is called the *siphon width*. Possible values for the *siphon angle* α are $0° \leq \alpha \leq 180°$. The α-siphon problem can be stated as follows.

α-Siphon problem. *Given a set S of n points in the plane and a fixed angle α, compute the α-siphon of largest width such that no points $p \in S$ lies in its interior (Figure 1b).*

The α-siphon has to intersect the convex hull of S producing a non-trivial partition S_1, S_2 of S; otherwise we will allow the α-siphon "to scratch the exterior" of S without actually passing through S and, therefore, the α-siphon can be arbitrarily wide (Figure 1c).

Notice that a 180°-siphon is just a corridor [3]. A 0°-siphon is a *silo* emanating from a point (the endpoint of a half-line) (Figure 1d) and it has been partially studied in [2] by giving an optimal $\Theta(n \log n)$-time algorithm in the case that the endpoint of the half-line is anchored on a given point.

* Corresponding author
Email addresses: dbanez@us.es (J.M. Díaz-Báñez), carlos.seara@upc.es (C. Seara), inmaculada.ventura@dmat.uhu.es (I. Ventura).

[1] The first author is partially supported by Project MCYT BFM2000-1052-C02-01.
[2] The second author is partially supported by projects MCYT-FEDER TIC-2001-2171, MCYT-FEDER BFM 2002-0557, Gen Cat 2001SGR00224.

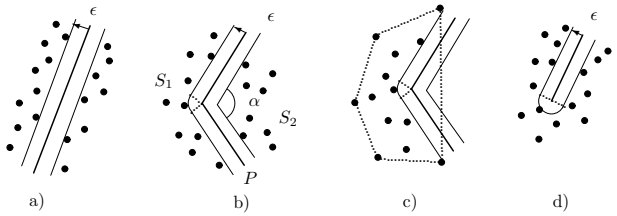


Fig. 1. a) Widest corridor, b) α-siphon c) unbounded width siphon, c) silo

Notice also that our α-siphon is a kind of corridor which is a "better" solution than Cheng's corridor in the following sense: suppose that we are interesting in transporting a circular object in between a set of points, Cheng's algorithm can gives a negative answer while our algorithm produces an affirmative answer; this is so because the width of the widest α-siphon is always larger than or equal to the width of the widest L-shaped corridor; in fact a siphon is the area swept by a disk whose center describes the route.

In this paper two variants for the α-siphon problem are consider: i) the *oriented α-siphon problem*, where we know the angle α and the direction of one of the half-lines of P; ii) the *arbitrarily oriented α-siphon problem*, where only the angle α is known. Most proofs are omitted in this extended abstract.

Let $S = \{p_1, p_2, \ldots, p_n\}$ be a planar point set. The points are in general position, this assumption is not essential for our algorithms to work, but handling degeneracies would require the description of many details and would hide the crucial ideas. We denote the Euclidean distance between two points p and q by $d(p, q)$. If p is a point and P is a closed subset in the plane, the distance between p and P is defined as $d(p, P) = \min\{d(p, q) : q \in P\}$.

An α-siphon is bounded by an outer boundary and an inner boundary; the outer boundary is formed by a circular arc joined to two half-lines, the *exterior boundary legs*; and the inner boundary is formed by two half-lines, the *interior boundary legs*. By *orthogonal siphon* we denote an oriented 90°-siphon such that its boundary legs are vertical and horizontal.

2. Oriented α-siphon

In this section we study the problem of computing an oriented α-siphon. First we consider the orthogonal siphon and next we will consider the general case. There are four possibilities for P in an orthogonal siphon according to the north, south, east and west directions. We only consider the case S-E, other cases can be handle analogously.

First notice that for any 1-corner polygonal chain P which does not contain points from S and produces a non-trivial partition of S, always exists a siphon defined by P. We call a siphon *non-expansive* if its interior boundary contains two points of S (one per each leg or only one point if this point is on the vertex of that boundary) and its exterior boundary contains one point of S.

Lemma 1 *For any fixed vertical and horizontal lines crossing the convex hull of S producing a non trivial partition of S, there exists a non-expansive siphon.*

Next we describe the algorithm which solves the S-E orthogonal siphon problem in $O(n^2)$ time. By Lemma 1 we know that an orthogonal siphon is defined by at most three points. First, the algorithm sorts the points in S by decreasing y-coordinate and by increasing x-coordinate in $O(n \log n)$ time. Let $O = \{p_1, \ldots, p_n\}$ and $A = \{q_1, \ldots, q_n\}$ be the respective lists of the sorted points. From a vertical-horizontal grid we construct a S-E staircase E which is updated each time we insert a new point, and in this case either the number of the steps of E increase by one or it decrease because the new point *dominates* some points of the current E.

Let p_1 and q_1 be the points of S with maximum y-coordinate and minimum x-coordinate, respectively. These two points determine the starting point of the algorithm. The staircase E in the initial stage is formed by the horizontal and vertical half-lines of the grid passing through p_1 and q_1. The algorithm compute all the possible orthogonal siphons which horizontal boundary leg is supported by $p_i \in O$, for $i = 2, \ldots, n$. If a point $p_i = (x_{pi}, y_{pi})$ is on the horizontal-interior boundary leg of a siphon then, there only exist siphons such that its "entries" are in-between points having x-coordinates and y-coordinates smaller than or equal to x_{pi} and y_{pi}, respectively; because the rest of points either are *dominated* by either the current staircase or the point p_i. The dominance relation between points of S is establish as in [4,5]. In [5] the maxima problem for a set of points is considered. The maxima problem consists of finding all the maxima of S under dominance and they can be computed in $\theta(n \log n)$ time. We are interested

in the maxima problem with the following dominance relation: $p_j \prec p_i \iff x_{pj} \leq x_{pi}$ *and* $y_{pj} \geq y_{pi}$. The corresponding set of maxima forms a *four quadrant staircase*.

We construct the staircase E in an incremental way with a cost of $O(\log n)$ time per point insertion. In the worst case the number of different siphons to be checked can be $O(n^2)$. Assume that O and A have been computed and also for each point p_i we have a pointer to q_l such that $p_i = q_l$.

ORTOGONAL-SIPHON-ALGORITHM
Input: S, O, A,
Output: Widest orthogonal siphon,

(i) **Initial stage:** O, A, $E :=$ staircase formed by the horizontal and vertical half-lines defined by q_1 and p_1. Compute the Voronoi diagram for S, $VD(S)$, and store it in a data structure such that a query point can be answered in $O(\log n)$ time.

(ii) **For** $i = 2$ to n, **do** "Compute the widest orthogonal siphon supported by p_i and q_j, such that $x_{q_j} < x_{p_i}$, $y_{q_j} < y_{p_i}$",

Let ϵ_1 be the distance between $y = y_{pi}$ and the horizontal line containing the segment of the current E which is intersected by $x = x_{qj}$. Compute $\epsilon_2 = x_{qj} - x_{q(j-1)}$ and $\epsilon_0 = \min\{\epsilon_1, \epsilon_2\}$. The orthogonal siphon supported by p_i and q_j has width smaller than or equal to $\epsilon_0/2$. Compute the part E_{ij} of E in-between the x-coordinates $x_{qj} - \epsilon_0$ and x_{qj} and the y-coordinates y_{pi} and $y_{pi} + \epsilon_0$. Check that E_{ij} is empty and compute the orthogonal siphon of width $\epsilon_0/2$ supported by p_i and q_j. Otherwise, we analyze each point of E_{ij} determining (if it exists) the corresponding orthogonal siphon as follows:

The vertex of the 1-corner polygonal line of the siphon will be located in the bisector of the second quadrant passing through (x_{qj}, y_{pi}). This vertex is the center of the circle which contains the arc of the siphon. By using E and $VD(S)$, we consider each of the three possible locations for the point belonging to the exterior boundary.

(iii) **Updating stage:** Insert p_i in E and update E in time $O(\log n)$ per insertion and in total time $O(n \log n)$ for the deletion of points from E (a point is deleted only once). Delete p_i from O, delete $q_l = p_i$ from A, update the widest width and the current siphon.

Lemma 2 *Each vertex c_i of the staircase E is analyzed at most once.*

Analysis of the algorithm: The number of stages is $O(n^2)$. At the stage (i, j) we check all the vertices in E_{ij} in $O(\log n)$ time. By Lemma 2 a vertex in E is analyzed only once, then the total time cost in the analysis of points in E is $O(n \log n)$. Therefore the running time of the algorithm is $O(n \log n) + O(n^2) = O(n^2)$.

Theorem 3 *The widest orthogonal siphon can be computed in $O(n^2)$ time.*

The techniques above can be adapted for computing the widest oriented α-siphon, i.e., a α-siphon with a given angle α, $0 < \alpha < 180°$ and a fixed direction of one of its half-lines.

