BIT 22 (1982), 274–281

STABBING LINE SEGMENTS

H. EDELSBRUNNER [1], H. A. MAURER [1], F. P. PREPARATA [2], A. L. ROSENBERG [3], E. WELZL [1], and D. WOOD [4]

1) *Institut für Informationsverarbeitung, Technical University of Graz, Schiesstattgasse 4a, A-8010 Graz, Austria*

2) *Coordinated Science Laboratory, University of Illinois, 1101 W. Springfield Avenue, Urbana, Illinois 61801, U.S.A.*

3) *Department of Computer Science, Duke University, Durham, North Carolina 27706, U.S.A.*

4) *Unit for Computer Science, McMaster University, Hamilton, Ontario L8S 4K1, Canada*

Abstract.

An algorithm for the geometric problem of determining a line (called a stabbing line) which intersects each of n given line segments in the plane is presented. As a matter of fact, the algorithm computes a description of all stabbing lines. A purely geometric fact is proved which infers that this description requires $O(n)$ space to be specified. Our algorithm computes it in $O(n \log n)$ time which is optimal in the worst case.

Using the description of the stabbing lines, we are able to decide in $O(\log n)$ time whether or not a specified line is a stabbing line. Finally, the problem of maintaining the description of all stabbing lines while inserting and deleting line segments is addressed.

Keywords: Computational geometry, elementary geometry, divide-and-conquer, plane-sweep, geometric transform, data structures, dynamization.

1. Introduction.

In the last few years, a flurry of activity can be observed in the design of algorithms for geometric problems. We interpret this as being caused by the growing interest in the manipulation of graphical data (e.g. in computer graphics, architectural design, geography, etc.) and also caused by the fact that human intuition is often helped by using a geometric setting for practical problems (e.g. in database organization, VLSI design, etc.). The geometric scenario favors the design of more efficient solutions as well as their transparent presentation.

This paper investigates a geometric problem that is closely related to reachability and visibility questions in the plane. Lee [6] and Lee, Preparata [7] studied the problem of finding optimal tours which avoid certain obstacles. Edelsbrunner, Overmars, and Wood [3] examined several facets of the generalized

Received April 13, 1982.

hidden line problem in the plane. Both types of problems are closely related since tours as well as visual rays have to avoid obstacles.

Beside this relationship to issues of obvious practical interest, the problem we are dealing with is of interest in its own right: Given a set of (potentially intersecting) line segments in the plane, determine a line, called a *stabbing line*, which intersects each segment in the set. In the visibility environment this problem may be interpreted as: find a direction from which one can look through a set of doors, where a door is an interval on an arbitrarily directed line.

Astonishingly, there seems to be no trivial brute-force method for this problem. As far as the authors know, the first algorithm that solves the problem is an application of a very general method due to Edelsbrunner, Overmars, and Wood [3]. However, due to its generality, their algorithm lacks a great deal of efficiency, i.e. it requires $O(n^2 \log n)$ time to find a stabbing line for n line segments.

Our tailor-made algorithm not only improves the time requirements to $O(n \log n)$; it also computes a description of all stabbing lines. From this description one can derive in constant time a specific stabbing line (provided one exists). Furthermore, this description is well suited for supporting a related searching problem, i.e. determine for specified query lines whether they are stabbing lines or not.

The organization of the paper is as follows. Section 2 provides the geometric preliminaries needed for the algorithm presented in Section 3. The related searching problem in a static and dynamic environment is briefly discussed in Section 3. Finally, Section 4 reviews our results and gives some open problems.

2. Geometric preliminaries.

This section provides the geometric tools and facts needed for the design and analysis of our algorithm which computes a description of all stabbing lines. At the base of our algorithm lies a geometric transform which is reviewed below. A more detailed discussion is found in Brown [1]. Furthermore, nontrivial properties about the intersection of so-called double wedges are presented.

The geometric transform, which we call T, maps a point into a line and a line into a point.

$$T\colon\ p = (a, b) \quad \to\ T_p\colon y = ax + b,$$

$$l\colon y = kx + d \quad \to\ T_l = (-k, d).$$

Note that difficulties may arise if vertical lines are to be transformed.

The reader can easily verify the following three facts which are crucial to our algorithm.

FACT 2.1. T retains the relative positions of points and lines, i.e. a point p lies above a line l if and only if the point T_l lies below the line T_p.

A line segment s is fully determined by its two endpoints. The two endpoints are transformed into two lines which determine four wedges. Then s is transformed into the two opposed wedges whose union does not contain a vertical line in its interior, see Figure 2.1.

