Separating and Shattering Long Line Segments ⋆

Alon Efrat

School of Mathematical Sciences, Tel Aviv University, Tel-Aviv 69982, Israel.
Email: alone@cs.tau.ac.il

Otfried Schwarzkopf

Dept of Computer Science, Postech, Hyoja-Dong, Pohang 790-784, South Korea.
Email: otfried@postech.ac.kr

A line l is called a *separator* for a set S of objects in the plane if l avoids all the objects and partitions S into two non-empty subsets, lying on both sides of l. A set L of lines is said to *shatter* S if each line of L is a separator for S, and every two objects of S are separated by at least one line of L. We give a simple randomized algorithm to construct the set of all separators for a given set S of n line segments in expected time $O(n \log n)$, provided the ratio between the diameter of S and the length of the shortest line segment is bounded by a constant. We also give a randomized algorithm to determine a set of lines shattering S, whose expected running time is $O(n \log n)$, improving (for this setting) the (deterministic) $O(n^2 \log n)$ time algorithm of Freimer, Mitchell and Piatko.

Key words: computational geometry, BSP-trees, line-separation.

1 Introduction

Given a set S of n objects in the plane, we say that a line l *avoids* S if l avoids every element in S. We call a line l a separator for S if l avoids S and partitions S into two non-empty subsets. In this paper we consider the problem of finding separators for a set of line-segments. Clearly this is sufficient to treat the case of general polygonal objects as well.

⋆ Work on this paper by the second author has been supported by the Netherlands' Organization for Scientific Research (NWO), by Pohang University of Science and Technology Grant 96F004, 1996, and partially by the nondirected research fund of the Korean Ministry of Education.

For a set S of line segments, a separator can be found using duality. Under duality, the segments transform to double wedges, and a separator line transforms to a point lying in the complement of the union of these double wedges, but not above all or below all of them. In other words, we can construct the set of all possible separators by computing the union of these double wedges in dual space. This can be done by constructing the arrangement of the $2n$ lines that are dual to the endpoints of the segments in S, and then determining for every face of the arrangement whether it is inside one of the double wedges. All this can be done in $O(n^2)$ time using for example the topological sweep technique of Edelsbrunner and Guibas [5].

On the other hand, it is easy to see that there are sets of n line segments for which the union of double wedges—and hence the set of all possible separators—has complexity $\Omega(n^2)$. So it seems that the algorithm sketched above is already the best one can do.

In fact, even if we do not need to compute the set of all possible separators, but are just interested in finding some arbitrary separator (if one exists at all), it is not known whether the problem can be solved in subquadratic time. In fact, it can be shown that the problem belongs to the class of so-called *three-sum hard problems*. The problems in this class are suspected to admit no solution in subquadratic time [10].

In this paper we investigate the special case in which the segments of S are *long*. By this, we mean that the ratio of the diameter of S and the length of the shortest segment is bounded by a constant. Or, rescaling the problem, we will assume that the segments are contained in the unit disk and their length is at least a constant $\lambda > 0$.

As Efrat, Rote, and Sharir [7] observe, the dual wedge—under a suitable duality transformation—of a line segment whose slope is bounded away from $90°$ is *fat*, meaning that its interior angle is bounded from below by a constant $\delta = \delta(\lambda) > 0$. It can be shown [7,12], that the overall complexity of the union of n fat double wedges is linear in n, with the constant of proportionality depending on δ, and this union can also be computed in close to linear time. Efrat, Rote, and Sharir used this fact to obtain an algorithm to find a separator for a set of long segments: They partition the set of segments into two subsets depending on their slope, use the above observation to conclude that the dual wedges for both subsets have linear complexity, and finally overlay the two unions until they find a point corresponding to a separating line, or determine that no such point exists. The algorithm is randomized, and its expected running time is $O(n \log n)$, or, more precisely, $O(n/\lambda^2 \log(1/\lambda) \log(n/\lambda))$.

In this paper we will show that the union of the dual wedges of a set of long segments actually has only linear complexity. As a consequence, we obtain a randomized algorithm that computes *all* separators for the set of segments (instead of just one) in the same expected time, and that is considerably simpler than the algorithm by Efrat et al. Furthermore, our algorithm can be extended to find a separator that partitions the segments as equally as possible, or even to find a line that stabs only a small number of segments. This is, for instance, attractive for the construction of binary space partition trees (BSP-trees).