Corollary 4 *The widest oriented α-siphon can be computed in $O(n^2)$ time.*

Theorem 5 *The problem of computing the widest oriented α-siphon has an $\Omega(n \log n)$ time lower bound in the algebraic decision tree model.*

Notice that the complexity of the algorithm above match the complexity of the algorithm for computing the widest corridor of a set of points. A small variation of the algorithm above can be used to compute the widest L-shaped orthogonal corridor (as defined by Chen [1]) with the same $O(n^2)$ running time.

Corollary 6 *The widest L-shaped orthogonal corridor can be computed in $O(n^2)$ time.*

A similar algorithm can be used if we want to compute a corridor of the same kind but with an angle different from 90° and knowing the direction of one of the links.

3. The arbitrarily-oriented α-siphon

In this section we deal with the computation of a widest-empty arbitrarily-oriented α-siphon, i.e., we only fix the *siphon angle* α. Assume that α is $\frac{\pi}{2}$.

Lemma 7 *There always exists an optimal $\frac{\pi}{2}$-siphon such that the interior boundary contains two points of S (one per each leg) or only one point if this point is on the corner of that boundary.*

The points in S that determine a tentative placement of an optimal α-siphon are called the *critical points*. Therefore we can classify the cases for critical points according to their location on the parts of the siphon, as it is shown in Figure 2.

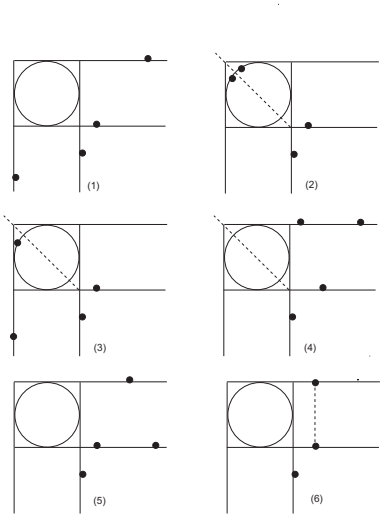


Fig. 2. Types of candidate siphons.

We sketch the idea of our approach without details. We only consider siphons that are bounded by a point of each its interior half-lines (according to Lemma 7). The orientation θ of our siphon is the smaller of the two angles given by the ortogonal lines supporting the interior boundary legs. Given two points p_i, p_j ($x_{p_i} > x_{p_j}$), we consider the rotation of the two perpendicular lines $r_i(\theta)$ (around p_i) and $s_j(\theta)$ (around p_j), say counterclockwise, to obtain all possible empty siphon supported at p_i and p_j. The essence of our algorithm is to generate a discrete set of subintervals in such rotation and compute a siphon of maximum width for each.

We begin the rotation in $\theta = 0$. A pair of orthogonal lines through p_i and p_j partitions the point set into four disjoint subsets which we label I, II, III and IV (corresponding to the four quadrants). As we change the orientation continuously, the partition survive till some two points become collinear. In fact, by insertion and deletion of the points, we can maintain dynamically each one of the corresponding partition. This spend $O(n^2)$ time and space. This produces a partition of the rotation interval. Taking into account the intervals of such partitions for each point $p_k \in S$, we define the following functions:

- $u_k(\theta) = d(p_k, r_i(\theta))$, for $p_k \in I$,
- $u_k(\theta) = d(p_k, r_i(\theta))$, $l_k(\theta) = d(p_k, s_j(\theta))$ and $c_k(\theta) = d(p_k, c_{ijk}(\theta))$, for $p_k \in II$,
- $l_k(\theta) = d(p_k, s_j(\theta))$, for $p_k \in III$,

where $c_{ijk}(\theta)$ is the center of the circle passing through p_k tangent to the lines $r_i(\theta)$ and $s_j(\theta)$.

Lemma 8 *Let p_k and p_l be two distinct points of S. Then, the graphs of two functions corresponding to p_k and p_l intersect at most twice.*

Let $\mathcal{L}$ be the lower envelope of the graphs of the functions u_k, l_k and c_k. Lemma 8 implies that the labels of the points corresponding to the edges of $\mathcal{L}$, when we traverse $\mathcal{L}$ from left to right, form a Davenport-Schinzel sequence of order two [6]. Therefore, the number of intervals in the partition is $O(n)$ [6], and by using a standard divide-and-conquer approach we can compute $\mathcal{L}$ in $O(n \log n)$ time. Thus, by traversing $\mathcal{L}$, from left to right, we can identify the highest vertex, which corresponds to the optimal direction for the $\frac{\pi}{2}$-siphon. In summary, working over all pair of points in S, we have proven the following result.

Theorem 9 *Given a set S of n points in the plane, the widest empty arbitrarily-oriented $\frac{\pi}{2}$-siphon can be computed in $O(n^3 \log n)$ time.*

An adaptation of above approach permits to solve the problem for a fixed angle α, $0° \leq \alpha \leq 180°$ in the same time bound.

A constrained version of this problem consists into anchoring the vertex of the 1-corner polygonal chain. In this case we obtain the following result.

Theorem 10 *Given a set S of n points in the plane, the widest-empty arbitrarily-oriented anchored α-siphon can be computed in optimal $\Omega(n \log n)$ time.*

References

[1] S-W. Chen, *Widest empty L-shaped corridor*, Information Processing Letters, 58, 1996, pp. 277–283.

[2] F. Follert, E. Schömer, J. Sellen, M. Smid, C. Thiel, *Computing the largest empty anchored cylinder, and related problems*, Inter. Journal of Comput. Geometry and Aplications, Vol. 7 (6), 1997, pp. 563–580.

[3] M. E. Houle, A. Maciel, *Finding the widest empty corridor through a set of points*, in Snapshots of Computational and Discrete Geometry, Godfried Toussaint, ed., Technical Report SOCS-88.11, School of Computer Science, McGill University, 1988.

[4] H. T. Kung, F. Luccio, F. P. Preparata, *On finding the maxima of a set of vectors*, Journal of ACM, 22(4), 1975, pp. 469–476.

[5] F. P. Preparata, M. I. Shamos, *Computacional Geometry, An introduction*, Springer-Verlag, 1988.

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

Defining discrete Morse functions on infinite surfaces

R. Ayala [a], L.M. Fernández [a], J.A. Vilches [a]
[a] *Dpto. de Geometría y Topología, Universidad de Sevilla, 41080, SPAIN*

Abstract

We present an algorithm which defines a discrete Morse function in Forman's sense on an infinite surface including a study of the minimality of this function.

Key words: discrete Morse function, infinite surface, critical point, Morse inequalities

1. Introduction

Under a classical or smooth point of view, Morse theory looks for links between global properties of a smooth manifold and critical points of a function defined on it. In [2], Forman introduced the notion of discrete Morse function defined on a finite *cw*-complex and, in this combinatorial context, he developed a discrete Morse theory as a tool for studying the homotopy type and homology groups of these complexes.

Once a Morse function has been defined on a complex, then its topological information can be deduced from the critical simplices of this function. Since we studied the problem concerning the definition of discrete Morse functions on infinite 1-complexes in other works [1,4], the goal of this paper is to continue with the following natural step: to develop a way of constructing discrete Morse functions on infinite 2-complexes, in particular on connected and non compact surfaces. Our methods are based on algorithms developed by T. Lewiner [3] for the finite case.

Given a simplicial complex M, R. Forman [2] introduces the notion of *discrete Morse function* as a function $f : M \longrightarrow R$ such that, for any p-simplex $\sigma \in M$:

(M1) $card\{\tau^{(p+1)} > \sigma / f(\tau) \leq f(\sigma)\} \leq 1$.

(M2) $card\{\upsilon^{(p-1)} < \sigma / f(\upsilon) \geq f(\sigma)\} \leq 1$.

A p-simplex $\sigma \in M$ is said to be *critical* with respect to f if:

(C1) $card\{\tau^{(p+1)} > \sigma / f(\tau) \leq f(\sigma)\} = 0$.

(C2) $card\{\upsilon^{(p-1)} < \sigma / f(\upsilon) \geq f(\sigma)\} = 0$.

2. Constructing discrete Morse functions

In order to define a discrete Morse function on an infinite surface S we recall that it can be expressed as a countable union of finite subcomplexes $S = \cup_{n \in N} K_n$, with $K_n \subseteq K_{n+1}$ for any $n \in N$. Indeed, let v_0 be any vertex of a triangulation of S and let K_1 be the closed star of v_0, that is, the smallest closed subcomplex of S which contains all edges and triangles including v_0. The following figure shows the closed star of v_0 in continous lines.

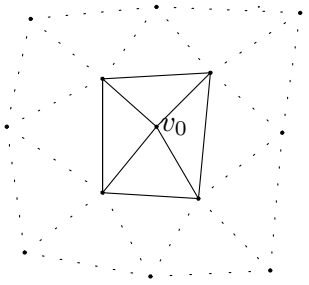


Fig. 1. Star of v_0.

Email addresses: lmfer@us.es (L.M. Fernández), vilches@us.es (J.A. Vilches).

[1] This work is partially supported by *P.A.I.*

To define the rest of subcomplexes K_n, we can do a successive thickening of K_1: K_2 will be the closed star of K_1 and, in the general case, K_n will be the closed star of K_{n-1}, where the closed star of a subcomplex means the smallest closed subcomplex that contains all edges and triangles which contain some simplex of the given subcomplex.

