Counting and reporting intersections in arrangements of line segments

L.J. Guibas, M.H. Overmars, M. Sharir

RUU-CS-89-12
May 1989

Rijksuniversiteit Utrecht

Vakgroep informatica

Padualaan 14 3584 CH Utrecht
Corr. adres: Postbus 80.089, 3508 TB Utrecht
Telefoon 030-531454
The Netherlands

Counting and Reporting Intersections in Arrangements of Line Segments

Leonidas J. Guibas[1,2], *Mark H. Overmars*[3] *and Micha Sharir*[4,5]

[1] Computer Science Department, Stanford University
[2] DEC Systems Research Center, Palo Alto, CA.
[3] Department of Computer Science, University of Utrecht
[4] Courant Institute of Mathematical Sciences, New York University
[5] School of Mathematical Sciences, Tel Aviv University

ABSTRACT

We present efficient algorithms for counting and reporting all intersections in an arrangement of n line segments in the plane. Specifically, we have a randomized algorithm for finding all k intersections in such an arrangement in expected time $O(n^{4/3+\delta} + k)$, for any $\delta>0$, and linear working storage. A variant of the algorithm counts the number of intersections in $O(n^{4/3+\delta})$ randomized expected time, for any $\delta>0$ and linear space. Our techniques are based on recursive decomposition of the problem into subproblems of smaller size, using plane partition methods that involve random sampling of the given segments, akin to the techniques of [HW] and [Cl].

1. Introduction

We begin with the following somewhat unusual opening remarks. The results of this paper have been conceived and developed in the summer of 1987, but for various reasons were left unpublished for almost two years. Meanwhile, there has been significant progress on the problems studied here. In particular, a recent work by Agarwal [Ag] presents techniques that transform our algorithm into a deterministic and slightly faster one, and also extend our algorithm to handle red-blue intersection problems, in which we want to count or report all intersections between two collections of segments, while ignoring intersections between pairs of segments in the same collection. (However, the working storage in [Ag] is no longer linear.) Agarwal's results are based on a decomposition technique that is different from the one used in this paper. Still, many of Agarwal's arguments are taken from this paper. As a public service, to make our results somewhat more accessible, we have decided to elevate our paper from the status of "unpublished manuscript" to that of a technical report.

Work on this paper by the third author has been supported by Office of Naval Research Grant N00014-82-K-0381, by National Science Foundation Grant No. NSF-DCR-83-20085, by grants from the Digital Equipment Corporation, and the IBM Corporation, and by a research grant from the NCRD - the Israeli National Council for Research and Development.

2. Efficient Calculation of the Intersections of Line Segments

Let $G = \{e_1, \ldots, e_n\}$ be a collection of line segments in the plane. We derive in this paper a randomized algorithm for counting all k intersections of the given segments in expected time $O(n^{4/3+\delta})$, for any $\delta>0$, and $O(n)$ space. Recently, Chazelle and Edelsbrunner [CE] have obtained a time-optimal algorithm for reporting all k intersections; their algorithm runs in $O(n \log n + k)$ time and uses $O(n+k)$ space. If k is small, this algorithm is perhaps the method of choice for counting intersections as well, but when k is large (k can be $\Theta(n^2)$ in the worst case) it becomes quite inefficient. The best previous solution to the intersection counting problem is due to Chazelle [Ch] and runs in time $O(n^{1.695})$. An improved $O(n^{1.5}\log n)$ algorithm is given in [MS] for the special case in which we want to count the intersections between two collections of n segments each, where the segments in each collection are non-intersecting; however, this algorithm has been superceded in a recent paper [CEGS], where an $O(n \log n)$ algorithm is presented.

Our algorithm can be extended to report all k intersections in randomized expected time $O(n^{4/3+\delta})$, for any $\delta>0$, using only linear working storage. When k is small, this is much worse than the algorithm of [CE], but for large values of k our algorithm becomes more attractive, because it uses only linear working storage. However, Clarkson [Cl2] has recently obtained a randomized algorithm that reports all intersections in expected time $O(n \log n + k)$ and linear working storage, which has made the extension of our algorithm somewhat obsolete. We describe it below anyway, because it is relatively simple and the ideas that it employs may be useful for other purposes (as indeed is the case in [Ag]).