We say that a set of segments S is *shatterable* if there exists a set of lines L avoiding S and such that every pair of segments is separated by at least one line in L. If such a set L exists, we say that L shatters S. Freimer, Mitchell and Piatko [8] gave an $O(n^2 \log n)$ time algorithm to find a linear-cardinality set L that shatters S, or to determine that no such L exists. We show that for a set of long segments this can be done in expected time $O(n \log n)$. We also show that $\Omega(n)$ lines are always needed to shatter a set of n long segments (in contrast to the case of arbitrary segments), and hence not much can be gained by looking for smaller shattering sets.

2 The combinatorial bound

Let S be a set of n line segments in the plane. We assume that they are all contained in the unit disk centered at the origin, and that their length is bounded from below by a constant $\lambda > 0$. We also assume (without loss of generality) that no segment in s has slope $30° + i \cdot 60°$, for $i = 0, 1, 2$.

We dualize every segment s to a double wedge s^* that is the union of all lines p^*, where p is a point on e. (We use the duality transformation that maps a point (a, b) to a line $y = ax - b$, and a line $y = cx - d$ to the point (c, d).) A non-vertical line l meets s if and only if its dual point l^* lies inside the double wedge s^*. It is not difficult to see that a non-vertical line l is a separator for S if and only if l^* lies in the complement of the union of the double wedges for the elements of S, but not above or below *all* the double wedges.

We will denote the union of a set S^* of double wedges by $\mathcal{U}(S^*)$, and prove the following simple lemma.

Lemma 1 *For a set S of line segments, let S_θ denote S rotated by an angle θ around the origin. Then the complexity of $\mathcal{U}(S^*)$ is the same as the complexity of $\mathcal{U}(S^*_\theta)$.*

PROOF. Note that vertex v of $\mathcal{U}(S^*)$ corresponds to a line in the primal plane that avoids S and is tangent to two of the segments of S. There is a line partitioning S_θ in the same manner, and hence there is a one-to-one correspondence between the vertices of $\mathcal{U}(S^*)$ and those of $\mathcal{U}(S^*_\theta)$. $\square$

We can now prove the main theorem.

Theorem 2 *Let S be a set of n long line segments. Then the complexity of the union of the double wedges s^*, for $s \in S$, is $O(n)$.*

PROOF. We partition the set S of segments into three subsets, depending on the angle that the segments make with the x-axis: segments in S_1 make an angle between 0° and 60°, segments in S_2 make an angle between 60° and 120°, and segments in S_3 make an angle between 120° and 180°; see Figure 1.

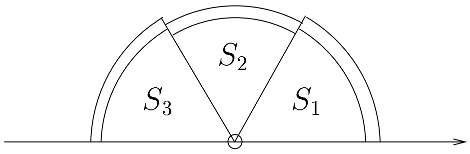

Fig. 1. Dividing S into S_1, S_2 and S_3, based on the angles the segments create with the x-axis

A vertex of $\mathcal{U}(S^*)$ is either the center of a double wedge, or is the intersection between the boundaries of two double wedges. There is only a linear number of vertices of the first kind, so it is sufficient to count only the second kind of vertices.

Consider a vertex v of $\mathcal{U}(S^*)$ that is formed by the boundary of the double wedges s^* and t^*. Let $i, j \in \{1, 2, 3\}$ be such that $s \in S_i$, $t \in S_j$. Clearly, v is also a vertex of $\mathcal{U}(S^*_i \cup S^*_j)$. It therefore suffices to show that the complexity of $\mathcal{U}(S^*_i \cup S^*_j)$ is $O(n)$ for all choices of i and j.

By rotating by 60° in either direction, we can reduce all cases to the case of $S_1 \cup S_3$. By Lemma 1, it therefore suffices to prove that the complexity of $\mathcal{U}(S^*_1 \cup S^*_3)$ is $O(n)$.