Next, we shall use a special graph which contains information of part of any K_n, in which all triangles and some edges of K are represented. If T_1 is a spanning tree in K_1, then the *complementary graph* of T_1 in K_1, denoted by D_1, is the graph constructed as follows: each vertex of D_1 corresponds to a triangle of K_1 or to a bounding edge of K_1 (that is, an edge which is in a unique triangle of K_1 and not in T_1) and there is an edge between two vertices of D_1 if these ones correspond to two triangles sharing an edge or to a triangle and a bounding edge which is in this triangle but not in T_1. Considering the preceding figure, D_1 is the graph drawed with discontinous lines in the following figure:

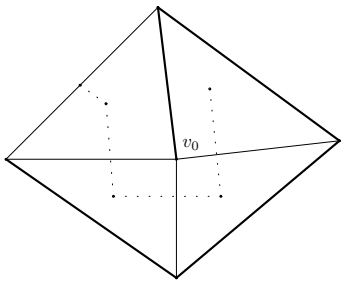


Fig. 2. D_1.

Now we enlarge the spanning tree T_1 until we get a spanning tree T_2 in K_2. Then, we obtain D_2 as an enlargement of D_1 and it is the complementary graph of T_2 in K_2. We can continue this process in successive steps. It is interesting to point out that this construction is only possible for 2-dimensional complexes because the union of T_n and D_n covers whole K_n, for any $n \in N$.

The definition of a discrete Morse function f on S starts with the definition of f in K_1. This process is divided in two steps: first, we define f on a spanning tree T_1 in K_1 starting at v_0. This assignment is made by an increasing way and such that we do not introduce any critical vertex or edge but the vertex v_0, which is a global minimum and hence it is a critical vertex [4].

On the other hand, in order to complete the definition of f on the whole K_1, we shall need to define f on D_1. To this end, we define the **degree of a vertex** corresponding to a triangle as the number of edges which are in this triangle such that we have not assigned them any value. Since S is a surface, the degree of any vertex is a number between 0 and 3. Now we start the definition of f on vertices of degree 1 assigning them the greatest value of f on T_1 plus one unit. This means that we have assigned that value to the triangles of K_1 corresponding to such vertices of degree one. Next, we assign the same value to the free (no values assigned) edges of such triangles. Then, we re-write the degrees of the vertices of D_1 because they could have changed and continue assigning values to vertices of degree one by increasing in one unit until finishing with all vertices of D_1. See the following figure as an example.

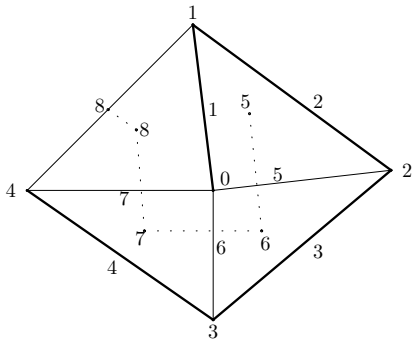


Fig. 3. f on K_1.

Now, to extend f to the subcomplex K_2, we repeat the increasing procedure to assign values of f on $T_2 - T_1$ and on $D_2 - D_1$. Since this procedure is well defined for any $n \in N$, it gives us a function f defined on whole surface S. It's important to point out that in this process may appear situations in which there are no degree one vertices in a D_n. It implies that the corresponding edges are in a cycle of non assigned edges.

3. Study of *f*

The following result states that the above defined function f is a discrete Morse function in

Forman's sense and give us information about its critical elements.

Proposition. *The function f given by the preceding procedure is a discrete Morse function defined on the infinite surface S and has a unique critical vertex, the initial vertex v_0, as many critical edges as many independent cycles of non assigned edges are found and do not have any critical triangle.*

References

[1] R. Ayala, L.M. Fernández and J.A. Vilches, Desigualdades de Morse generalizadas sobre grafos, *Actas de las III jornadas de Matemática Discreta y Algorítmica* (Universidad de Sevilla, Spain, 2002) 159–164.

[2] R. Forman, Morse Theory for cell complexes, *Adv. in Math.* **134** (1998) 90-145.

[3] T. Lewiner, G. Tavares and H. Lopes. Optimal Morse-Forman functions for combinatorial 2-manifolds, *to appear in Computational Geometry: Theory and Applications* (2003).

[4] J.A. Vilches, Funciones de Morse discretas sobre complejos infinitos, *book* (Edición Digital @tres, Sevilla, 2003).

On Geometric Properties of Enumerations of Axis-Parallel Rectangles [1]

Kira Vyatkina

Research Institute for Mathematics and Mechanics, St Petersburg State University, 28 Universitetsky pr., Stary Peterhof, 198504, St Petersburg, Russia

Abstract

We show that for any set of non-overlapping axis-parallel rectangles in the plane, there exists a *sloping* enumeration, such that the numbers of rectangles intersected by any line with a non-negative slope increase along this line. Such enumeration can be computed in the optimal time $\Theta(n \log n)$ using linear space. The notion of a sloping enumeration can be generalized to higher dimensions; however, already in three-dimensional space it may not exist. We also consider a strip packing problem for a set of rectangles with a fixed enumeration, which is required to be sloping for the resulting packing. This problem is proved to be NP-hard in any dimension $d \geq 2$.

Key words: rectangles, enumeration, orthogonal packing, NP-hardness

1. Introduction

A *Young diagram* is a collection of boxes, arranged in left-justified rows, with a (weakly) decrasing number of boxes in each row (Fig. 1a). A *standard Young tableau* is obtained by placing the numbers $1, 2, \ldots, n$ in the n boxes of the diagram in such way that the numbers increase across each row and down each colomn (Fig. 1b).

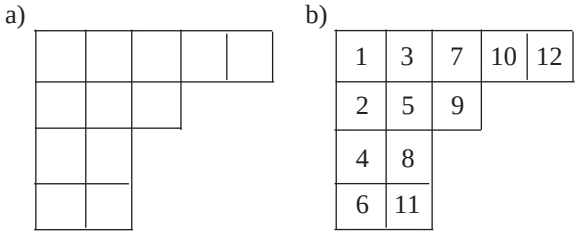


Fig. 1. a) A Young diagram; b) a standard Young tableau.

Young diagrams were introduced by Alfred Young in 1900 as a combinatorial tool (see [9]); at present combinatorics of Young tableaux has a wide range of applications in algebraic geometry and representation theory (see for example [6]).

Sometimes Young diagrams are written upside down (Fig. 2a).

Instead of considering a Young tableau as a combinatorial structure, let us look at its geometric representation: consider a standard Young tableau written upside down as a set of enumerated unit squares drawn in the plane, touching their boundaries. It is easy to see that for any line l with a non-negative slope, the numbers of squares intersected by l increase along l (Fig. 2a).

We generalize this observation in the following way: for any set of non-overlapping axis-parallel rectangles in the plane, there exists a *sloping* enumeration, such that for any line l with a non-negative slope, the numbers of rectangles intersected by l increase along l (Fig. 2b). Such enumeration can be efficiently computed in $\Theta(n \log n)$ time and $O(n)$ space. These results can be viewed as a new interpretation of our recent results [3].

All standard Young tableaux corresponding to a given Young diagram can be obtained as a topological ordering of vertices of a particular graph associated with the diagram (see [9]). All sloping enumerations for a set R of rectangles can be ob-

Email address: kira@meta.math.spbu.ru (Kira Vyatkina).

[1] This work is partially supported by Russian Foundation for Basic Research (grant 04-01-00173).

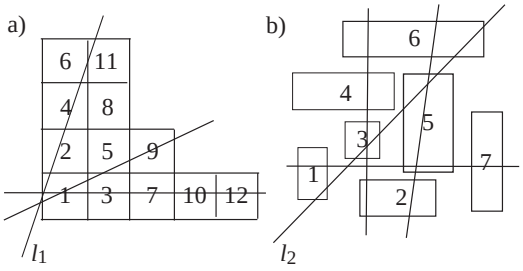


Fig. 2. a) A Young tableau written upside down; line l_1 consequently intersects squares 1, 2, 4, 6, 11. b) An enumerated set of rectangles; line l_2 consequently intersects rectangles 1, 3, 5, 6. For both a) and b), numbers of squares/rectangles intersected by any line with a non-negative slope increase along this line.

tained in the same way from a *placement graph* associated with R.

It is straightforward to generalize the notion of a sloping enumeration to higher dimensions; however, we provide an example showing that even in three-dimensional space, a sloping enumeration may not exist.

A *strip packing problem* is a special type of orthogonal packing problems (see the classification schemes introduced in [5,11]); much research has been recently carried out on problems of this kind (see the references in [5]). We consider the strip packing problem in the following form: given a set of enumerated rectangles and a horizontal strip of height H, we have to pack all rectangles into the strip, in such way that the given enumeration would be sloping for the resulting packing. As a matter of fact, we usually follow these restrictions when packing our luggage – since, for example, we do not want heavy objects to press down fragile ones. Asking for the minimal length of the strip, sufficient to store all rectangles, is shown to be NP-hard. This result can be generalized to an arbitrary dimension.

