Maximal Segment Stabbing of Segments in the Plane

(Extended Abstract)

Bill Jones and Yan Ke
Department of Computational Science, University of Saskatchewan
Saskatoon, Saskatchewan S7N 0W0 Canada
{jones, ke}@skorpio.usask.ca

1 Introduction

Given a set of n (connected) objects in d dimensions, an object which intersects each of them is said to be a *stabber* or *transversal* of the set.

The general problem of finding a transversal admits an enormous number of specializations depending on the types of the participating objects, as well as on the form of answer desired: we may variously be asked to decide whether a transversal exists, construct a single example, or construct all possible transversals.

The problem's origin within the mathematical community [10, 12] was combinatorial in nature, deriving Helly–type theorems stating that a transversal exists for the whole set if one exists for every subset of size k. Early work on the problem by computational geometers considered cases where the set consisted of line segments and the stabber was an unbounded straight line. One of the first algorithms, by O'Rourke [15], constructs a stabbing line through (sorted) vertical segments in linear time. Edelsbrunner *et al.* [6] present an algorithm which computes a stabbing line for general segments in $O(n \log n)$ time; in fact, it computes a description of all stabbing lines, and for this latter task is optimal. Algorithms to compute line transversals for other planar objects have been obtained by Güting [11], Edelsbrunner [4], Atallah and Bajaj [1], and Egyed and Wenger [8]; transversals in higher dimensions have been considered by Avis and Doskas [2], Avis and Wenger [3], Edelsbrunner, Pach, Schwartz, and Sharir [7], and Jaromczyk and Kowaluk [13].

Less attention has been paid to generalizations of the stabber, or to variant types of stabbing where a complete transversal does not exist. We are aware only of Goodrich and Snoeyink's work [9] which uses a convex polygon to stab parallel segments in $O(n \log n)$ time, and of the result of Edelsbrunner and Guibas [5] where a line cutting the maximal number of planar segments can be found in $O(n^2)$ time. In this paper we consider the problem of finding a maximal stabbing of planar line segments by a *fixed–length segment*, and present three algorithms for successively more general variations. For reasons of brevity, descriptions are necessarily at a high level.

2 Algorithms

Let $S = \{s_1, \ldots, s_n\}$ be a set of line segments in the plane, and let the desired transversal t be a segment of fixed length $||t||$. Without loss of generality, in the sections which follow we assume that no two of the segments' endpoints have the same x-coordinate or y-coordinate.

2.1 Horizontal Stabber, Vertical Segments

In this section we consider a constrained version of the stabbing problem, where the stabber t is a horizontal line segment, and the segments in S are required to be vertically oriented.

Each segment s_i has y-extent from its bottom s_i^b to its top s_i^t and thus defines a y-interval $[s_i^b, s_i^t]$. Similarly, its x-coordinate s_i^x, when coupled with its translate at $s_i^x + ||t||$, defines an x-interval. The Cartesian product of these defines an isorectangle $[s_i^b, s_i^t] \times [s_i^x, s_i^x + ||t||]$: t stabs a segment s_i if and only if its right endpoint falls within this isorectangle. Our problem is thus equivalent to having n isorectangles, each of width $||t||$, and asking what is the region of maximum overlap. As with other problems related to isorectangles, plane sweep is a natural technique for solution. First sort the segments by their lower y-coordinate, and consider a horizontal line l which

sweeps the plane from bottom to top. As l sweeps up the plane, it will pass through $2n+1$ *slabs*, each of which is associated with the bottom or top endpoint of a segment s_i. Each slab is associated with a sequence $B = b_0, \ldots, b_k$ of stabbing numbers defined by the x-coordinates of segments (and their translates) active within the slab. Elements b_0 and b_k are always zero, and the others in the sequence differ from their predecessor by $+1$ or -1 depending on whether the segment traversed represented the left or right edge of its isorectangle.

