Topologically Sweeping the Complete Graph in Optimal Time

Eynat Rafalin [*] Diane Souvaine[*]

1 Introduction

We present a novel, simple and easily implementable[1] approach to sweep a complete graph of N vertices and k intersection points and report all intersections in optimal $O(k) = O(N^4)$ time and $O(N^2)$ space. Our method borrows the concept of horizon trees from the topological sweep method [6] and uses ideas from [9] to handle degeneracies. The novelty of the approach is the use of a *moving wall* that separates the graph into two regions at all times: the region in front of the wall that has known structure, and the region behind the wall that may contain intersections generated by edges that the sweep process has not yet predicted. This method has applications in computing the simplicial depth median of a point set in $\mathbb{R}^2$ [1]. Continuing research concentrates on modifying the topological sweep to work for arbitrary graphs. For some sparse graphs, where the total size of the cuts is limited, the proposed algorithm may be particularly effective.

A graph poses challenges for a sweep-line algorithm, that are not present in line arrangements, and that most existing sweep techniques do not handle. The structure holding the sweep-line status needs to be dynamic, as vertices and edges are constantly inserted and deleted. The event points now have two types and include both intersection points and graph vertices. Finally, *short* edges not yet encountered by the sweep line may create intersections that should be processed before intersections created by *long* edges that have already been detected (see Figure). Processing in the wrong order may introduce intersections not in the graph or ignore others, causing errors.

Existing algorithms for segment intersection detection do not work well for the complete graph. *Plane sweep* [3, 4] adds a $\log N$ factor to the optimal time complexity (yielding $O(N^4 \log N)$ time and $O(N^2)$ space). The optimal deterministic algorithm for the intersection reporting problem [5, 2] using $O(n \log n + k)$ time and $O(n)$ space, is dominated by $O(k) = O(N^4)$ when applied to a complete graph and is highly complex to implement. *Topological sweep* [6, 7, 9] offers a $\log n$ improvement factor over the vertical line sweep on an arrangement of n infinite lines, but does not handle finite line segments.

[*]Department of Computer Science, Tufts University, Medford, MA 02155. {erafalin, dls}@cs.tufts.edu

[1]On C++ code. Experimental results verify the time and space complexities.

2 Algorithm Overview

Let G be a planar embedding of a complete graph on N vertices. Assume vertices are in general position. The complete graph contains exactly $n = \frac{N(N-1)}{2}$ edges and k, the number of segment intersections, is $O(N^4)$. The number of edges cut by any sweep line is $O(N^2)$, but can be $\Theta(N^2)$ with $N/2$ vertices on each side of the line and $(\frac{N}{2})^2$ edges crossing it.

We sweep G from left to right to report all intersection points using a topological line (*cut*): a monotonic line in y-direction, intersecting each of the n edges *at most* once. The sequence of *active segments*, one per edge intersected by the topological line forms the *cut*. A segment of an edge is delimited by two adjacent intersection points or by the rightmost/leftmost intersection point and a vertex.

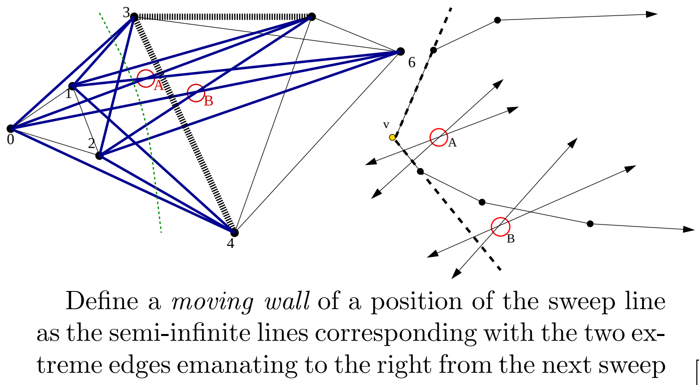
A sweep begins with the leftmost cut, which intersects no edges of the graph, and proceeds to the right in a series of elementary steps until it becomes the rightmost cut. An elementary step comprises the sweeping of the topological line past a vertex of the graph whose incoming edges are all currently intersected by the cut *or* past a *ready* intersection of edges that are consecutive in the current cut. There always exists a *ready* intersection or a *ready* vertex, whose incoming edges already lie on the cut, unless the cut is *rightmost* [10].

To maintain optimal time complexity linear in the size of the arrangement and space complexity linear in the maximal size of the cut, we build upon the concept of *horizon trees* [6]. Our *horizon graphs* are often not trees but *horizon forests*. The upper (resp. lower) *horizon forest* of the cut UHF (resp. LHF) is formed by extending the cut edges to the right. When two edges intersect, only the one of lower (resp. higher) slope continues to the right. Given LHF and UHF, the intersection of the right delimiters of the two forests produces the cut, which is computed from the updated LHF and UHF in constant time per active edge. A set of segments along the cut that contain the same intersection point as their right endpoint, generates a *ready intersection*, see [9, 6].