2. Preliminaries

Further we shall assume that all rectangles are axis-parallel, and no two of them overlap. We borrow terminology from [1,2,4].

For a rectangle p, let us denote its lower left, upper left, upper right and lower right vertices by A_p, B_p, C_p, and D_p, respectively. A *zone* $Z(p)$ of rectangle p is an open lower left quadrant built from point C_p (Fig. 3a). Given a set R of rectangles, its *placement graph* G_R has a vertex for each rectangle $p \in R$; for two vertices p and q, G_R contains an arc (p, q) iff $p \cap Z(q) \neq \emptyset$ (Fig. 3b).

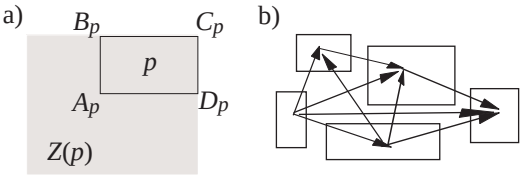


Fig. 3. a) Rectangle p and its zone $Z(p)$; b) a set of rectangles and the corresponding placement graph.

The following properties were observed in [2,10]:

Lemma 1 *Let $p, q \in R$. If $p \cap Z(q) \neq \emptyset$, then $q \cap Z(p) = \emptyset$.*

Theorem 2 *G_R is acyclic.*

3. Sloping Enumerations

Theorem 2 implies that the vertices of G_R can be topologically sorted; it follows (see [2,10,1]) that:

Theorem 3 *For any set R, there exists an enumeration $I_R : R \leftrightarrow \{1, 2, \ldots, N\}$, where $N = |R|$, such that $\forall p \in R$: $\cup\{Z(q) | I_R(q) < I_R(p)\} \cap p = \emptyset$.*

It is easy to prove that enumeration I_P satisfies our requirements.

Theorem 4 *I_R is sloping.*

PROOF. Consider a line l with a non-negative slope; let us assume that l intersects at least two rectangles. Denote the intersected rectangles by $p_{k_1}, \ldots, p_{k_m}$, according to the order, in which they are intersected by l (Fig. 4).

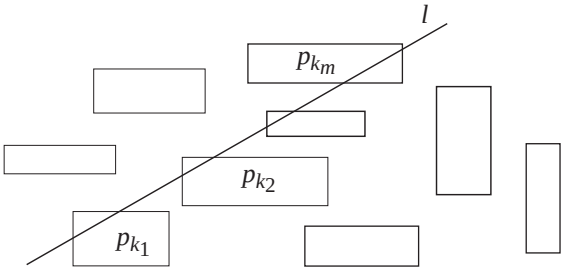


Fig. 4. Line l consequently intersects rectangles p_{k_1}, p_{k_2}, $\ldots$, p_{k_m}. For $1 \leq i < j \leq m$, we have $p_{k_i} \cap Z(p_{k_j}) \neq \emptyset$.

Obviously, $1 \leq i < j \leq m$ implies $p_{k_i} \cap Z(p_{k_j}) \neq \emptyset$, and thus, $I_R(p_{k_i}) < I_R(p_{k_j})$ must hold. It can be easily proved by induction that $I_R(p_{k_1}) < \cdots < I_R(p_{k_h})$ for any $1 \leq h \leq m$; this completes our proof.

A sloping enumeration need not be unique. To build *some* sloping enumeration, one may construct the placement graph G_R and sort topologically its vertices.

Graph G_R can be constructed with a plane sweep algorithm in $O(n \log n + |E_R|)$ time and $O(n)$ space, where $|E_R|$ is the number of its edges. This algorithm is optimal in the comparison tree model. The sorting can then be performed in $O(n + |E_R|)$ time; thus, computing a sloping enumeration in this way would take us $O(n \log n + |E_R|)$ time.

The method described above is not optimal; however, a sloping enumeration can be obtained in the optimal $\Theta(n \log n)$ time and in linear space using a plane sweep algorithm; the reader is referred to [3] for details.

4. Strip Packing Problem

Let us consider a two-dimensional strip packing problem with respect to a given enumeration (SPPE-2).

The optimization problem is:

SPPE-2: Given a set of enumerated rectangles and a horizontal strip of height H, we ask for the minimal length L of the strip, sufficient to pack all rectangles in such way that the given enumeration is sloping for the resulting packing.

The corresponding decision problem is:

SPPE-2*: For a given a set of enumerated rectangles and a horizontal strip of height H and length L, is there a feasible packing, for which the given enumeration is sloping?

Theorem 5 *SPPE-2 is NP-hard.*

PROOF. Clearly, SPPE-2*∈NP. Let us reduce to SPPE-2* the PARTITION problem, which is known to be NP-complete [8,7]. In PARTITION, we are given a finite set A and a size $s(A) \in Z^+$ for each $a \in A$, and we ask if there exists a subset $A' \subseteq A$, such that $\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)$.

For set $A = \{a_i\}_{i=1}^n$, let $H = 5$, and $L = 2\sum_{i=1}^n s(a_i)$. For each $a_i \in A$, we construct rectangle $r(a_i)$ of size $2s(a_i) \times 2$. In addition, we construct two rectangles r_0 and r_{n+1}, each of size $\sum_{i=1}^n s(a_i) \times 2$. We enumerate rectangles as follows: $I_R(r_0) = 1$, $I_R(r_{n+1}) = n + 2$, $I_R(r(a_i)) = i + 1$, for $1 \leq i \leq n$.

Obviously, the length of the strip needed to pack all rectangles is at least L. It is precisely L iff the answer for the PARTITION decision problem is "yes".

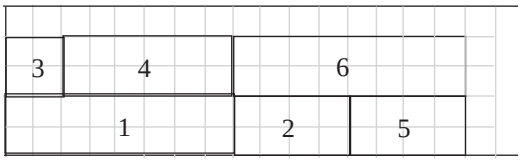


Fig. 5.

Fig. 5 illustrates the reduction described above for set $A = \{a_1, a_2, a_3, a_4\}$, with sizes $s(a_1) = 2$, $s(a_2) = 1$, $s(a_3) = 3$, $s(a_4) = 2$. Thus, $L = 16$; $H = 5$. Choosing $A' = \{a_1, a_4\}$ gives us a partition, and the packing shown on Fig. 5 fits into a strip of size $H \times L$. Clearly, our enumeration is sloping for this packing.

5. Higher-Dimensional Case

The notion of a sloping enumeration can be easily generalized to higher dimensions. In d-dimensional space, we shall consider d-dimensional *boxes* instead of rectangles.

Let us consider a set of vectors $V = \{\mathbf{v} | \mathbf{v} = \sum_{i=1}^d a_i \mathbf{e_i}, a_i \geq 0, 1 \leq i \leq d\}$, where $\mathbf{e_i}$ are unit vectors pointing along coordinate axes. Enumeration I_R of a set R of boxes is *sloping*, if for any line l, such that $\exists \mathbf{v} \in V : l || \mathbf{v}$, the numbers of boxes intersected by l increase along l.

For $d = 2$, this definition is equivalent to the one given before.

In the three-dimensional case, a sloping enumeration may not exist. A construction from [2], shown on Fig. 6, illustrates such situation. Notice that if we set in the definition $a_1 \leq 0$, $a_2, a_3 \geq 0$, there *will be* a sloping enumeration. However, in [3] we introduce a construction, for which no sloping enumeration exists for any choice of signs for a_1, a_2,

a_3. (In [3], it serves as an example of a set of parallelepipeds admitting no right-angled cut partition.)

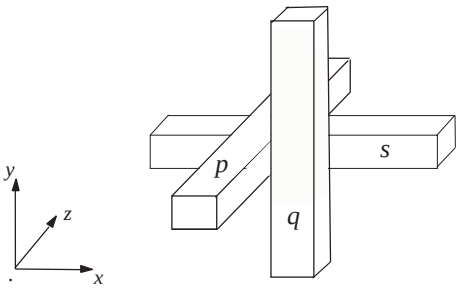


Fig. 6. For the set $\{p, q, s\}$, there exists no sloping enumeration. However, there will be a sloping enumeration, if we set $a_1 \le 0$, $a_2, a_3 \ge 0$: $I_R(q) = 1$, $I_R(s) = 2$, $I_R(p) = 3$.

Let us generalize to higher dimensions the strip packing problem.

SPPE-d: Given a set of enumerated boxes in d-dimensional space, we ask for the minimal length W_1 of a container, sufficient to pack all boxes in such way that the given enumeration is sloping for the resulting packing, where the sizes in the other $d-1$ dimensions $W_2, \ldots, W_d$ are fixed.

SPPE-d*: For a given a set of enumerated boxes in d-dimensional space and a container of size W^d, is there a feasible packing, for which the given enumeration is sloping?