A maximal-stabbing region for placing t then corresponds to the maximal element of B over all slabs (which need not be unique). A straightforward implementation would calculate the stabbing sequence in $O(n)$ time for each of the $O(n)$ slabs, and consume $O(n \log n) + O(n^2) = O(n^2)$ time. However, it is possible to do better by examining how that sequence varies between slabs. Adding a new segment when it is encountered transforms a sequence $b_0, \ldots, b_i, \ldots, b_j, \ldots, b_k$ into the sequence $b_0, \ldots, b_i, b_i+1, \ldots, b_j+1, b_j, \ldots, b_k$ where all the elements from b_i to b_j are temporarily incremented by 1 for the span of influence of that segment, and the original b_i and b_j are replicated alongside; deletion is just the reverse.

In general, the number of elements affected is $O(n)$, so we cannot afford to update each one individually. Instead, we store the evolving sequence in a balanced tree such as a 2–3 tree which supports updates in $O(\log n)$ time. Elements are indexed by the x-coordinate that defines their segment, and each node has associated with it two supplementary values: m, which records the maximal stabbing value of the subtree below the node; and δ, which provides the incrementing mechanism. As updates are performed on the tree, we maintain the invariant $m_{\text{parent}} = \max_{i \in \text{children}}\{m_i + \delta_i\}$ so that the m-value for the root represents the maximal stabbing number achieved in the entire tree. To add a new segment s_i, we first insert the pair s_i^x and its translate $s_i^x + ||t||$ in the normal fashion, and christen each with the b-, m-, and δ-values of its sibling towards the inside of the tree. Then by finding the common parent of these new nodes we isolate the subsequence which needs to be incremented; for each of the $O(\log n)$ nodes along the paths from the two new nodes to that common parent, we increment the δ-values of its siblings towards the inside by 1. (See Figure 1 for an illustration.) Finally, we can recompute the m-values for the tree by evaluating the invariant in bottom-up fashion from the new nodes up to the root. For deletions, the procedure is analogous but performed in reverse order to decrement the δ-values for siblings before removing the segment pair. Both types of update affect only $O(\log n)$ nodes, and can therefore be accomplished in $O(\log n)$ time, leading to an overall running time of $O(n \log n) + 2n \times O(\log n) = O(n \log n)$.

Theorem 1 *Given a set S of vertical segments in the plane and a horizontal stabbing segment t of fixed length, a maximally-stabbing position for t can be computed in time $O(n \log n)$.*

That this time is optimal can be seen by the following linear-time transformation from the element uniqueness problem, which requires $\Omega(n \log n)$ time. Given n real numbers $x_1, \ldots, x_n$ we create a (zero-length) y-interval for each x_i of $[x_i, x_i]$, and apply the horizontal-stabbing algorithm using a stabber of any length. All elements are distinct if and only if the reported maximal-stabbing number is 1.

2.2 Horizontal Stabber, Arbitary Segments

In this section we retain a horizontal stabber but relax the orientation constraint on the segments in S, allowing them to be in general position and perhaps intersecting.

Plane sweep remains a viable algorithmic method here; however, if segments are allowed to intersect with other segments or their translates, as shown in Figure 2, then the number of event points, or slabs, to consider grows to $O(n^2)$ along with the number of intersections.

We refine the algorithm of the previous section to work correctly for this problem as well. Within each slab, segments are nonintersecting as is desired, and the stabbing region is a trapezoid or triangle. We must be careful about handling a slab transition which arises from segment intersection: in this case the segment crossing from left to right, bottom to top (as seen when sweeping upward) must be temporarily removed and then reinserted on the right side of the segment it is crossing. Apart from similar modifications to deal with nonvertical segments, all other parts of the algorithm work as before, and the over-

all running time is $O(n^2 \log n) + O(n^2) \times O(\log n) = O(n^2 \log n)$.

Theorem 2 *Given a set S of arbitrarily oriented segments in the plane and a horizontal stabbing segment t of fixed length, a maximally-stabbing position for t can be computed in time $O(n^2 \log n)$.*

2.3 Arbitrary Stabber, Arbitary Segments

In this section we completely relax our constraint to allow an arbitrarily oriented stabber in addition to the arbitrarily oriented segments in S.