Let $p_1, \ldots, p_m$ ($m \le 2n$) be the endpoints of the given segments. We construct a partition-tree T for this set of points, with a predetermined set of query lines, consisting of the lines $l_1, \ldots, l_n$ containing the segments $e_1, \ldots, e_n$. The tree T is constructed top-down in a recursive manner. With each node v of T there is associated a convex polygonal region Q_v (obtained from the plane partitionings done at the ancestors of v), the subset P_v of the endpoints p_i which lie inside Q_v, and the subset L_v of the query lines l_j that intersect Q_v. The recursive goal at v is to report (or count) all intersections of the segments e_i which lie within Q_v. We put $n_v = |P_v|$, $m_v = |L_v|$.

The root of T corresponds to the entire plane. Let v be a node of T. If $m_v \ge n_v^2$ we do not continue the construction of T below v (so v is a leaf of T), and instead apply the procedure described below to obtain all intersections within Q_v. Otherwise we partition Q_v into a collection of subregions, using a technique akin to the ϵ-net approach of Haussler and Welzl [HW] or the random sampling technique of Clarkson [Cl]. Thus we fix some integer r, draw a random sample of r of the lines in L_v, clip each sample line to obtain its portion within Q_v, construct the arrangement of these clipped portions, and triangulate each face of this arrangement. By the ϵ-net theory, with high probability, each of the resulting $M = O(r^2)$ triangles will be cut by at most $O(\frac{cm_v}{r} \log r)$ lines of L_v, for some constant $c>0$. We then create M children $w_1, \ldots, w_M$ of v; each w_i is assigned one of the triangles Q_{w_i} of this "sample arrangement", the corresponding subset P_{w_i} of the points of P_v within Q_{w_i}, and the subset L_{w_i} of the lines in L_v which cut Q_{w_i}. In addition, we also record the (at most two) intersection points of each line l_j in L_{w_i} with ∂Q_{w_i}; if the portion of l_j within Q_{w_i} is disjoint from e_j we exclude l_j from L_{w_i}. The overall cost of this expansion step at v is $O(m_v+n_v)$ (randomized) time.

To achieve space efficiency, the tree T is constructed in a depth-first manner. Thus at any given time we maintain only a single path within T. Moreover, in passing from a node v to one of its children w, the lines in L_w are taken away from L_v, and are placed back there upon returning from w. It is easy to see that in this way only linear space is needed to maintain T.

The heart of our procedure is the processing of nodes v of T that lie at the bottom of the recursion. Let v be such a node. There are n_v endpoints of segments within Q_v, so Q_v contains (portions of) at most n_v segments having an endpoint within it. On the other hand $m_v \geq n_v^2$ lines l_j go through Q_v, so for the majority of these lines, the corresponding segment e_j has no endpoint within Q_v, and thus it cuts all the way through that face. Let A_v denote the set of such segments, and B_v the complementary set of segments having an endpoint within Q_v. Thus $|B_v| \leq n_v$, $|A_v| \leq m_v$.

The intersection-reporting procedure at v consists of three substeps. Finding intersections (within Q_v) among the segments in A_v, finding intersections (within Q_v) among the segments in B_v, and finding intersections (within Q_v) between segments in A_v and segments in B_v.

Intersections within A_v.

We have m_v segments $g_1, \ldots, g_{m_v}$, each of which starts and ends on the circumference of the convex region Q_v. Let a_i, b_i denote the endpoints of g_i, $i=1, \ldots, m_v$. Note that any intersection between a pair of these segments must lie within the convex hull C of the points a_i, b_i. Our first step is thus to calculate C, in time $O(m_v \log m_v) = O(m_v \log n_v)$, from which we also obtain the circular sequence $c_1, c_2, \ldots, c_{2m_v}$ of the points a_i, b_i in their clockwise order along ∂C (note that each of these points actually appears along ∂C).