The segments of $S_1 \cup S_3$ have slope between $-60°$ and $+60°$. As observed before, their dual wedges therefore have an interior angle larger than some constant $\delta = \delta(\lambda) > 0$. Using the result on the union of fat wedges [7,12], it follows that the complexity of $\mathcal{U}(S^*_1 \cup S^*_3)$ is $O(n)$, completing the proof. $\square$

3 Computing all separators and other applications

The set of all separators is the complement of the union of the double wedges $\mathcal{S}^*$ (minus the two regions above and below all the double wedges). By Theorem 2, the complexity of this set is linear. We can compute it quite easily using a randomized incremental algorithm, inspired by the algorithm by Miller and Sharir [12,13] for the computation of the union of fat-triangles. [1]

Our algorithm treats the wedges in random order, maintaining the trapezoidation (vertical decomposition) of the complement of the union of the wedges so far. We also maintain, in what is by now a quite standard fashion, a history graph of the trapezoidation as defined by Boissonnat et al. [2]. The nodes of this directed acyclic graph are the trapezoids created during the course of the computation, and its leaves are the trapezoids of the current trapezoidation. A child trapezoid in the graph intersects the parent trapezoid, and all children of a trapezoid together cover its parent completely. Any node has at most a constant number of children (at most six, to be precise).

In every step, we have to identify the trapezoids intersected by or contained in the new double wedge. This can be done in a standard way by a traversal of the history graph until we reach the leaf nodes (which correspond to the trapezoids in question). We then update those trapezoids to create the new trapezoidation. This updating can be done in time linear in the number of trapezoids deleted and created in this step as in the case of the trapezoidation of a set of segments considered by Boissonnat et al. [2].

The analysis of the algorithm is quite standard: The expected number of trapezoids created in stage r of the algorithm is proportional to the average number of trapezoids incident to a double wedge in a set of r double wedges. By Theorem 2 this number is constant. It follows that the expected number of trapezoids created by the algorithm, and therefore the expected size of the history graph, is $O(n)$. The remaining running time of the algorithm is dominated by the time necessary for the graph traversals. The standard analysis for randomized incremental construction [2,4] for first-order moments can be applied to show that the expected time for this step is $O(n \log n)$.

Theorem 3 *Given a set S of n long segments, the set of all lines separating S can be computed in expected time $O(n \log n)$.*

It is easy to extend the algorithm to compute a separator that splits S as equally as possible. This is useful for the computation of *binary space partition trees* (BSP-trees) in the plane.

[1] A triangle is called *δ-fat* if all its angles are at least δ.

We will not go into details about the construction of BSP-trees. We just remark that it is based on repeatedly finding lines that partition a set of segments while intersecting only a few of them. If a set of segments admits no separator, or if all its separators split off only a small number of segments, it is therefore useful to find a line that splits it as equally as possible while intersecting not more than k of the segments, where k is a parameter.

Our argument can be extended to deal with this situation. A line l intersects at most k segments when it lies in the $\leqslant k$-level of the set of double wedges dual to the segments of S. Using Clarkson and Shor's analysis [3,4], it can be shown that under our conditions on the set S, the complexity of the $\leqslant k$-level is $O(kn)$. The algorithm can be extended to compute it in time $O(nk \log n)$, and we can find the line that splits S as equally as possible while intersecting only k segments in the same time.

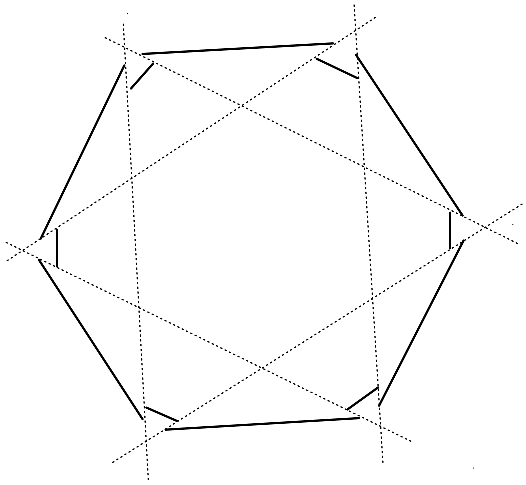

Fig. 2. The set of 12 segments S (plotted by solid lines) is shatterable, but no separator has more than 3 segments on one of its sides.