Here the arbitrary orientation of the stabber forces us to abandon plane sweep and adopt instead a configuration-space approach, as pioneered by Schwartz and Sharir [16] and later refined by Levin and Sharir [14]. We start by momentarily fixing an orientation θ for our stabber, and running a variant of the preceding algorithm, the result of which is a set of sequences of segments $\beta = \{\beta_1, \ldots, \beta_n\}$, each of size $O(n)$ and defining a companion sequence of stabbing numbers. Then, unless our original choice was unlucky, we can likely perturb the orientation θ slightly without causing a change in any sequence β_i; in fact, such changes will occur at only a finite number of *critical orientations* which partition the 180° of orientation space into an equally finite number of regions. If the sequence updates necessary when crossing a critical orientation can be computed efficiently, we have reason to hope that the resulting algorithm will be efficient overall. But first we must answer the questions of what defines a critical orientation, how many there are, and how we can update the segment sequences when crossing them.

The possible critical orientations for our problem are similar to those encountered by Levin and Sharir, and are illustrated in Figure 3; as for them, ours arise from combinatorial properties of the segments and their endpoints. A critical orientation of type (1) is created by pairs of segment endpoints, of which there are $O(n^2)$; type (2) comes from triples of segments, of which there are $O(n^3)$; and type (3) is essentially a degeneracy of type (2) where segments s_2 and s_3 have fused into one, something which can occur $O(n^2)$ times. The total number for us is thus $O(n^3)$, with type (2) predominating. (In contrast with the $O(n^2)$ bound of Levin and Sharir, our stabber may penetrate lines and rotates freely about the endpoint of s_2, whereas their ladder must avoid them and slides while retaining contact with s_2 and s_3.)

Ruthlessly omitting details of the precise sequence changes caused by types (1) through (3), we observe that in each case the change, in terms of the number of segments added or deleted, is always of constant size and affects only the two adjoining sequences, hence can be readily managed using the techniques developed earlier in this paper. Each transition update can thus be accomplished in $O(\log n)$ time. As hinted above, we first compute orientations and sort them into the orientation space, then treat each one in sequence by the incremental method just described, giving an overall running time for our algorithm of $O(n^3 \log n) + O(n^3) \times O(\log n) = O(n^3 \log n)$.

Theorem 3 *Given a set S of segments in the plane and a stabbing segment t of fixed length, a maximally-stabbing position for t, in arbitrary orientation, can be computed in time $O(n^3 \log n)$.*

3 Discussion

The time bound derived for the first algorithm is optimal; those for the second and third are tight insofar as they are derived from combinatorial bounds on the number of event positions. However, it is by no means true that all of them need be considered, and further time reductions are possible from properties of where the maximal stabber can appear. For instance, in the case of the second problem, it will be seen that when segments do not intersect the maximal stabber is always associated with, in critical orientation parlance, a type (1) position. This gives the promise of a better combinatorial bound on the number of positions to consider, leaving the challenge of how to exploit this property computationally: between successive type (1) positions there may lie $O(n)$ positions of type (2), and it is necessary to find a manner of determining their global effect on the stabbing sequences, rather than processing each one individually.

4 Acknowledgements

We thank Elina Pasheva for her helpful comments on an earlier draft of this paper. The support of the Natural Sciences and Engineering Research Council of Canada under grant OGP0093742 is gratefully acknowledged.

References

[1] M. J. Atallah and C. Bajaj. Efficient algorithms for common transversals. *Inform. Process. Lett.*, 25:87–91, 1987.

[2] D. Avis and M. Doskas. Algorithms for high–dimensional stabbing problems. Technical report, McGill Univ., TR SOCS 87.2, Montreal, Quebec, Canada, 1987.

[3] D. Avis and R. Wenger. Algorithms for line transversals in space. In *Proc. 3rd Ann. ACM Sympos. Comput. Geom.*, pages 300–307, 1987.