Note that two segments g_i, g_j in A_v intersect within C if and only if their four endpoints appear in interleaving order along ∂C (i.e. on each portion of ∂C between a_i and b_i there is one endpoint of g_j). This suggests the following simple approach.

Process the points $c_1, \ldots, c_{2m_v}$ in order. We maintain a stack S of segments of A_v, which initially is empty. For each point c_k, if it is the first endpoint of some segment g_j, we push g_j on the stack S. If c_k is the second endpoint of g_j, we scan S backwards from its top, and report the intersection of g_j with each segment g_i on S, until we encounter g_j, which is then deleted from the stack. This procedure runs in time $O(m_v+t)$, where t is the number of intersections of segments in A_v within C. (Note that if all we need is to *count* the number of these intersections, we can store S as a balanced search tree and modify the above procedure so that it only counts how many segments lie in S between g_j and the top of S. This yields an $O(m_v \log m_v) = O(m_v \log n_v)$ counting procedure.)

Intersections within B_v.

This is a very simple task to achieve. We simply check every pair of segments in B_v for intersection, and report (or count) the resulting intersections. This takes only $O(n_v^2)=O(m_v)$ time.

Intersections between A_v and B_v.

It is clear that a segment $g \in A_v$ intersects a segment $e \in B_v$ if and only if the line l containing g intersects e. We can therefore regard A_v as a set of lines rather than of segments. It is well known that a line l intersects a segment $e=ab$ if and only if the dual point l^* of l lies in the double wedge (not containing any vertical line) formed between the two dual lines a^*,b^* of the endpoints of e.

Passing to the dual plane, the problem can then be re-formulated as follows. Given a collection of n_v double wedges $W_1, \ldots, W_{n_v}$, formed by $2n_v$ lines, and m_v points $q_1, \ldots, q_{m_v}$, report for each q_i the subset of wedges W_j containing it. Again, since we can afford quadratic complexity (in n_v), this task is not difficult, and can be accomplished by the following line sweeping procedure.

Sweep a vertical line L through the plane from left to right, and maintain a sorted list H of the $2n_v+1$ vertical intervals along L delimited by its intersections with the $2n_v$ wedge boundaries. With each interval I of H we associate a list V_I of all the wedges containing I. If we reach during the sweep a point q_i, we locate it in H and report all the wedges in V_I, where $I \in H$ is the interval containing p_i.

If we sweep through an intersection of two wedge boundaries l,l' we update the list structures along L as follows. Generally, one interval I of H has to be replaced by another interval I'. If l and l' are boundaries of the same wedge, $V_{I'}=V_I$ and no further changes are needed. If l,l' bound distinct wedges W,W', the value of $V_{I'}$ depends on the nature of l,l', according to the following cases (where l is assumed to lie above l' to the left of their intersection).

(i) l and l' are top boundaries of W,W'. Then $V_{I'}=(V_I-\{W\})\cup\{W'\}$.

(ii) l and l' are bottom boundaries. $V_{I'}=(V_I-\{W'\})\cup\{W\}$.

(iii) l is a top boundary and l' is a bottom boundary. $V_{I'}=V_I-\{W,W'\}$.

(iv) l is a bottom boundary and l' is a top boundary. $V_{I'}=V_I\cup\{W,W'\}$.

Thus each intersection between wedge boundaries can be processed in $O(\log n_v)$ time, and thus the entire procedure runs in time $O((m_v+n_v^2)\log n_v + t)$ $= O(m_v \log n_v + t)$, where t is the number of desired intersections, and in space $O(n_v^2)=O(m_v)$. Combining all three subprocedures, we conclude that we can report all k intersections within Q_v in time $O(m_v \log n_v + k)$ and space $O(m_v)$.

