Approximating the Shortest Path in Line Arrangements

David Hart *

Abstract

Suppose one has a line arrangement and one wants to find a shortest path from one point lying on a line of the arrangement to another such point. The best known time bound for computing this is $O(n^2)$. We develop an algorithm that finds a $1 + \epsilon$ approximation of the shortest path in time $O(n \log n + (\min\{n, \frac{1}{\epsilon^2}\})\frac{1}{\epsilon}\log\frac{1}{\epsilon})$.

1 Introduction

In this paper, we look at a problem that combines two frequently encountered constructs of computer science: shortest path problems and arrangements of lines or segments. The combined problem is that of finding shortest paths in arrangements.

That shortest path problems have wide applicability is well known. Consider the relationship of arrangements. Arguably, most shortest path problems of interest involve physical paths, and moreover, involve paths in the plane. There has been great interest in shortest paths in the plane, but almost all publications concern paths that go around obstacles and otherwise are unconstrained. There is a complementary problem: paths that are constrained to follow given routes. This is the better model for some real world problems, such as for shortest paths along roads. Yet there are very few published results.

For shortest paths along given routes, it is often the case that the allowed routes are lines. If not, one might also look at linear routes as a first approximation of curves. So, we might model these problems as shortest paths in an arrangement of segments. In the present paper, however, we look only at paths in arrangements of lines, for reasons explained below.

If one has an arrangement of lines or segments and has start and end points, one can compute a shortest path using two known algorithms as subroutines. First compute the arrangement, then apply a linear time shortest path algorithm for planar graphs. The time needed is $O(n^2)$.

This has been the best known time for the problem ever since the discovery, by Klein, et al. in 1994, of the linear time, planar, shortest path algorithm. Before this, using Dijkstra's algorithm for general graphs gave an additional $\log n$ factor. In fact, this decrease by a $\log n$ factor is the

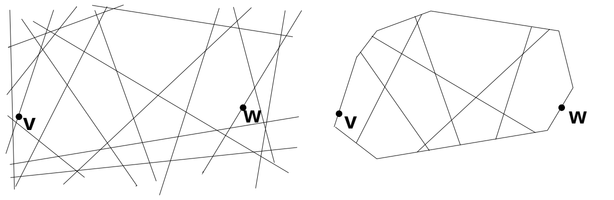

Figure 1: An example of an input line arrangement with path end points v and w. One can easily show that a shortest path never leaves the convex region shown. All segments crossing through the interior of the convex region are called *interior segments.*

only asymptotic time improvement on this problem in the 15 years since discovery (by Edelsbrunner, et al.) of an optimal algorithm for computing arrangements.

The paradox in this is the use of an algorithm for planar graphs. The variety of geometric knowledge, such as the triangle inequality, that one expects would give a faster algorithm goes unused, since it does not apply to all planar graphs. It may be easy to understand why computational geometers would be interested. Marc van Kreveld advertised the problem of shortest paths in line arrangements six years ago, and the problem has appeared on Jeff Erickson's web page through the present.

As mentioned, we here look at arrangements of lines as opposed to segments. Note that arrangements of segments are the more general problem, and an algorithm that runs in $o(n^2)$ for segments can also achieve this for line arrangements. If it is difficult to find improved algorithms for line arrangements, segment arrangements are still more difficult. Given the lack of progress on shortest paths in line arrangements, we confine our attention to this. Figure 1 illustrates the input to the problem.

In this paper we look at achieving a faster running time by approximating the shortest path.

1.1 Previous Results

The first (mathematical) result on exact shortest paths in a line arrangement is due to C. Davis [3] in 1948. He proves that a shortest path would not travel on certain segments of the arrangement.

As mentioned, the best known algorithm for the exact shortest path required $\theta(n^2)$ space and time in the worst

*Department of Information and Computer Science, University of California at Irvine, Irvine, CA 92697-3425. Email: dhart@ics.uci.edu

case. In 1998, Chen et al. [2] found a way to report the shortest path using $O(n^2)$ time but only $O(n)$ space.