Theorem 6 *SPPE-d is NP-hard.*

To prove this theorem, we apply reduction similar to the one described above, setting $W_2 = 5$, $W_3 = \cdots = W_d = 1$, and requiring box $r(a_i)$ to have size $2s(a_i) \times 2 \times 1 \cdots \times 1$, $1 \le i \le n$, and boxes r_0 and r_{n+1} – to have size $\sum_{i=1}^{n} s(a_i) \times 2 \times 1 \cdots \times 1$.

6. Conclusion

For an arbitrary set of non-overlapping axis-parallel rectangles in the plane, we have proved existence of a sloping enumeration, and proposed methods for computing it efficiently. We have also considered a strip packing problem for an enumerated set of rectangles; an additional requirement concerned with the enumeration is the following: the given enumeration must be sloping for the resulting packing. This problem is shown to be NP-hard.

We have generalized the notion of a sloping enumeration to the case of d-dimensional space. For the three-dimensional case, we gave an example of a set of parallelepipeds, which admits no sloping enumeration. However, the strip packing problem may be stated for any dimension d; we have proved it to be NP-hard in arbitrary dimension.

All valid enumerations arise as a topological ordering of vertices of a placement graph. This graph is known to be acyclic [2,10]; however, we suppose that much more could be derived on its structure. We are going to consider this question, along with the opposite one: what kind of dags can appear as placement graphs?

References

[1] V. Bukhvalova, and L. Faina, Rectangular Cutting Layout Problem, *http://loris.dipmat.unipg.it/usa/index.html.* (1998).

[2] V. Bukhvalova, *Rectangular Cutting Problem: Zone Method and Other Algorithms* (Saint Petersburg State Univ. Press, St Petersburg, 2001). (in Russian)

[3] V. Bukhvalova, and K. Vyatkina, An Optimal Algorithm for Partitioning a Set of Axis-Parallel Rectangles with Right-Angled Cuts, in: M. Neamtu and M. Lucian, eds., *Proceedings SIAM Conference on Geometric Design and Computing* (Seattle, USA, 2003). (to be submitted)

[4] L. Faina, An application of simulated annealing to the cutting stock problem, *European Journal on Operational Research*, **114(3)** (1999) 542–556.

[5] S. Fekete, and J. Schepers, A Combinatorial Characterization of Higher-Dimensional Orthogonal Packing, *http://arxiv.org/abs/cs.DS/0310032.* (2003).

[6] W. Fulton, *Young Tableaux with Applications to Representation Theory and Geometry* (New York: Cambridge University Press, 1997).

[7] M.R. Garey, and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness* (San Francisco: Freeman, 1979).

[8] R.M. Karp, Reductibility among combinatorial problems, in: R.E. Miller and J.W. Thatcher, eds., *Complexity of Computer Computations* (Plenum Press, New York, 1972) 85–103.

[9] D. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching, Second Edition* (Reading, Massachusetts: Addison-Wesley, 1998).

[10] A.I. Lipovetsky, Properties of Rectangular Packings, *manuscript* (1988). (in Russian)

[11] M. Wottawa, *Struktur und algorithmische Behandlung von praxisorientiren dreidimensionalen Packungsproblemen* (Ph.D. thesis, Universität zu Köln, 1996).

Maximum Weight Triangulation of A Special Convex Polygon [1]

Jianbo Qian [2] and Cao An Wang [3].

Abstract

In this paper, we investigate the maximum weight triangulation of a special convex polygon, called 'semi-circled convex polygon'. We prove that the maximum weight triangulation of such a polygon can be found in $O(n^2)$ time.

1 Introduction

Triangulation of a set of points is a fundamental structure in computational geometry. Among different triangulations, the *minimum weight triangulation* (MWT for short) of a set of points in the plane attracts special attention [1,2,3]. The construction of the MWT of a point set is still an outstanding open problem. When the given point set is the set of vertices of a convex polygon (so-called *convex point set*), then the corresponding MWT can be found in $O(n^3)$ time by dynamic programming [1,2].

On the contrast, there is not much research done on *maximum weight triangulation* (MAT for short). From the theoretical viewpoint, the maximum weight triangulation problem and the minimum weight triangulation problem attracts equally interest, and one seems not to be easier than the other. The study of maximum weight triangulation will help us to understand the nature of optimal triangulations.

The first work in the MAT [4] showed that if an n-sided polygon P inscribed on a circle, then $MAT(P)$ can be found in $O(n^2)$ time.

In this paper, we study the MAT of the following special convex polygon: let P be a convex $n-$sided polygon such that the entire polygon P is contained inside the circle with an edge of P as diameter. We call such a polygon as *semi-circled*. We propose an $O(n^2)$ algorithm for computing the $MAT(P)$ of a semi-circled convex polygon P. Recall that a straightforward dynamic programming method will take $O(n^3)$ to find $MAT(P)$.

[1]This work is supported by NSERC grant OPG0041629 and the National Natural Science Foundation of China under Grant No. 70221001.

[2]Institute of Applied Mathematics, Chinese Academy of Science, Beijing, China. (email: jbqian@mail.amss.ac.cn)

[3]Department of Computer Science, Memorial University of Newfoundland, St. John's, Newfoundland, Canada A1B 3X5 (email: wang@garfield.cs.mun.ca)

2 Preliminaries

Let S be a set of points in the plane. A *triangulation* of S, denoted by $T(S)$, is a maximal set of non-crossing line segments with their endpoints in S. It follows that the interior of the convex hull of S is partitioned into non-overlapping triangles. The weight of a triangulation $T(S)$ is given by

$$\omega(T(S)) = \sum_{\overline{s_i s_j} \epsilon T(S)} \omega(\overline{i,j}),$$

where $\omega(\overline{i,j})$ is the Euclidean length of line segment $\overline{s_i s_j}$.

A *maximum weight triangulation* of S ($MAT(S)$) is defined as for all possible $T(S)$, $\omega(MAT(S)) = max\{\omega(T(S))\}$.

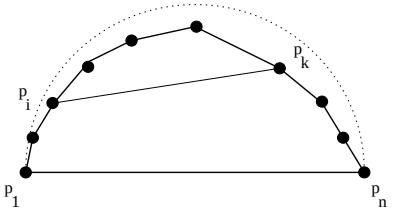


Figure 1: An illustration of semi-circled convex polygon.

Let P be a convex polygon (whose vertices form a *convex point set*) and $T(P)$ be its triangulation. A *semi-circled convex polygon* P is a special convex polygon such that its longest edge is the diameter and all the edges of P lie inside the semi-circle with this longest edge as diameter. (Refer to Figure 1).

The following two properties are easy to verify for a semi-circled convex polygon $P = (p_1, p_2, ..., p_k, ..., p_{n-1}, p_n)$.

Property 1: Let $\overline{p_i p_k}$ for $1 \leq i < k \leq n$ be an edge in P. Then the area bounded by $\overline{p_i p_k}$ and chain $(p_i, ..., p_k)$ is a semi-circled convex polygon.

Property 2: In P, edge $\overline{p_i p_j}$ for $1 \leq i < j < n$ or $1 < i < j \leq n$ is shorter than $\overline{p_1 p_n}$.

3 Finding an MAT of a semi-circled convex polygon

Lemma 1 *Let $P = (p_1, p_2, ..., p_k, ..., p_{n-1}, p_n)$ be a semi-circled convex polygon. Then, one of the edges: $\overline{p_1 p_{n-1}}$ and $\overline{p_n p_2}$ must belong to its maximum weight triangulation $MAT(P)$.*

proof: Suppose for contradiction that none of the two extreme edges: $\overline{p_1 p_{n-1}}$ and $\overline{p_n p_2}$ belongs to $MAT(P)$. Then, there must exist a vertex p_k for $2 < k < n-1$ such that both $\overline{p_1 p_k}$ and $\overline{p_n p_k}$ belong to $MAT(P)$. Without loss of generality, let p_k lie on one side

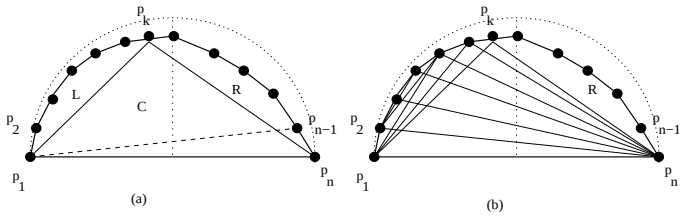


Figure 2: For the proof of Lemma 1.

of the perpendicular bisector of edge $\overline{p_1p_n}$, say the lefthand side (if p_k is on the bisector, the following argument is still applicable), The two edges $\overline{p_1p_k}$ and $\overline{p_np_k}$ partition P into three areas, denoted by R, L, and C. Both L and R are semi-circled convex polygons by Property 1. (Refer to part (a) of Figure 2.)