A somewhat surprising fact, observed by Sariel Har-Peled [9], is that a set of segment S can be shatterable, and still have no separator l that divides S in a *balanced* way, that is, at least a constant fraction of S lies at each side of l. This is demonstrated in Figure 2. We do not know yet whether such an example can be found for long segments.

4 The shattering problem

A related problem is the problem of shattering a set S of line segments. We consider two problems:

(i) Given a set S of n line segments, find a set L of $O(n)$ lines shattering S.

(ii) Given a set S of n line segments and a set L of $O(n)$ lines, determine whether L shatters S.

A general algorithm (that is, not assuming that the line segments are long) for the first problem was given by Freimer, Mitchell and Piatko [8]; its running time is $O(n^2 \log n)$. The first step in this algorithm is to create a set of lines that are suspected to be separators for S. These lines are the lines containing the visibility edges of the visibility graph of S. Their number is $\Theta(n^2)$ in the worst case.

For the case that the segments are long, we can limit our interest to a much smaller set: As we have seen above, there are only $O(n)$ combinatorially distinct separators for the set of line segments. Clearly, if the set S can be shattered at all, then the set of these separators must do the job. Hence, we can reduce the first problem to the second one in expected time $O(n \log n)$.

In the rest of this section we will therefore concentrate on the second problem. It can be solved as follows: We select a representative endpoint for every line segment in S, resulting in a point set P. We then compute all the faces in the arrangement $\mathcal{A}(L)$ of the lines L that contain at least one point of P. After that, we perform point location queries with the points of P to determine whether there are two points in P lying in the same face.

In the general case, the faces of the arrangement $\mathcal{A}(L)$ containing the points P can be computed in time $O(n^{4/3} \log n)$ time using the algorithm of Edelsbrunner, Guibas and Sharir [6].

To improve on that, we first prove that, if the segments in S are long, the total complexity of all faces in the arrangement containing at least one long segment is only linear.

Lemma 4 *Given a set S of long line segments, and a set L of m lines. Then the total complexity of all faces of the arrangement $\mathcal{A}(L)$ containing at least one segment of S is $O(m)$.*

PROOF. We cover the unit circle with a grid of $O(1/\lambda)$ horizontal and vertical lines H such that every cell of the arrangement of H inside the unit circle has diameter less than λ. That implies that every segment in S must intersect a line of H. Consequently, every face of $\mathcal{A}(L)$ that contains a segment of S lies in the zone of one of the lines H. By the zone theorem, the total complexity of all these faces is $O(m/\lambda)$. $\square$

The idea of the proof immediately leads to an algorithm for our problem: We create a set H of $O(1/\lambda)$ lines as in the proof of the lemma, and compute all faces of $\mathcal{A}(L)$ that intersect at least one line of H. That can be done in expected time $O(n \log n)$ using a lazy randomized incremental algorithm [1]. We then compute a point location structure for the subdivision we have obtained so far, and perform $2n$ point location queries with the endpoints of the segments S. This takes time $O(n \log n)$ in total. If the two endpoints of a segment in S do not lie in the zone of H, or do not lie in the same face of the zone, then the set L contains lines that do not separate S. If there are two endpoints of different line segments that lie in the same face of the subdivision, then the two line segments are not separated by L. Otherwise, L shatters S. To summarize, we have shown the following theorem.

Theorem 5 *Given a set S of n long segments, and a set L of $\Theta(n)$ lines, we can determine in expected time $O(n \log n)$ if L shatters S.*

Corollary 6 *Given a set S of n long segments, we can determine in expected time $O(n \log n)$ a set L of $O(n)$ lines shattering S, or determine that no such set exists.*

5 The number of lines needed to shatter a set

The algorithm described above does not try to minimize the number of shattering lines. The following theorem proves that such a minimization cannot decrease the size of the shattering set by more than a constant factor.