[4] H. Edelsbrunner. Finding transversals for sets of simple geometric figures. *Theoret. Comput. Sci.*, 35:55–69, 1985.

[5] H. Edelsbrunner and L. J. Guibas. Topologically sweeping an arrangement. In *Proc. 18th Ann. ACM Sympos. Theory Comput.*, pages 389–403, 1986.

[6] H. Edelsbrunner, H. A. Maurer, F. P. Preparata, A. L. Rosenberg, E. Welzl, and D. Wood. Stabbing line segments. *BIT*, 22:274–281, 1982.

[7] H. Edelsbrunner, J. Pach, J. T. Schwartz, and M. Sharir. On the lower envelope of bivariate functions and its applications. In *Proc. 28th Ann. IEEE Sympos. Found. Comput. Sci.*, pages 27–37, 1987.

[8] P. Egyed and R. Wenger. Stabbing pairwise–disjoint translates in linear time. In *Proc. 5th Ann. ACM Sympos. Comput. Geom.*, pages 364–369, 1989.

[9] M. T. Goodrich and J. Snoeyink. Stabbing parallel segments with a convex polygon. *Comput. Vision Graph. Image Process.*, 49:152–170, 1990.

[10] B. Grünbaum. On common transversals. *Arch. Math.*, 9:465–469, 1958.

[11] R. H. Güting. Stabbing c–oriented polygons. *Inform. Process. Lett.*, 16:35–40, 1983.

[12] H. Hadwiger, H. Debrunner, and V. Klee. *Combinatorial Geometry in the Plane.* Holt, Rinehart and Winston, 1964.

[13] J. W. Jaromczyk and M. Kowaluk. Skewed projections with an application to line stabbing in r^3. In *Proc. 4th Ann. ACM Sympos. Comput. Geom.*, pages 362–370, 1988.

[14] D. Leven and M. Sharir. An efficient and simple motion planning algorithm for a ladder moving in two-dimensional space amidst polygonal barriers. *J. Algorithms*, 8:192–215, 1987.

[15] J. O'Rourke. An on-line algorithm for fitting straight lines between data ranges. *Comm. ACM*, 24:574–578, 1981.

[16] J. T. Schwartz and M. Sharir. On the "piano movers" problem i: the case of a two-dimensional rigid polygonal body moving admidst polygonal barriers. *Comm. Pure Appl. Math.*, 36:345–398, 1983.

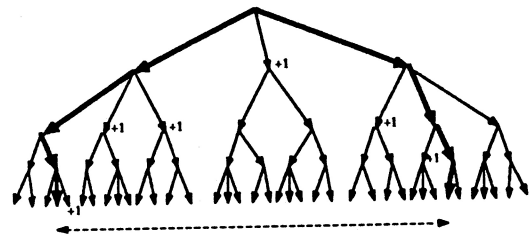

Figure 1: O(log n) nodes to increment

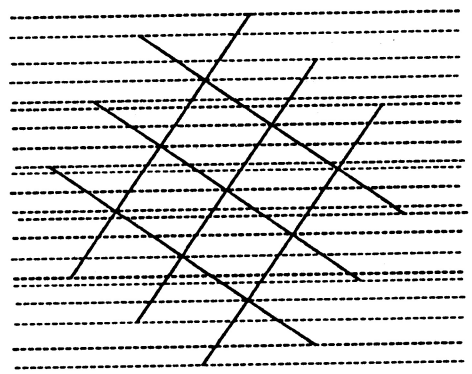

Figure 2: O(n*n) slabs from O(n) segments

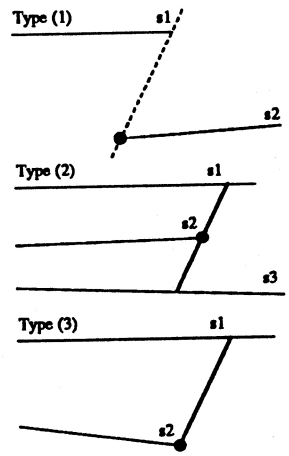

Figure 3: critical orientations for stabber