David Eppstein and the present author looked at special cases of arrangements. We first looked at an arrangement that consists only of vertical and horizontal segments [4]. We showed that one can compute a shortest path in $O(n^{1.5})$ time and $O(n^{1.5})$ space. M. van Kreveld then improved this time to $O(n \log n)$. We next looked at line arrangements where we specify that the number of different line orientations is only k (that is, many of the lines are parallel). Our algorithm finds the shortest path in $O(n + k^2)$ time.

In the latter paper, we also discovered a theorem that applies to general line arrangements and makes it possible to considerably simplify the structure of the arrangement. The generality of the theorem was not used in that paper and was mentioned only in passing. We have need for the full generality of the theorem here.

The present paper looks at approximating the shortest path. The previous result on approximation is a paper by Bose, et al. [1], who describe an $O(n \log n)$ algorithm that finds an approximate path that is within a factor of 2 times the optimum.

1.2 New Results

In this paper, we describe an algorithm that finds a path within a $1 + \epsilon$ factor of the length of the shortest path in time

$$O(n \log n + (\min\{n, \frac{1}{\epsilon^2}\})\frac{1}{\epsilon} \log \frac{1}{\epsilon})$$

(Note that the previously known approximation algorithm has a fixed error factor of 2.)

In our approximation algorithm, we describe a new sub-algorithm that transforms the input arrangement into an arrangement of segments, where the new arrangement consists only of quadrilaterals contained in a convex, polygonal boundary. This new arrangement contains an exact shortest path of the original arrangement. The simple structure of the quadrilateral grid is the first step to making our approximation algorithm work. The proof uses a theorem, which we call the Sandwich Theorem, proven in part by Eppstein and Hart [5] and in part by Davis [3].

2 The Algorithm

2.1 Intuition

The key ideas of this algorithm are unintuitive, and some of what it does is quite difficult to picture (or to illustrate), despite the fact that we are working with lines in the plane.

The algorithm would be quite simple if not for a worst case scenario, where the input arrangement contains many lines that are very close to being parallel (but not exactly parallel, in which case we could deal with them easily). The positioning of the lines is also highly constrained for the worst case to occur.

When lines are very nearly parallel, it remains difficult to approximate the shortest path. The problem is that if one chooses the wrong lines to travel on, one can still introduce a large error into the approximation. We discuss this below.

Let V be the line through path starting point v; we will consider the angles made by lines of the arrangement with V. In the worst case, half of the lines intersect V below v making a small angle pointing up and half of the lines are above v, making a small angle pointing down. The reason we must have very small angles in the worst case is that the algorithm selects a subset of lines such that the angles with respect to V are getting small exponentially fast (specifically, the angles will be smaller than $\frac{\pi}{2}(1 - \epsilon/8)^m$, where m is the number of lines we have added to the subset). But in the worst case, the lines of the arrangement already have these exponentially decreasing angles, so our subset includes the entire arrangement!

Intuitively, we need to travel away from line V to reach w. If we are traveling on a line that makes an angle of 10^{-53} degrees with V and reach a line making an angle of 10^{-37}, this may seem like a much better line on which to travel. But by traveling a bit further on the first line, we may reach a line with angle 10^{-35}, and this is the actual shortest path. Determining which path will make a good approximation in the near-parallel lines remains difficult.

2.2 Algorithm: Key Steps and Ideas

Several steps of the algorithm are present solely to deal with the worst case arrangements. Note that the first three steps of the algorithm discard lines of the arrangement, and that none of these steps leads to any asymptotic time improvement. But these steps build structure into the remaining arrangement and we use this in later steps.

The algorithm proceeds as follows:

1. It almost trivial to show that a shortest path never leaves a convex region; we compute this region by taking the intersections of half planes bounded by certain lines of the arrangement. We discard any lines outside this region. For lines passing through the the convex region, we discard all but the segment inside it. See Figure 1.