Let the edges of $MAT(P)$ lying inside area L be $(e_1, e_2, ..., e_{k-3})$ and let $E_L = (e_1, e_2, ..., e_{k-3}, \overline{p_1p_k})$. Let E_L^* denote the edges: $(\overline{p_np_2}, \overline{p_np_3}, ..., \overline{p_np_{k-1}})$. Then, there is a perfect matching between E_L and E_L^*. This is because E_L and E_L^* respectively triangulate the same area $L \cup C$, hence the number of internal edges of the two triangulations must be equal. Let us consider a pair in the matching $(e_i, \overline{p_np_j})$ for $1 < i, j < k$. It is not hard to see that $\omega(e_i) < \omega(\overline{n, j})$ because any edge in E_L is shorter than $\overline{p_1p_k}$ by Property 2, any edge in E_L^* is longer than $\overline{p_np_k}$, and $\overline{p_np_k}$ is longer than $\overline{p_1p_k}$ (due to p_k lying on the lefthand side of the perpendicular bisector of $\overline{p_1p_n}$). Therefore, $\omega(E_L)$ is less than $\omega(E_L^*)$. Now, we shall construct a new triangulation, say $T(P)$, which consists of all the edges in $MAT(P)$ except replacing the edges of E_L by E_L^*. We have that $\omega(T(P)) > \omega(MAT(P))$, which contradicts the $MAT(P)$ assumption. Then, such a p_k cannot exist and one of $\overline{p_1p_{n-1}}$ and $\overline{p_np_2}$ must belong to $MAT(P)$. □

By Lemma 1 and by Property 1, we have a recurence for the weight of a $MAT(P)$. Let $\omega(i, j)$ denote the weight of the $MAT(P_{i,j})$ of a semi-circled convex polygon $P_{i,j} = (p_i, p_{i+1}, ..., p_j)$. Let $\omega(\overline{i, j})$ denote the length of edge $\overline{p_ip_j}$.

$$\omega(i,j) = \begin{cases} \omega(\overline{i,i+1}) & j = (i+1) \\ max\{\omega(i,j-1) + \omega(\overline{j-1,j}), \omega(i+1,j) + \omega(\overline{i,i+1})\} + \omega(\overline{i,j}) & \text{otherwise} \end{cases}$$

It is a straight-forward matter to design a dynamic programming algorithm for finding the $MAT(P)$.

ALGORITHM $MAT - FIND(P)$

Input: a semicircled convex $n-$sided polygon: $(p_1, ..., p_n)$.

Output: $MAT(P)$.

Method:

1. **for** $i = 1$ **to** $n - 1$ **do**

   $\omega(i, i+1) = \omega(\overline{i, i+1})$

2. **for** $l = 2$ **to** $n - 1$ **do**

3. **for** $i = 1$ **to** $n - l$ **do**

$$\omega(i, i+l) = max\{\omega(i, i+l-1)+\omega(\overline{i+l-1, i+l}), \omega(i+1, i+l)+\omega(\overline{i, i+1})\} + \omega(\overline{i, i+l})$$

4. Identify the edges of $MAT(P)$ by checking the ω's.

5. **end.**

Since the loop indices i and l range roughly from 1 to n and each evaluation of $\omega(i, j)$ takes constant time, all $\omega(i, j)$ for $1 \leq i, j \leq n$ can be evaluated in $O(n^2)$ time. If we record these ω's, we can find the edges in $MAT(P)$ by an extra $O(n)$ time to examine the record.

Therefore, we have the following theorem.

Theorem 1 *The maximum weight triangulation of a semi-circled convex $n-$gon P, $MAT(P)$, can be found in $O(n^2)$ time.*

Proof: The correctness is due to Property 1. It is clear that the number of executions of Step 3 dominates the time complexity. The number of executions is $(n-3)+(n-4)+...+2+1)\epsilon O(n^2)$. □

4 Conclusion

In this paper, we proposed an $O(n^2)$ dynamic programming algorithm for constructing the $MAT(P)$ of a semi-circled convex $n-$sided polygon.

It is still an open problem whether one can design an $o(n^3)$ algorithm for finding the $MAT(P)$ for a general convex $n-$sided polygon P.

References

[1] Gilbert P., New results on planar triangulations, Tech. Rep. ACT-15 (1979), Coord. Sci. Lab., University of Illinois at Urbana.

[2] Klincsek G., Minimal triangulations of polygonal domains, *Annual Discrete Mathematics* 9 (1980) pp. 121-123.

[3] Preparata F. and Shamos M., *Computational Geometry* (1985), Springer-Verlag.

[4] Wang C., Chin F., and Yang B., Maximum Weight triangulation and graph drawing, *The Information Processing Letters*, 70(1999) pp. 17-22.

The Minimum Manhattan Network Problem— Approximations and Exact Solutions

Marc Benkert [a], Takeshi Shirabe [b], and Alexander Wolff [a]

[a] *Faculty of Computer Science, Karlsruhe University, Germany. WWW: i11www.ilkd.uka.de/algo/group*

[b] *Institute for Geoinformation, Technical University of Vienna, Austria. Email: shirabe@geoinfo.tuwien.ac.at*

1. Introduction

A *Manhattan p–q path* is a geodesic in the Manhattan (or L_1-) metric that connects p and q, i.e. a staircase path between p and q. Given a set of points P in the plane, a *Manhattan network* is a set of axis-parallel line segments that contains a Manhattan p–q path for each pair $\{p, q\}$ of points in P.

In this paper we consider the *minimum* Manhattan network problem which consists of finding a Manhattan network of minimum total length, an *MMN* in short, i.e. a 1-spanner for the Manhattan metric. The problem is likely to have applications in VLSI layout. Its complexity status is unknown.

The problem has been considered before. Gudmundsson et al. [1] have proposed a factor-8 $O(n \log n)$-time and a factor-4 $O(n^3)$-time approximation algorithm, where n is the number of input points. Later Kato et al. [2] have given a factor-2 $O(n^3)$-time approximation algorithm. However, their correctness proof is incomplete.

In this paper we give a geometric factor-3 approximation algorithm that runs in $O(n \log n)$ time and the first mixed-integer programming (MIP) formulation for the MMN problem. We have implemented and evaluated both approaches.

2. Preliminaries

We will use the notion of a *generating set* that has been introduced in [2]. A generating set is a subset Z of $\binom{P}{2}$ with the property that a network containing Manhattan paths for all pairs in Z is a Manhattan network of P.

The authors of [2] defined a generating set Z with the nice property that Z consists only of a linear number of point pairs. Here we use the same generating set Z, but more intuitive names for the subsets of Z. In [2] $Z = Z_h \cup Z_v \cup Z_x \cup Z_y \cup Z_2$. Here we define $Z = Z_{\text{hor}} \cup Z_{\text{ver}} \cup Z_{\text{quad}}$, where $Z_{\text{hor}} = Z_h \cup Z_y$, $Z_{\text{ver}} = Z_v \cup Z_x$, and $Z_{\text{quad}} = Z_2$. We consider Z_{quad} a set of ordered pairs.

Now let $\mathcal{R}_{\text{hor}} = \{\text{BBox}(p, q) \mid \{p, q\} \in Z_{\text{hor}}\}$, where $\text{BBox}(p, q)$ is the smallest axis-parallel closed rectangle that contains p and q. Note that $\text{BBox}(p, q)$ is just the line segment $\overline{pq}$ if p and q lie on the same horizontal or vertical line. In this case we consider $\text{BBox}(p, q)$ a *degenerate rectangle*. Define $\mathcal{R}_{\text{ver}}$ and $\mathcal{R}_{\text{quad}}$ analogously. Let $\mathcal{A}_{\text{hor}}$, $\mathcal{A}_{\text{ver}}$, and $\mathcal{A}_{\text{quad}}$ be the subsets of the plane that are defined by the union of the rectangles in $\mathcal{R}_{\text{hor}}$, $\mathcal{R}_{\text{ver}}$, and $\mathcal{R}_{\text{quad}}$, respectively. As Kato et al. we start with some basic, yet incomplete network whose length is bounded by the length of an MMN.

Definition 1 *[2] A set of vertical line segments $\mathcal{V}$* covers *$\mathcal{R}_{\text{ver}}$, if for any horizontal line ℓ and any $R \in \mathcal{R}_{\text{ver}}$ with $R \cap \ell \neq \emptyset$ there is a $V \in \mathcal{V}$ with $V \cap \ell \neq \emptyset$. We say that $\mathcal{V}$ is a* minimum vertical cover (MVC) *if $\mathcal{V}$ has minimum length among all covers of $\mathcal{R}_{\text{ver}}$. The definition of a* minimum horizontal cover (MHC) *is analogous.*

Kato et al. have observed the following.

Lemma 2 *[2] The union of an* MVC *and an* MHC *has length bounded by the length of an* MMN.

In general such a union does not satisfy, i.e. connect by Manhattan paths, all pairs in Z_{ver} and Z_{hor}. Additional segments must be added to the union to achieve this. To ensure that the total length of these segments can be bounded, we need covers with a special property. Obviously the segments in an MVC can be moved such that each segment is contained in a vertical edge of a rectangle in $\mathcal{R}_{\text{ver}}$. We say that an MVC is *nice* if additionally each cover segment is incident to a point in P. Note that each vertical rectangle edge contains at most one segment of a nice MVC, since degenerate rectangles do not share edges with other rectangles

and must therefore be covered completely. In order to show that every point set has in fact a nice MVC, we need the following definitions.