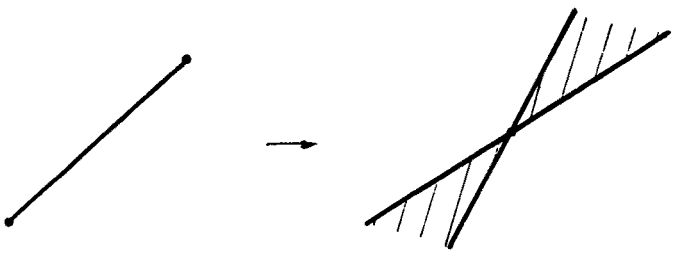

Fig. 2.1. A line segment and its transform under T.

We call these two wedges the *double wedge* corresponding to a line segment. A vertical line segment (a special case) is mapped into the area between the two parallel lines corresponding to the endpoints.

FACT 2.2. A line l intersects a line segment s if and only if the point T_l lies in the double wedge T_s of s.

As an immediate consequence we obtain a characterization of the points corresponding to the stabbing lines for a given set of line segments.

FACT 2.3. The stabbing lines for a set of line segments stand in one-to-one correspondence with the intersection points of their double wedges, see Figure 2.2.

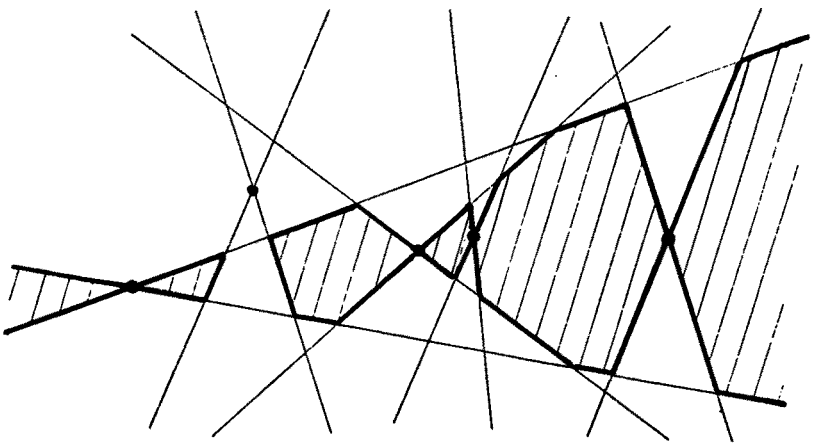

Fig. 2.2. Intersection of double wedges.

Hence, the intersection of the double wedges, which we call the *stabbing region*, serves as a description of all stabbing lines.

Note that the segments may allow only vertical stabbing lines. In this case, the associated stabbing region consists solely of singular points at infinity. This can be handled by treating either the vertical stabbing lines separately or the infinite parts of the stabbing region very carefully. Either approach can be followed without affecting the asymptotic complexity of our algorithm.

We now consider relevant properties about the stabbing region for n line segments. Subsequently, Section 3 focusses on the computation of the stabbing region.

LEMMA 2.4: *The stabbing region for n line segments consists of at most $n+1$ convex and potentially unbounded polygons whose orthogonal projections onto the x-axis do not intersect except potentially at their endpoints.*

PROOF: Note first that the two wedges of a double wedge may be separated by the vertical line through the intersection point of the two lines determining the double wedge. This fact immediately implies that the orthogonal projections onto the x-axis do not properly intersect and that the stabbing region consists of at most $n+1$ polygons. Since each polygon is the intersection of convex wedges, it is also convex. ■

Since the size of the description of a stabbing region constitutes a trivial lower bound for the time needed to construct it and since this size is at least as big as the total number of edges bordering the polygons of a stabbing region, we are interested in the latter quantity.

THEOREM 2.5: *The number of edges bordering the stabbing region for n line segments in the plane is bounded by $O(n)$.*

PROOF: Let w denote an arbitrary double wedge determined by two lines intersecting at a point p. Then w is bordered by four half-rays emanating from p which we term w's upper right ray (i.e. the half-ray on the line with the greater slope leaving p to the right), upper left ray, lower left ray, and lower right ray.

Let S denote the stabbing region for n line segments consisting of m polygons $P_1, P_2, \ldots, P_m$, such that P_i lies to the left of P_j provided i is smaller than j. We classify the edges of each polygon by the kind of half-ray supporting them. We prove that each class of half-rays is associated with at most $2n+1$ edges.