Intersection event points: The active segments switch position. To update UHF (resp. LHF) after processing the intersection point of segments $s_i, \ldots s_{i+k}$, for each segments s of the intersection apart from the first (last) one, traverse the *bay* formed by the segments above (below) s, until reaching the segment that intersects the extension of s (see [10, 6]).

Vertex event points: The sweep processes the vertices of the graph only when no intersection points are ready and in a left-to-right order, precisely when all the vertices' incoming edges are active: In a vertex event point all of the incoming active edges for vertex v are deleted from the set of active segments and all its outgoing edges are inserted. To update the horizon forests delete in-edges of v and insert its out-edges. Then update the horizon forest, walking in counterclockwise order around the bay formed by the previous edges to find the intersection point with an active edge. The edges emanating from the new vertex to the right, added to the set of active edges, may cut some of the existing active segments, and change the horizon forests and the cut. To update the horizon forests, test every existing forest edge for intersection with the new segment, in time linear in the size of the cut.

A moving wall: The set of active edges does not span the whole arrangement. Consequently, the sweep can encounter intersection points created by active edges before identifying intermediary edges that block them. For example, Figure (a) contains a graph of 7 vertices. The dotted sweep line produces the bold active segments. Intersection B of $\overline{06}$ and $\overline{25}$ is ready to be processed, as it is the right endpoint of the active segments associated with these edges. B, however, cannot be processed yet, as $\overline{06}$ and $\overline{25}$ intersect $\overline{34}$ before point B. Furthermore, if intersection B is processed, $\overline{06}$ and $\overline{25}$ will switch position in the cut, causing $\overline{34}$ to be inserted incorrectly. *Intersection points that are to the right of any edge that is not yet active cannot be* **ready**. Alternatively, consider intersection A. Given the dotted sweep line, both this intersection and vertex 3 are ready to be processed (in vertical sweep vertex 3 would be processed prior to A). However, the order of the cut edges currently places the edge $\overline{16}$ over edge $\overline{05}$. If vertex 3 is processed at this time, the update of the horizon forests and the cut will be 5 incorrect. *All intersection points that are to the left of all segments emanating from vertex v must be processed before v is processed.*



Define a *moving wall* of a position of the sweep line as the semi-infinite lines corresponding with the two extreme edges emanating to the right from the next sweep vertex v_x. The moving walls for all vertices can be computed in $O(N \log N)$ time using [8]. At any time, the sweep line has to be forced to the current position of the moving wall (by processing a ready vertex only if it is to the left of the line defining the wall). Provably, a ready intersection always exists unless the sweep line is aligned with the moving wall or has reached the rightmost cut. To align the sweep line with the cut ensure that no intersection that is inside the wall is ready and that ready intersections that affect the wall are swept before the next vertex is processed.

There may be intersections inside the moving wall associated with v_x that can still be *legally* swept (e.g. intersection B in Figure (b)). The sweep line is *forced* only to the wall formed by the two extreme segments emanating to the right from v_x and their extension to infinity instead of including all *legal* ready intersections, so intersections that are ready but inside the moving wall are discarded (Figure (b)). These intersections are rediscovered when v_x is swept, by checking every pair of adjacent active segments in the cut for ready intersections. This step is performed once for each vertex, with a cost linear in the size of the active set.

Acknowledgment The authors wish to thank Prof. Ileana Streinu, Michael A. Burr and Ryan Coleman.

References

[1] G. Aloupis, S. Langerman, M. Soss, and G. Toussaint. Algorithms for bivariate medians and a Fermat-Torricelli problem for lines. *Comp. Geom. Theory and Appl.*, 26(1):69–79, 2003.

[2] I. J. Balaban. An optimal algorithm for finding segment intersections. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 211–219, 1995.

[3] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28(9):643–647, 1979.

[4] K. Q. Brown. Comments on "Algorithms for reporting and counting geometric intersections". *IEEE Trans. Comput.*, C-30:147–148, 1981.

[5] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *J. ACM*, 39(1):1–54, 1992.

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

[7] H. Edelsbrunner and D. L. Souvaine. Computing median-of-squares regression lines and guided topological sweep. *J. Amer. Statist. Assoc.*, 85:115–119, 1990.

[8] F. P. Preparata. An optimal real-time algorithm for planar convex hulls. *Comm. ACM*, 22(7):402–405, 1979.

[9] E. Rafalin, D. Souvaine, and I. Streinu. Topological sweep in degenerate cases. In *Proc. 4th ALENEX*, volume 2409 of *LNCS*, pages 577–588. Springer-Verlag, 2002.

[10] E. Rafalin and D. L. Souvaine. Topologically sweeping the complete graph in optimal time and space. Tech. Report 2003-05, Tufts University.