   Definition 1 *An* interior segment *is any segment that crosses through the convex region described above. A* boundary segment *is any segment on the boundary of the convex region, including segments through v and w. The* reference lines *V and W are the lines through v and w.*

2. We now discard parts of interior segments, where the Sandwich Theorem (mentioned in the introduction) tells us that we can discard pieces of interior segments, knowing that a shortest path would never travel

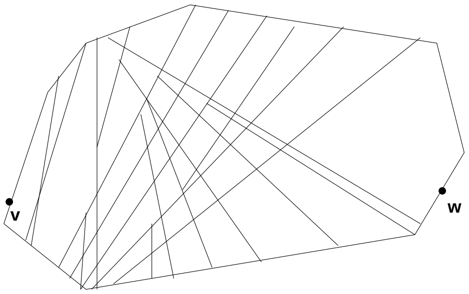

Figure 2: The intersection of two of the four sets of interior segments found by the algorithm. The two sets of interior segments produce only quadrilaterals in their intersection, but where the boundary interferes we have cells with other shapes.

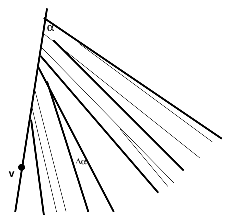

Figure 3: The subgrid includes segments (shown in bold) with angular spacing that is neither too large nor too small. (Boundary omitted for clarity.)

on those pieces. This procedure requires four passes through the entire set of interior segments. For each segment we may have two subsegments remaining in the arrangement (it is possible the entire segment is discarded). This process takes $O(n \log n)$ time.

After this step, the segment arrangement still contains an exact shortest path of the input arrangement. Note that if we attempt to find a shortest path in this segment arrangement, we still may use $\theta(n^2)$ time. What we have accomplished in this step is that the intersections of all remaining interior subsegments form only quadrilaterals. We thus have a grid of quadrilaterals contained in a convex boundary Additionally, the interior subsegments have an ordering by angle (we don't detail this here). See Figure 2.

We use both the simpler structure and the angular ordering in the next steps and in the proof of error bound.

3. The algorithm next finds a smaller subset of interior segments. It chooses interior segments with specified angular spacing relative to adjacent interior segments. The angles between adjacent segments of the new subset are not too large: this assures that the approximation will have a small error. The angles are also not too small: our method of choosing angles (a fraction of the angle with respect to line V through point v) leads to a recurrence, and this recurrence gives exponentially decreasing angles. See Figure 3.

   Again, we may have no asymptotic improvement in time. In fact, we may have exactly the same arrangement as before this step. What we gain in this step is a guarantee that the angles with respect to V are approaching 0 exponentially fast (before this step, we knew nothing about the angles). The case where we don't eliminate any segments is the worst case scenario that we discussed above.

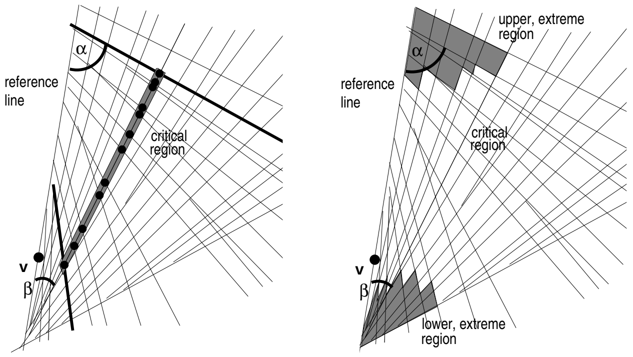

Figure 4: Illustration of the bounds of the critical region for one upward segment. The upper bold line marks where α becomes much larger than β. The lower bold line has α much smaller than β. We compute the distance to all intersection points marked by dots and none to points above or below. The second drawing shows the extreme regions in grey. If drawn to scale, they would be almost invisibly small. In the worst case arrangement, most segment intersection points are in these small regions. (Boundary omitted for clarity.)

4. We consider segments intersecting reference line V above v and making a small angle pointing down, and a similar set intersecting below and making small angles pointing up. For both sets, if we have many lines, we have angles that become extremely small. We scan each interior segment (say the ones intersecting V above v). As we scan down, the angles of intersecting segments go from very small to very large (relative to each other). We find an interval on the upward facing segment where the angles α of intersecting segments (measured against reference line V) are nearly equal to the angle β of the upward facing segment, starting with somewhat smaller angles (by a factor of ϵ) above and somewhat larger angles (a factor of $1/\epsilon$) below. See Figure 4.

   We find this interval for every interior segment. The combination of all these intervals is a contiguous strip through the grid of quadrilaterals. We call this strip the *critical region*. Approximating the shortest path is particularly difficult in this region, and we will compute a shortest path to every intersection point in the region.