Without loss of generality consider edges associated with upper right rays only. Assign to each polygon P_i, for $i = 1, \ldots, m$, the list of upper right rays which determine the bordering edges of P_i. Obviously, only the upper right ray with minimal slope in the list for polygon P_i can also determine a bordering edge for a polygon P_j, for j greater than i. Now, remove the upper right rays with

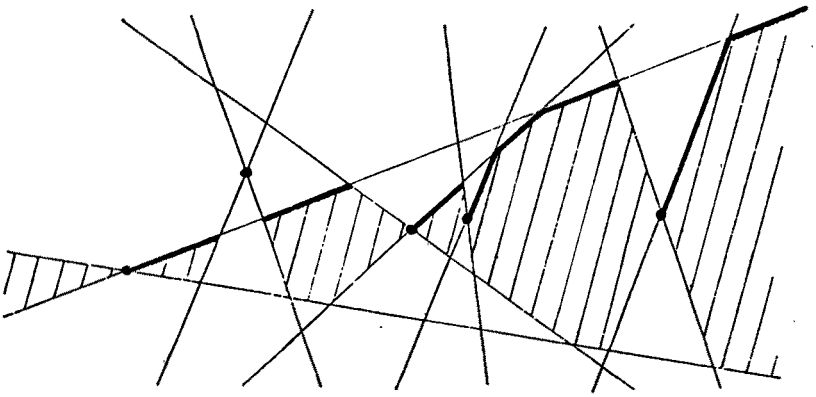

Fig. 2.3. Stabbing region with relevant edges marked.

minimal slope from the m (potentially empty) lists. Each upper right ray occurs at most once in these reduced lists. Consequently, the original lists collectively contain at most n half-rays with non-minimal slope and at most m half-rays with minimal slope, respectively.

As m is no more than $n+1$ and the same argument holds for the remaining 3 types of half-rays, we conclude that $8n+4$ is an upper bound for the number of edges. ■

Clearly, $8n+4$ edges is not the tightest possible upper bound. We conjecture that it can be improved considerably as our best example gives rise to $6n-2$ edges.

3. Computing the stabbing region.

Let us briefly restate the problem. Given a set S of n line segments in the plane, compute all stabbing lines, i.e. the lines which intersect all segments. The algorithm does this by determining the stabbing region for the segments. It uses the divide-and-conquer heuristic and the plane sweep technique for the merge step.

CASE 1. If there is only one line segment in the set S, the stabbing region is the double wedge of this segment.

CASE 2. Otherwise (i.e. if there are at least two segments) divide the set S into two equal size subsets, and compute the stabbing region of both subsets recursively. Compute the stabbing region of the whole set by intersecting the stabbing regions of the subsets.

LEMMA 3.1: *The intersection of two stabbing regions for n_1 and n_2 segments can be computed in $O(n_1+n_2)$ time.*

PROOF. Recall that the stabbing region for n segments consists of at most $n+1$ convex polygons whose orthogonal projections onto the x-axis do not intersect except potentially at their endpoints (Lemma 2.4).

The intersection of two stabbing regions for n_1 and n_2 segments is computed by the application of the plane-sweep technique. A stabbing region is represented by the list of vertices sorted with respect to the x-coordinate. Additionally, each vertex is associated with the two (or potentially four) adjacent vertices of the (potentially two) polygons it belongs to.

A vertical line sweeping from left to right intersects at most one polygon of a stabbing region at a time. Consequently, it intersects at most four edges of both stabbing regions, so all required computations can be accomplished in constant time. Some care has to be taken in the efficient handling of the intersections which occur. Note that at each position of the sweeping line there are at most four anticipated intersections, which can be treated in constant time each. ■

As an immediate consequence we obtain:

THOREM 3.2: *The stabbing region for n line segments in the plane can be computed in $O(n \log n)$ time.*

PROOF: The assertion follows directly from Lemma 3.1 and our divide-and-conquer strategy: the time $T(n)$ required by our algorithm is $2T(n/2)$ to solve the two subproblems of size $n/2$, plus $O(n)$ to combine these solutions into a solution for the entire problem. Hence, $T(n) = 2T(n/2)+O(n) = O(n \log n)$. ■

Obviously, the algorithm above for constructing the stabbing region is optimal in the worst case, since we can reduce the sorting of n reals to an instance of the stabbing region problem. Although this does not imply that our algorithm is optimal for finding a stabbing line or for deciding whether or not there exists a stabbing line, we conjecture that there are no algorithms for the latter tasks which require asymptotically less time than our algorithm.

Let us finally address a searching problem related to the problem of finding stabbing lines. Given a set of n line segments in the plane, decide whether or not a given line intersects all line segments. The discussion above suggests the use of the stabbing region for this problem as well.