For a horizontal line ℓ consider the graph $G_\ell(V_\ell, E_\ell)$, where V_ℓ is the intersection of ℓ with the vertical edges of rectangles in $\mathcal{R}_{\text{ver}}$, and there is an edge in E_ℓ if two intersection points belong to the same rectangle. We say that a point v in V_ℓ is *odd* if the number of points to the left of v that belong to the same connected component of G_ℓ is odd, otherwise v is even. For a vertical edge e of a rectangle in $\mathcal{R}_{\text{ver}}$, let an *odd segment* be an inclusion-maximal connected set of odd points on e. Define *even segments* accordingly. For example, the segment s (drawn bold in Figure 1) of the edge f is an even segment, while $f \setminus s$ is odd. We say that the parity of an edge changes where two segments of different parity touch.

Theorem 3 *Every point set P has a nice* MVC *and a nice* MHC*.*

PROOF. We only show the statement for the vertical case, the horizontal case is analogous. Our proof is constructive. Let $\mathcal{V}$ be the union of all odd segments and all degenerate rectangles in $\mathcal{R}_{\text{ver}}$. Clearly $\mathcal{V}$ covers $\mathcal{R}_{\text{ver}}$. Let ℓ be a horizontal that intersects $\mathcal{A}_{\text{ver}}$. Consider a connected component C of G_ℓ and let k be the number of vertices in C. If k is even then any cover must contain at least $k/2$ vertices of C, and $\mathcal{V}$ contains exactly $k/2$. On the other hand, if $k > 1$ is odd then any cover must contain at least $(k-1)/2$ vertices of C, and $\mathcal{V}$ contains exactly $(k-1)/2$. Thus $\mathcal{V}$ is an MVC.

To see that $\mathcal{V}$ is nice, we consider a vertical edge e of a rectangle in $\mathcal{R}_{\text{ver}}$ and the input point p_0 on e. We show that either e is even or p_0 lies on the only odd segment of e. In both cases $\mathcal{V}$ contains only cover segments that touch an input point.

Wlog. let p_0 be the topmost point of e. Let $p_0, p_1, \ldots, p_k$ be the input points in order of decreasing x-coordinate that span the rectangles in $\mathcal{R}_{\text{ver}}$ that are relevant for the parity of e. Let $p_i = (x_i, y_i)$ and $\overline{y_0} = y_0$, $\overline{y_1} = y_1$. For $2 \le i \le k$ define recursively $\overline{y_i} = \min\{y_i, \overline{y_{i-2}}\}$ if i is even, and $\overline{y_i} = \max\{y_i, \overline{y_{i-2}}\}$ if i is odd. Let $\overline{p_i} = (x_i, \overline{y_i})$, and let $\overline{\mathcal{L}}$ be the polygonal chain through $p_0, p_1, \overline{p_2}, \overline{p_3}, \ldots, \overline{p_k})$ in this order, see Figure 1. Note that the parity of a point v on e is determined by the number of segments of $\overline{\mathcal{L}}$ that intersect the horizontal h through v.

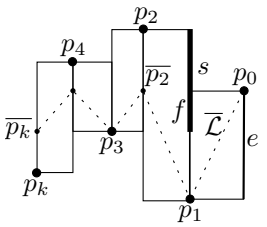


Fig. 1. Proof of Theorem 3.

If h is below $\overline{p_k}$, then it intersects a descending segment for each ascending segment of $\overline{\mathcal{L}}$, hence v is even. If on the other hand h goes through or is above $\overline{p_k}$, then it intersects an ascending segment for each descending segment—plus $\overline{p_1 p_0}$, hence v is odd. So e can change parity only in $(x_0, \overline{y_k})$. □

A simple sweep-line algorithm yields the following.

Lemma 4 *A nice* MVC *and a nice* MHC *can be computed in $O(n \log n)$ time using linear space.*

3. An Approximation Algorithm

Our algorithm ApproxMMN proceeds in three phases, see Algorithm 1. In phase I we compute the generating set $Z_{\text{ver}} \cup Z_{\text{hor}} \cup Z_{\text{quad}}$. In phase II we satisfy all pairs in $Z_{\text{ver}} \cup Z_{\text{hor}}$ by computing a nice MVC $\mathcal{C}_{\text{ver}}$ and a nice MHC $\mathcal{C}_{\text{hor}}$, and by then adding at most one additional line segment for each rectangle $\mathcal{R}_{\text{ver}} \cup \mathcal{R}_{\text{hor}}$. Since each rectangle $R = \text{BBox}(p, q) \in \mathcal{R}_{\text{ver}}$ ($\mathcal{R}_{\text{hor}}$) is covered nicely, it suffices to add a horizontal (vertical) segment whose length is the width (height) of R in order satisfy $\{p, q\}$. Let $\mathcal{S}$ be the set of these additional segments. Consider the vertical strip that is defined by a rectangle $R \in \mathcal{R}_{\text{ver}}$. By definition of $\mathcal{R}_{\text{ver}}$, R is the only rectangle in $\mathcal{R}_{\text{ver}}$ that intersects the interior of the strip. Thus the total length of the additional horizontal (vertical) segments is the width W (height H) of $\text{BBox}(P)$. By Lemma 2 the network $N_1 = \mathcal{C}_{\text{ver}} \cup \mathcal{C}_{\text{hor}} \cup \mathcal{S}$ has length $\le |N_{\text{opt}}| + H + W$, where N_{opt} is a fixed MMN and $|M|$ is the total length of a set M of line segments.

In phase III we satisfy the pairs in Z_{quad}. Let $Q(r, 1) = \{s \in \mathbb{R}^2 \mid x_r < x_s \text{ and } y_r < y_s\}$ be the first quadrant of the Cartesian coordinate system with origin r. Define $Q(r, 2)$, $Q(r, 3)$, $Q(r, 4)$ analogously and in the usual order. Let $P(q, t) = \{p \in P \cap Q(q, t) \mid (p, q) \in Z_{\text{quad}}\}$ for $t = 1, 2, 3, 4$. Let $\Delta(q, t) = \bigcup_{p \in P(q,t)} \text{BBox}(p, q) \setminus \text{int}(\mathcal{A}_{\text{hor}} \cup \mathcal{A}_{\text{ver}})$, where $\text{int}(M)$ denotes the interior of a set $M \subseteq \mathbb{R}^2$. Let $\delta(q, t)$ be the union of those connected components of $\Delta(q, t)$ that are incident to some $p \in P(q, t)$. Note that each connected component A of $\delta(q, t)$ is a staircase polygon—by definition of Z_{quad} there is a strictly x- and y-monotone ordering of the points in $P(q, t)$. The C-hull of A is the union of A and the bounding boxes of neighboring

Algorithm 1 ApproxMMN

Phase I: Compute $Z = Z_{\text{ver}} \cup Z_{\text{hor}} \cup Z_{\text{quad}}$.
Phase II: Satisfy $Z_{\text{ver}} \cup Z_{\text{hor}}$:
  compute a nice MVC $\mathcal{C}_{\text{ver}}$ and a nice MHC $\mathcal{C}_{\text{hor}}$
  compute set $\mathcal{S}$ of additional horizontal (vertical) segments for rectangles in $\mathcal{R}_{\text{ver}}$ ($\mathcal{R}_{\text{hor}}$)
  $N_1 \leftarrow \mathcal{C}_{\text{ver}} \cup \mathcal{C}_{\text{hor}} \cup \mathcal{S}$, $N_2 \leftarrow \emptyset$, $N_3 \leftarrow \emptyset$
Phase III: Satisfy Z_{quad}:
  for each region δ of type $\delta(q,t)$ **do**
    for each connected component A of δ **do**
      compute rectangulation $R_{A'}$ of A'
      $N_2 \leftarrow N_2 \cup \partial A' \cup \{s_{A'}\}$
      $N_3 \leftarrow N_3 \cup (R_{A'} \setminus \partial A')$
    end for
  end for
return $N = N_1 \cup N_2 \cup N_3$

input points on the boundary ∂A of A. It is known that any MMN rectangulates the C-hull of A [1, Lemma 4]. The same holds for a slightly smaller region A' that can be connected to N_1 via at most one segment $s_{A'}$. There is a simple $O(n)$-time factor-2 approximation algorithm B for rectangulating staircase polygons [1]. We use B to compute the rectangulation $R_{A'}$ of each A'. Let N_2 be the union of all $\partial A'$ and all $s_{A'}$. Let N_3 be the union of the rectangulations $R_{A'}$ without $\partial A'$. Our algorithm returns the line segments in $N = N_1 \cup N_2 \cup N_3$.