Theorem 7 *Let S be a set of long line segments, and assume that a set L of lines shatters S. Then $|L| = \Omega(n)$.*

PROOF. By our assumption, each element $s \in S$ is contained inside the unit disk D. Let $\mathcal{A}(L)$ denote the arrangement formed by L. The perimeter of the cell r_s of $\mathcal{A}(L)$ containing s is either infinity, or at least 2λ. Therefore, as is easily shown using the triangle inequality, the length of the region of the boundary of r_s, contained inside D, is at least λ; see Figure 3. Summing these lengths over all segments $s \in S$, we see that the total length of the segments $\{l_i \cap D \mid l_i \in L\}$ is at least $(\lambda/2)|S|$, since both sides of such a segment $l_i \cap D$ can contribute to the boundaries of two cells r_s and $r_{s'}$, for some $s,\ s' \in S$.

On the other hand, for each $l_i \in L$ the length of the segment $l_i \cap D$ is at most 2. Hence

$$2|L| \geqslant \frac{\lambda}{2}|S|$$

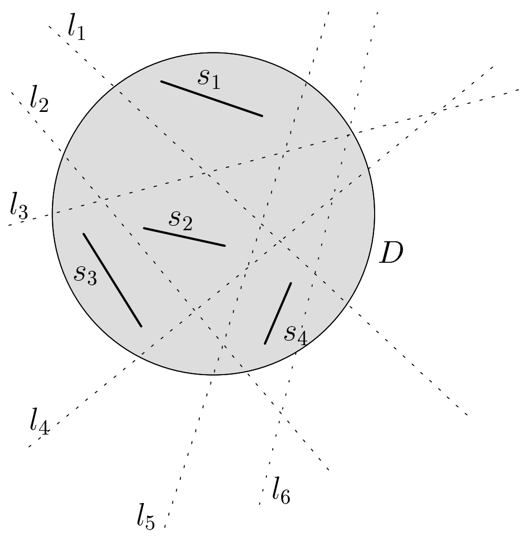

Fig. 3. The set of lines $L \equiv \{l_1, \dots l_6\}$ shatters $S \equiv \{s_1, \dots, s_4\}$

yielding $|L| = \Omega(|S|) = \Omega(n)$ $\square$

Acknowledgment. Discussion with Alon Itai contributed significantly to this paper, and we are grateful to him.

References

[1] M. de Berg, K. Dobrindt, and O. Schwarzkopf. On Lazy Randomized Incremental Construction. *Proceedings 26 Annual ACM Symposium on Theory of Computing*, 1994, pages 105–114.

[2] J.-D. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec. Applications of random sampling to on-line algorithms in computational geometry. *Discrete and Computational Geometry* 8 (1992), 51–71.

[3] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete and Computational Geometry* 2 (1987), 195–222.

[4] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete and Computational Geometry* 4 (1989), 387–421.

[5] H. Edelsbrunner and L.J. Guibas. Topologically sweeping an arrangement. *J. Comput. Syst. Sci.* 38 (1989), 165–194.

[6] H. Edelsbrunner, L. Guibas, and M. Sharir. The complexity and construction of many faces in arrangements of lines and of segments. *Discrete and Computational Geometry* 5 (1990), 161–196.

[7] A. Efrat, M. Sharir, and G. Rote. On the union of fat wedges and separating a collection of segments by a line. *Computational Geometry: Theory and Applications* 3 (1994), 277–288.

[8] R. Freimer, J. S. B. Mitchell, and C. D. Piatko. On the complexity of shattering using arrangements. *Proceedings 2 Canadian Conf. Computational Geometry*, 1990, pages 218–222.

[9] Sariel Har-Peled. Private communication.

[10] A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry: Theory and Applications*, 5 (1995), 165–185.

[11] J. Hershberger and S. Suri. Applications of a semi-dynamic convex hull algorithm. *Proceedings 2 Scan. Workshop on Algorithms Theory*, *Lecture Notes in Computer Science*, 1990, vol. 447, Springer-Verlag, New York, pages 380–392.

[12] J. Matoušek, N. Miller, J. Pach , M. Sharir, S. Sifrony, and E. Welzl. Fat triangles determine linearly many holes. *Proceedings 32 Annual ACM Symposium on Theory of Computing*, 1991, pages 49–58.

[13] N. Miller and M. Sharir. Efficient randomized algorithms for constructing the union of fat triangles and of pseudodiscs. Manuscript, 1991.