THEOREM 3.3: *There exists a data structure for n line segments in the plane that requires $O(n)$ space and $O(n \log n)$ time for construction which enables a query of the form: is a given line a stabbing line or not, to be answered in $O(\log n)$ time.*

PROOF: By Brown's transform, this problem reduces to that of deciding whether or not a given point resides in any of $n+1$ convex polygons whose orthogonal

projections onto the x-axis do not properly intersect. Due to Theorems 2.5 and 3.2, the polygons require $O(n)$ space and $O(n \log n)$ time for construction. Furthermore, a query can be answered in $O(\log n)$ time by using binary search. ■

In dynamic environments, the data structure has to support insertions and deletions of line segments as well.

THEOREM 3.4: *The stabbing region for a set of n line segments can be maintained with $O(n \text{ loglog } n)$ space and a penalty of $O(n)$ time per insertion and deletion.*

PROOF. Due to Lemma 3.1, the intersection of the two stabbing regions for sets A and B of line segments (that is the stabbing region for the union of A and B) can be found in linear time. The assertion is a consequence of this fact and a very general method for dynamizing similar structures due to Overmars [8] and Gowda, Kirkpatrick [5]. ■

4. Discussion.

In this paper, we present an algorithm which computes the stabbing region for n line segments in $O(n \log n)$ time. The stabbing region features a useful description of all stabbing lines and supports the related searching problem with a query time of $O(\log n)$.

The algorithm is a combination of three of the most prominent general paradigms for designing efficient algorithms. A geometric transform is used for obtaining an intuitively clear scenario, the stabbing region is computed using a prototype of the divide-and-conquer scheme, and the merge step is accomplished by the application of the plane sweep technique. Except for the divide-and-conquer heuristic, the paradigms are the recent outcome of increasing study of geometric problems. See Brown [1] for an excellent introduction to the effective use of geometric transforms, and see Shamos and Hoey [9] for the introduction of the plane sweep technique in computational geometry.

We consider our algorithm to be a good example of the successful treatment of a problem concerned with arbitrarily oriented line segments. However, recent results due to Edelsbrunner, Kirkpatrick, and Maurer [2] and Fredman [4] substantiate the thesis that a number of problems are considerably less complex in an orthogonal environment (i.e. with axis-parallel objects only) as opposed to a non-orthogonal environment (i.e. with objects of arbitrary direction).

In closing this section we point out several open problems that seem to bear close relationship to the one investigated in this paper. Again a set of line segments in the plane constitutes the environment for questions.

A general method of Edelsbrunner, Overmars, and Wood [3] can be used (i) to determine the maximal stabbing number, i.e. the maximal number of

segments that can be intersected by a single line, (ii) to find a line that does not intersect a single segment and separates the set into two nonempty subsets, (iii) to determine a direction such that the shadows of the segments do not intersect, and (iv) to determine a direction such that the view of the segments is connected. However, the $O(n^2 \log n)$ time requirements beg for more efficient solutions.

The generalizations of the stabbing problem and the open problems above to three and higher dimensions are also of interest. For example investigate the problem of finding a plane that intersects each of a given set of line segments in three dimensions.

REFERENCES

1. K. Q. Brown, *Geometric transforms for fast geometric algorithms*, Report CMU-CS-80-101, Department of Computer Science, Carnegie-Mellon University (1980).
2. H. Edelsbrunner, D. G. Kirkpatrick and H. A. Maurer, *Polygonal intersection searching*, Report F64, Institut für Informationsverarbeitung, Technical University of Graz (1981).
3. H. Edelsbrunner, M. H. Overmars and D. Wood, *Graphics in flatland: A case study*, Report F79, Institut für Informationsverarbeitung, Technical University of Graz (1981).
4. M. L. Fredman, *The inherent complexity of dynamic data structures which accommodate range queries*, Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science (1980), 191–199.
5. I. G. Gowda and D. G. Kirkpatrick, *Exploiting linear merging and extra storage in the maintenance of fully dynamic geometric data structures*, Proceedings of the 18th Annual Allerton Conference on Communication, Control, and Computing (1980), 1–10.
6. D. T. Lee, *Proximity and reachability in the plane*, Report R-831, Coordinated Science Laboratory, University of Illinois (1978).
7. D. T. Lee and F. P. Preparata, *Finding shortest paths with parallel segments as obstacles*, In preparation.
8. M. H. Overmars, *Dynamization of order decomposable set problems*, Journal of Algorithms 2 (1981), 245–260.
9. M. I. Shamos and D. Hoey, *Geometric intersection problems*, Proceedings of the 17th Annual IEEE Symposium on Foundations of Computer Science (1976), 208–215.