To bound the length of N we partition the plane into two regions and compare N to N_{opt} in each region separately. Region $\mathcal{A}_3$ is the union of $\text{int}(A')$ over all areas of type A', while $\mathcal{A}_{12} = \mathbb{R}^2 \setminus \mathcal{A}_3$. We have $N_1 \cup N_2 \subseteq \mathcal{A}_{12}$ and $N_3 \subseteq \mathcal{A}_3$, and the interiors of different regions of type A do not intersect. On the one hand the approximation factor of B yields that $|N \cap \mathcal{A}_3| \leq 2|N_{\text{opt}} \cap \mathcal{A}_3|$. On the other hand we can show that $|N_2| \leq 2|N_{\text{opt}}| - (H+W)$. Thus $|N \cap \mathcal{A}_{12}| = |(N_1 \cup N_2) \cap \mathcal{A}_{12}| \leq 3|N_{\text{opt}} \cap \mathcal{A}_{12}|$, which in turn yields $|N| \leq 3|N_{\text{opt}}|$.

Theorem 5 *A 3-approximation of an* MMN *can be computed in $O(n \log n)$ time and $O(n)$ space.*

4. A MIP Formulation

In this section we give the first MIP formulation of the MMN problem. This formulation gives us the possibility to implement an exact solver for the MMN problem that can solve small examples in a bearable amount of time. Those will be used as benchmarks for our approximation algorithm.

We need some notation: For a set P of n input points $p_1(x_1, y_1), \ldots, p_n(x_n, y_n)$ let $x^1 < \cdots < x^u$ and $y^1 < \cdots < y^w$ be the ascending sequences of x- respectively y-coordinates of the input points. The grid Γ induced by P consists of the *grid points* (x^i, y^j) with $i = 1, \ldots, u$ and $j = 1, \ldots, w$. In this section we will only consider pairs $\{p, q\} \in Z$ with $x_p \leq x_q$. This is no restriction since we can flip the names of p and q. For each such pair let $V(p,q) = \Gamma \cap \text{BBox}(p,q)$ and let $A(p,q)$ be the set of arcs between horizontally or vertically adjacent grid points in $V(p,q)$. Horizontal arcs are always directed from left to right, vertical arcs point upwards (downwards) if $y_p < y_q$ ($y_p > y_q$). Our formulation is based on the *grid graph* $G_P(V, A)$, where $V = \bigcup_{\{p,q\} \in Z} V(p,q)$ and $A = \bigcup_{\{p,q\} \in Z} A(p,q)$. Let $E = \{\{g, g'\} \mid (g, g') \in A \text{ or } (g', g) \in A\}$ be the set of undirected edges.

For each pair $\{p, q\} \in Z$ we enforce the existence of a p–q Manhattan path by a flow model as follows. We introduce one 0–1 variable $f(p, q, g, g')$ for each arc (g, g') in $A(p, q)$, which encodes the size of the flow along arc (g, g'). For each grid point g in $V(p, q)$ we introduce the flow constraint

$$\left.\begin{array}{l} \displaystyle\sum_{(g,g') \in A(p,q)} f(p,q,g,g') \\ \displaystyle- \sum_{(g',g) \in A(p,q)} f(p,q,g',g) \end{array}\right\} = \begin{cases} +1 & \text{if } g = p, \\ -1 & \text{if } g = q, \\ 0 & \text{else.} \end{cases} \quad (1)$$

Next we introduce a continuous variable $F(g, g')$ for each edge $\{g, g'\}$ in E. This variable will in fact be forced to take a 0–1 value by the objective function and the following constraints. The MMN that we want to compute will consist of all grid edges $\{g, g'\}$ with $F(g, g') = 1$. We now add one or two constraints for each $\{g, g'\}$ in E and each $\{p, q\} \in Z$ with $\overline{gg'} \subseteq \text{BBox}(p, q)$:

$$F(g,g') \quad \geq \quad \begin{cases} f(p,q,g,g') & \text{if } (g,g') \in A, \\ f(p,q,g',g) & \text{if } (g',g) \in A. \end{cases} \quad (2)$$

Note that the two conditions are not mutually exclusive. Our objective function expresses the total length of the selected grid edges:

$$\text{min!} \sum_{\{g,g'\} \in E} |gg'| \cdot F(g,g'), \quad (3)$$

where $|gg'|$ is the Euclidean distance of g and g'.

This MIP formulation uses $O(n^3)$ variables and constraints. By treating pairs in Z_{quad} more carefully, a reduction to $O(n^2)$ is possible, see full paper. It is not hard to see that our formulation always yields an MMN:

Theorem 6 *Let P be a set of points and let Z, A, and E be defined as above. Let $F : E \to \mathbb{R}_0^+$ and $f : Z \times A \to \{0, 1\}$ be functions that fulfill (1) & (2) and minimize (3). Then the set of line segments $\{\overline{gg'} \mid \{g, g'\} \in E,\ F(g, g') \geq 1\}$ is an* MMN *of P.*

Due to our objective function (3), Equation (1) can be replaced by an inequality (with direction $\geq$). If the resulting constraint matrix was totally unimodular (every square submatrix has determinant in $\{-1, 0, +1\}$), every vertex of the solution polyhedron would be integral and the MMN problem would in fact correspond to an LP. Unfortunately it turned out that this is not the case and that there are instances with fractional vertices that minimize our objective function.

5. Experiments

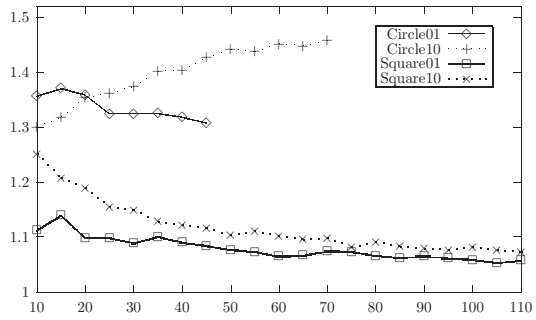
We used two different types of random instances.

Square$k$ instances were generated by drawing n different points with uniform distribution from a $kn \times kn$ integer grid. We wanted to see the effects of having more (k small) or less (k large) points with the same x- or y-coordinate. If a pair of points shares a coordinate, the manhattan path connecting them is uniquely determined.

Circle$k$ instances consist of a point p_1 at the origin and $n - 1$ points on the upper half of the unit circle. The points are distributed as follows. The interval $I = [0, \pi/4]$ is split into k subintervals $I_1, \ldots, I_k$ of equal length. We used $k \in \{1, 2, 5, 10\}$. Then $n - 1$ random numbers $r_2, \ldots, r_n$ are drawn from I. If the number r_i falls into a subinterval of even index, it is mapped to the point $p_i = (\cos r_i, \sin r_i)$ otherwise to $p_i = (-\cos r_i, \sin r_i)$. The resulting points p_i (except for the topmost point in each quadrant and the "bottommost" point in each subinterval) all form pairs $\{p_i, p_1\}$ that are in Z_{quad}. This makes Circle instances very different from Square instances where only few point pairs belong to Z_{quad}.

We generated instances of the above types and solved them with ApproxMMN and with Cplex using the MIP formulation of Section 4. We implemented ApproxMMN in C++ using the compiler gcc-3.3. The asymptotic runtime of our implementation is $\Theta(n^2)$, the real runtime was measured on an AMD Athlon 1800+ with 512 MB RAM under Linux-2.4.20. To compute exact solutions we used the LP Barrier Solver of ILOG Cplex-9.0 on an IBM RS/6000. The results of our experiments can be found in the diagram below. The sample size, i.e. the number of points per instance, is shown on the x-axis. For each sample size we generated 30 instances and averaged the results over those. The y-axis shows the performance ratio of ApproxMMN, i.e. the ratio of the length of the network computed by ApproxMMN over the length of the MMN computed by Cplex.

Cplex ran out of memory on Circle01 instances of more than 45 points and on Square10 instances of more than 110 points. Below these thresholds ApproxMMN always had a performance ratio below 1.55, which is much better than what the approximation factor of 3 suggests. While the ratio seems to approach 1 on Square instances of increasing size, the picture is not so clear for Circle instances:



The runtime of ApproxMMN was practically independent of the type of instance. The CPU times we measured reflected the quadratic asymptotic runtime. 500 points took roughly 0.3 seconds.

The exact solver depended much more on the type of instance than the approximation algorithm. It solved Square instances much faster than Circle instances. This is due to the fact that pairs in Z_{quad} require a quadratic number of variables and constraints in our MIP formulation, while pairs in Z_{ver} and Z_{hor} need only a linear number. 100 points of type Square$k$ took 0.6–1.6 seconds and 6–13 seconds for $k = 1$ and 10, respectively, while 40 points of type Circle$k$ took 61–480 seconds and 1.2–2.4 seconds for $k = 1$ and 10, respectively.

References

[1] J. Gudmundsson, C. Levcopoulos, and G. Narasimhan. Approximating a minimum Manhattan network. *Nordic J. Comput.*, 8:219–232, 2001.

[2] R. Kato, K. Imai, and T. Asano. An improved algorithm for the minimum Manhattan network problem. In *Proc. ISAAC'02*, vol. 2518 of *LNCS*, pages 344–356, 2002.