An $O(n \log n)$ Algorithm for the All-Farthest-Segments Problem for a Planar Set of Points

R. L. Scot Drysdale* Asish Mukhopadhyay†

Abstract

In this paper, we propose an algorithm for computing the farthest-segment Voronoi diagram for the edges of a convex polygon and apply this to obtain an $O(n \log n)$ algorithm for the following proximity problem: Given a set P of $n(> 2)$ points in the plane, we have $O(n^2)$ implicitly defined segments on pairs of points. For each point $p \in P$, find a segment from this set of implicitly defined segments that is farthest from p. We improve the previously known time bound of $O(nh + n \log n)$ for this problem, where h is the number of vertices on the convex hull of P.

1 Introduction

There is an extensive literature on Voronoi diagrams [1, 3], including algorithms for computing farthest-point and closest-segment diagrams. However, little seems to be known about the problem of computing farthest-segment Voronoi diagram for a set of line segments in the plane. Our particular interest in this paper is the construction of the farthest-segment Voronoi diagram for the edges of a convex polygon. This was triggered by an attempt to improve an $O(nh + n \log n)$ time solution [10] for the following problem: Given a set P of $n(> 2)$ points in the plane, for each point $p \in P$ find a segment determined by two other points of P that is farthest from p. The distance from a point p to a segment is the minimum distance from p to this segment. Indeed, we have been successful in improving the time complexity of the latter problem to $O(n \log n)$.

The rest of the paper is structured as follows. In the next section, we briefly review prior work on the farthest segment Voronoi diagram and the problems of computing the nearest and farthest segment from among a set of segments implicitly defined by a planar set of n points. In the following section we briefly review the algorithm of Mukhopadhyay et al. [10]. In the fourth section we outline our improvement to this algorithm, while in the last two sections we discuss the properties of the farthest-segment Voronoi diagram for edges of a convex polygon and an algorithm for its construction.

*Department of Computer Science, Dartmouth College, scot@cs.dartmouth.edu

†School of Computer Science, University of Windsor, asishm@cs.uwindsor.ca.

2 Prior Work

While there appear to be no references in the literature to the problem of computing the farthest-segment Voronoi diagram of a given set of line segments, the work by Barequet et al. [2] discusses how to compute what are called 2-site Voronoi diagrams; various distance functions from an arbitrary point in the plane to pairs of points taken from of a set of n sites are considered. One distance function considered is the distance to the segment defined by a pair of sites, but the diagram they consider is for all $\binom{n}{2}$ possible segments as opposed to a selected set of segments. For the case when all points are on a convex polygon, their farthest-pair diagram is equivalent to the farthest-segment Voronoi diagram for edges of the polygon. Several of the properties that they derive for their diagram therefore apply to ours. They give no explicit algorithm for computing this diagram.

For the *nearest* version of the above proximity problem, Daescu and Luo [4] presented an $O(n \log n)$ algorithm; Duffy et al [8] presented an $O(n^2)$ algorithm for the *all-nearest* version, and also provided evidence that this might be an $\Theta(n^2)$-hard problem. Daescu and Luo [4] also presented an $O(n \log n)$ algorithm for the *farthest* version of this problem (see also [5]). Mukhopadhyay et al [10] showed that the *all-farthest* version of the problem can be solved in $O(nh+n \log n)$ time, where h is the number of vertices on the convex hull of the n points.

3 The Previous Algorithm

In [10], the authors give a simple $O(nh + n \log n)$ algorithm for solving the all-farthest-segments problem. If $\overline{p_j p_k}$ is a farthest segment from the point p_i, it is classified as a member of one of two groups, depending on the location of c_i, the closest point of $\overline{p_j p_k}$ to p_i. Type A segments are those where c_i is an interior point of $\overline{p_j p_k}$, while type B segments are those where c_i is an endpoint of $\overline{p_j p_k}$.

They prove the following lemmas that characterize type A and type B segments, assuming that no three points are co-linear (which we will also assume).