Outside this region, we still have a grid of quadrilaterals, and in fact most intersection points of the grid are outside the region. We call this outer region the *extreme region*, for the extreme differences in angles (relative to the reference segment V) of intersecting segments. For this region, we can be lazy about shortest path computations—saving computation time. In particular, we do not compute shortest paths to each intersection point. We assume that a shortest path may travel through this region, but we don't allow it to make turns, so the path is a straight line from the boundary until it reaches the critical region, where it can begin to make turns. Thus, though we have many segments crossing other segments, our shortest path computation finds only the path length from the boundary to points on the fringe of the critical region. This is one distance computation per segment, as opposed to a huge number of computations if we found the path length to all intersection points.

Note that this is not simply a truncation of the grid. We do computations of paths through all parts of the grid; we just don't allow the path to make turns in the extreme regions.

An actual shortest path reaches the critical region on some last segment traveled in the extreme region. We show in the proof of correctness that, if the approximate path travels only on the boundary until it reaches this particular segment of the true shortest path, and it then travels on only that segment to the critical region (where we begin to do more careful path calculations), the error introduced cannot be very large.

5. There is an additional step of simply counting interior segments (in a particular order) and discarding the rest. This is possible because if the shortest path both stays trapped in the critical region (one case in the proof of correctness) and travels on too many segments, at some point it has oscillated up and down too many times to be a shortest path. Thus, we know that there is a limit to the number of interior segments we need in the arrangement, and (omitting details) we know which to discard.

6. The last step is to compute the shortest path. The path can travel on the convex boundary and on the grid of quadrilaterals. If it were not for the extreme regions, we would have a (very simple) planar graph. Because we have segments crossing other segments (in the extreme region) but don't treat intersections as vertices, this is not a planar graph. Nevertheless, the computation of shortest path is quite simple.

   From a boundary, we look at a straight path to a point on the fringe of the critical region, and we then compute lengths to vertices inside the critical region one by one (trivial in a grid of quadrilaterals). When we again reach the fringe of the critical region, we compute a straight line path to the boundary.

   A simple modification using a divide and conquer scheme reports the path using only $O(n)$ space.

3 Algorithm Running Time

It is not hard to see that the computation of path length takes $O(1)$ time for each vertex to which we compute the distance. To find the algorithm running time, we need to know the number of vertices in the shortest path tree.

The boundary has $O(n)$ vertices. Most vertices in the computation are vertices of the critical region. One can consider the thickness of the critical region to be the number of intersections, for any one segment, between the small-angle, intersecting line and the large-angle, intersecting line. These segments differ in angle by a factor of ϵ^2.

This is where the exponentially-fast change of angles is crucial. Because the angles change exponentially fast, we take the $\log$ base $(1-\epsilon/8)$ of ϵ^2. This turns out to be $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$. The width of the critical region can be as much as n but is truncated to be $\frac{1}{\epsilon^2}$. Multiplying these, and adding the $O(n\log n)$ time required in the first two steps of the algorithm, gives the previously stated running time.

Acknowledgements. The author thanks David Eppstein for numerous suggestions leading to improvements in this paper.

References

[1] P. Bose, W. Evans, D. Kirkpatrick, M. McAllister, and J. Snoeyink. Approximating shortest paths in arrangements of lines. In *Proc. 8th Canad. Conf. Computational Geometry*, pp. 143-148, 1996.

[2] D. Chen, O. Daescu, X. Hu, and J. Xu. Finding an optimal path without growing the tree. *6th Annual European Sym. on Algorithms*, Springer LNCS, pp. 356-367, 1998.

[3] C. Davis. The short-cut problem. *American Mathematical Monthly*, pp. 147-150, March, 1948.

[4] D. Eppstein and D. Hart. An efficient algorithm for shortest paths in vertical and horizontal segments. In *Proc. 5th Worksh. Algorithms and Data Structures*, pp. 234–247. Lect. Notes in Comp. Sci. 1272, Springer-Verlag, 1997.

[5] D. Eppstein and D. Hart. Shortest paths in an arrangement with k-orientations. *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms*, pp. 310-316, 1999.