We can now analyze the time performance of the entire algorithm. Arguing as in [EGS], it follows that the time $T(m_v,n_v)$ needed to process recursively a node v of the tree obeys the following recurrence relationship (where k_v is the number of intersections within the corresponding region Q_v).

$$T(m_v,n_v) = O(m_v \log n_v + k_v), \quad \text{if } m_v \ge n_v^2 ,$$

$$T(m_v,n_v) = \sum_{i=1}^{M} T(m_{w_i},n_{w_i}) + O(m_v+n_v), \quad \text{if } m_v < n_v^2 ,$$

where $M = O(r^2)$, $m_{w_i} = O(\frac{m_v}{r} \log r)$ for all i, and $\sum_{i=1}^{M} n_{w_i} = n_v$. One can then show, as in [EGS], that the solution of this recurrence satisfies

$$T(m_v, n_v) = O(m_v^{2/3-\delta}\, n_v^{2/3+2\delta} + (m_v+n_v) \log n_v + k_v)$$

for any $\delta>0$. Substituting the root of the tree in this formula, we obtain

Theorem 2.1. One can report all k intersections between n given segments in randomized expected time $O(n^{4/3+\delta} + k)$, for any $\delta>0$, using $O(n)$ working storage.

Next suppose we want just to count how many intersections occur between the n given segments. It is easily checked that each of our three procedures at the bottom of the recursion can be appropriately modified so as to yield the number of corresponding intersections, rather than report all of them, in time $O(m_v \log n_v)$. We have already noted this for the first and second procedures (intersections within A_v and within B_v). As to the third procedure, every time we sweep through one of the given m_v points we can simply add the cardinality of the corresponding list V_I to the running total sum, rather than report that list (in this case it suffices just to maintain the size of each such list, rather than the list itself). We thus obtain

Theorem 2.2. The number of intersections between n given line segments can be calculated in randomized expected time $O(n^{4/3+\delta})$, for any $\delta>0$, and in $O(n)$ space,

We also note that the term $O(n^{4/3+\delta})$ in the above algorithms may in practice be a gross over-estimation of the actual complexity. This is because we avoid propagating down the tree lines l_j for which the corresponding segment e_j does not contain the face Q_v of the current node v. Thus, if the segments e_j do not intersect in too many points, we can expect each line l_j to reach far fewer nodes of T than is implied by this bound. (Of course, if the number of intersections k exceeds this bound, then the reporting algorithm actually run in $O(k)$ time.)

References

[Ag] P.K. Agarwal, A deterministic algorithm for partitioning arrangements of lines and its applications, *Proc. 5th ACM Symp. on Computational Geometry*, 1989, to appear.

[Ch] B. Chazelle, Reporting and counting segment intersections, *J. Computer Systems Sciences* 32 (1986) pp. 156-182.

[CE] B. Chazelle and H. Edelsbrunner, An optimal algorithm for intersecting line segments in the plane, *Proc. 29th IEEE Symp. on Foundations of Computer Science*, 1988, pp. 590-600.

[CEGS] B. Chazelle, H. Edelsbrunner, L. Guibas and M. Sharir, Algorithms for bichromatic line segment problems and polyhedral terrains, in preparation.

[Cl] K. Clarkson, New applications of random sampling in computational geometry, *Discrete Comput. Geom.* 2 (1987) pp. 195-222.

[Cl2] K. Clarkson, Applications of random sampling in computational geometry, II, *Proc. 4th ACM Symp. on Computational Geometry*, 1988.

[EGS] H. Edelsbrunner, L. Guibas and M. Sharir, The complexity of many faces in arrangements of lines and of segments, *Proc. 4th ACM Symp. on Computational*

Geometry, 1988, pp. 44-55.

[HW] D. Haussler and E. Welzl, Epsilon nets and simplex range queries, *Discrete Comput. Geom.* 2 (1987) pp. 127-151.

[MS] H. Mairson and J. Stolfi, Reporting and counting intersections between two sets of line segments, *Theoretical Foundations of Computer Graphics and CAD,* (R. Earnshaw, Ed.), NATO ASI Series, Vol. F-40, Springer Verlag, Berlin 1988, pp. 307-325.