Lemma 1 *If the segment $\overline{p_j p_k}$ is a type A farthest segment for a point p_i then $\overline{p_j p_k}$ is an edge on the convex hull of P.*

Lemma 2 *If the segment $\overline{p_j p_k}$ is a type B farthest segment for a point p_i then either $\overline{p_j p_k}$ is an edge on the convex hull of P or p_j is farthest from p_i among all the points that are interior to the convex hull of the point set, while p_k is a convex hull vertex of P.*

This leads to their algorithm. For each point p_i they find the convex hull edge e_i that is farthest from p_i and the interior point p_j that is farthest from p_i. Given p_j they find a convex hull vertex p_k whose distance from p_i is greater than p_j's distance from p_i. They report the farther of e_i and $\overline{p_j p_k}$.

They compute the farthest point from p_i via point location in a pre-computed farthest-point Voronoi diagram of the points of P which lie interior to the convex hull of P. This requires $O(\log n)$ time per point. They find e_i and p_k by a linear search of the convex hull. This requires $O(h)$ per point. Thus their overall run-time is in $O(nh + n \log n)$.

4 Our Algorithm

We use the algorithm described above, but improve the run time by avoiding a linear search of the convex hull.

To find p_k, a hull point farther from p_i than p_j is, we find the intersection of the convex hull and the ray $\overrightarrow{p_i p_j}$. We consider the endpoints of the segment intersected by the ray.

Lemma 3 *The farther of the two endpoints from p_i must be further from p_i than p_j is, and that endpoint is chosen as p_k.*

Proof: The ray $\overrightarrow{p_i p_j}$ intersects the convex hull boundary that lies in the half-plane, defined by a line orthogonal to p_j, that does not contain p_i (see Fig. 1). Now, at least one of the endpoints of the convex hull edge that the ray intersects must lie in the same half-plane by convexity arguments. If p_k is this endpoint then the angle $\angle p_i p_j p_k$ is obtuse and hence $\overline{p_i p_k} > \overline{p_i p_j}$. If the other endpoint of the intersected segment also belongs to this half-plane, the same argument also applies to it; in this case we set p_k to be the one that is farther from p_i. □

Because the convex hull is a convex polygon, we can use binary search to find the intersection of the ray and the convex hull. This takes $O(\log h)$ time.

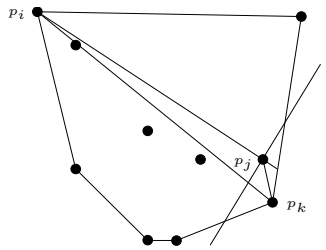

Figure 1: *The point p_k is farther from p_i than p_j*

To find e_i, the edge of the convex hull furthest from p_i, we locate p_i in a farthest-segment Voronoi diagram of the edges of the convex hull. In the next section we show that this diagram has complexity $O(h)$, so the point location can be done in $O(\log h)$ time. We also show that the diagram can be computed in $O(h \log h)$ time.

The preprocessing requires computing a farthest-point Voronoi diagram of the points of P that lie interior to the convex hull of P and a farthest-segment Voronoi diagram of the edges of the convex hull of P. This takes a total of $O(n \log n)$ time. For each point we perform a point location in both Voronoi diagrams and a binary search, requiring $O(\log n)$ time per point. The entire algorithm therefore runs in $O(n \log n)$ time.

5 Properties of the farthest-segment Voronoi diagram

We define the farthest-segment Voronoi diagram in the standard way. Given a set S of n line segments in the plane, we define a (possibly empty) region for each line segment $s \in S$. Let $d(p, s)$ to be the Euclidean distance from point p to the closest point on segment s. The Voronoi region of a segment s is defined to be $V(s) = \{p | d(p, s) \geq d(p, s') \forall s' \in S\}$.

This gives a diagram where points equidistant from two or more segments lie in multiple regions. For disjoint segments these regions of overlap consist of one-dimensional curves: rays, segments, and pieces of parabolas. These curves form the boundaries between regions. However, if two segments can share an endpoint then all points closer to the shared endpoint than to the interior of the segments are equidistant from both segments, and the overlap contains a two-dimensional cone. To avoid the computational problems that arise from this we somewhat arbitrarily define the angle bisector as the boundary curve between the points "closer" to one segment and the points "closer" to the other segment. (Eliminating this two-dimensional

overlap region by defining a boundary curve is standard when computing Voronoi diagrams of segments [6, 7, 9]).

The general farthest-segment Voronoi diagram has properties that are quite different from those of the farthest-point Voronoi diagrams. In the farthest-point diagram a point's region is always convex, and only those points lying on the convex hull have non-empty regions [11]. In contrast, in the farthest-segment Voronoi diagram, segments which have no points on the convex hull can have non-empty regions and the region assigned to a segment can be disconnected. There are no published algorithms for computing the general farthest-segment Voronoi diagram.

However, for our application we need only compute the farthest-segment Voronoi diagram for the edges a convex polygon. This greatly simplifies the task. Figure 2 shows the farthest-segment Voronoi diagram of a set of line segments that form a quadrilateral.

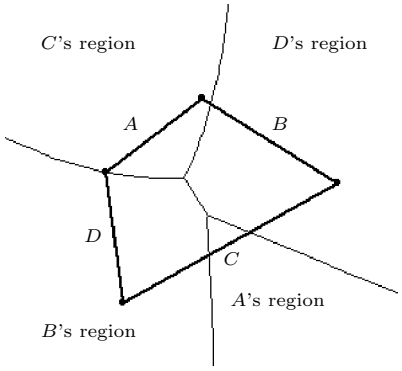

Figure 2: *Farthest-segment Voronoi diagram for a quadrilateral*

We first prove some properties of the farthest-segment Voronoi diagram for this special case. The facts that regions are infinite and that the size of the diagram is $O(n)$ also follow from Barequet et al. [2] .

Lemma 4 *In the farthest-segment Voronoi diagram for a convex polygon each segment s has a non-empty, connected, infinite Voronoi region, $V(s)$. The Voronoi region boundaries of the diagram form a tree and the size of the diagram is $O(n)$.*

Proof: Let s be a an edge of the convex polygon and l its supporting line. To prove our claim about s, consider any ray r perpendicular to s that passes through the interior of the polygon. Consider a circle C tangent to s centered on r. As its center moves away from s (see Fig. 3), its radius increases, and C approaches l in the limit. Since the polygon is convex, all other segments have their interiors in the half-plane

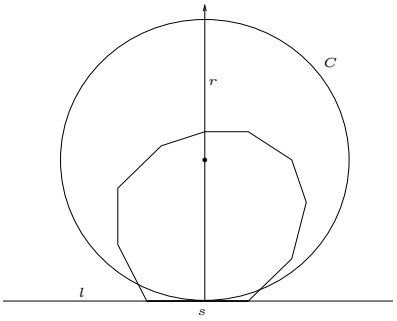

Figure 3: *A circle C tangent to s that intersects all edges of the convex hull*

that is bounded by l and contains the interior of the polygon. Therefore, when C is large enough at least one point of every other segment will lie within C, and all points on r further from s than the center of this circle will be within $V(s)$. This proves that $V(s)$ is non-empty and infinite.

To prove that $V(s)$ is connected, we show that given any two points p and q in $V(s)$, we can construct a path between them lies entirely in $V(s)$. Consider the ray r with endpoint p that is perpendicular to l and points away from l. We claim that r lies within $V(s)$. To see this, consider the circle C centered at p which passes through the nearest point in s. If C is tangent to s then by the previous argument all points on the ray r from p, pointing away from s, are in $V(s)$.

Otherwise, C will intersect l twice, once at an endpoint of s and once somewhere else. As the center of C moves away from l along r the two intersection points will remain the same. The part of C on the far side of l will shrink and the part of C on the near side of l will grow. Because none of the segments can lie on the far side of l any segment points in the circle centered at p will also lie in every larger circle. Therefore all points on r will be in $V(p)$.

We further note that because C approaches l in the limit it is possible to find a point on r such that the circle centered there that is tangent to l contains a point from every other segment. If we construct a ray r' at q in the same way we constructed r at p then r' will lie completely in $V(s)$, and we can find a circle centered on r' tangent to l that contains a point from every other segment. If we slide the larger of these two circles along l from one point of tangency to the other, from continuity arguments, this circle will always contain at least one point of every other segment. Therefore the segment s' traced by the center of the sliding circle will lie in $V(s)$. This gives us a path from p to q - go out r to s', cross s' to r', and follow r' back down to q. This

shows that $V(s)$ is connected.

A similar argument can be used to show that each region's boundary is connected. Each segment's region shares a boundary with both of its neighboring segments, so it is possible to get from any region to any other region by following segments around the polygon. A connected boundary with no finite regions can have no cycles, so must be a tree. The diagram is a planar graph and the bisector between two segments contains a constant number of curves. From these facts we can conclude that the diagram has size $O(n)$. □

6 Computing the farthest-segment Voronoi diagram of a convex polygon

We can compute the farthest-segment Voronoi diagram using a divide-and-conquer algorithm. The final merge joins two halves of the polygon. Earlier merges join two consecutive chains of segments around the polygon.

There are two parts to this merge. The first is finding an infinite part of the bisector curve between the two sets of segments to be merged. This becomes the starting curve for the merge. The two parts of the convex polygon to be merged share a vertex (or two in the case of the final merge). The two segments adjacent to this vertex will each have an infinite region, and these regions will be adjacent. Therefore there will be an infinite bisector separating these two regions. The final portion of this bisector will be a part of of the perpendicular bisector of the non-shared segment endpoints. It will go to infinity in the direction away from the segments. This is because the two segment endpoints are the closest points on the segments in this direction.

The second part is to do the standard segment Voronoi diagram merge step, tracing the bisector between points closer to a segment in the first group and points closer to a segment in the second group. This merge uses a clockwise/counter-clockwise merge step that is very similar to the one presented in the literature for closest-segment Voronoi diagrams [6, 7, 9]. The one difference is that the farthest-segment diagram merge keeps the parts of the merged diagrams that the closest-segment merge discards and discards the portions that the closest-segment merge keeps. (This is exactly the same relationship that the merge step for computing the farthest-point Voronoi diagram has to the merge step for computing the closest-point diagram.)

The papers on computing closest-segment Voronoi diagrams prove that the merge procedure traces the bisector curve correctly. The only difficulty that arises is "islands," which are closed loops of the bisector curve. Lemma 4 guarantees that there will be no islands, because all regions are infinite.

Theorem 5 *A divide-and-conquer algorithm can be used to compute the farthest-segment Voronoi diagram of an n-sided convex polygon in time $O(n \log n)$.*

Proof: The algorithm and a proof of its correctness are given above. The standard segment Voronoi diagram merge step runs in time proportional to the size of the two diagrams to be merged. Therefore the divide-and-conquer algorithm runs in $O(n \log n)$ time. □

References

[1] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric structure. *ACM Comput. Surv.*, 23(3):345–405, 1991.

[2] G. Barequet, M. T. Dickerson, and R. L. S. Drysdale. 2-point site voronoi diagrams. *Discrete Applied Mathematics*, 122:37–54, 2002.

[3] B. Boots, A. Okabe, and K. Sugihara. *Spatial Tesselations: Concepts and Applications of Voronoi Diagrams*. John Wiley, 1992.

[4] O. Daescu and J. Luo. Proximity problems on line segments spanned by points. In *Proc. of 14th Annual Fall Workshop on Computational Geometry*, pages 9–10, 2004.

[5] O. Daescu, J. Luo, and D. M. Mount. Proximity problems on line segments spanned by points. In *Proc. of 17th Canadian Conference on Computational Geometry*, pages 224–228, 2005.

[6] R. Drysdale. *Generalized Voronoi Diagrams and Geometric Searching*. Stan-cs-79-705, Department of Computer Science, Stanford University, Stanford, California, 1979.

[7] R. Drysdale and D. Lee. Generalized voronoi diagrams in the plane. In *Proc. of the 16th Allerton Conference on Communications, Control, Computing*, pages 833–842, Oct 4-6 1978.

[8] K. Duffy, C. McAloney, H. Meijer, and D. Rappaport. Closest segments. In *Proc. of CCCG 2005*, pages 229–231, 2005.

[9] D. Lee and R. Drysdale. Generalization of voronoi diagrams in the plane. *SIAM J. Comput.*, 10(1):73–87, 1981.

[10] A. Mukhopadhyay, S. Chatterjee, and B. Lafreniere. On the all-farthest-segments problem for a planar set of points. In *Abstracts of the 22nd Eoropean Workshop on Computational Geometry*, pages 47–49, March 27-29 2006.

[11] M. I. Shamos and D. Hoey. Closest-point problems. In *Proc. 16th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 151–162, 1975.