

Approximating shortest paths in arrangements of lines

Prosenjit Bose William Evans David Kirkpatrick
Michael McAllister Jack Snoeyink
Department of Computer Science
University of British Columbia

1 Introduction

The problem of computing a shortest path between a pair of points has been extensively studied in many different settings, under many different metrics. In this paper, we study the problem of computing the shortest path between a pair of points on an arrangement of lines with respect to the Euclidean distance metric. Let $L = \{\ell_1, \ell_2, \dots, \ell_n\}$ be a set of n lines and let \mathcal{A} represent the arrangement induced by those lines. Given two points x, y on the arrangement, we want to find the shortest path between x and y such that each edge of the path lies on one of the lines in L .

One approach to solve this problem is to compute the arrangement \mathcal{A} and use Dijkstra's algorithm [2] to find the shortest path between x and y on this arrangement. The time complexity of this approach depends on the complexity of the arrangement of lines, which may be $\Theta(n^2)$. A straightforward reduction from the convex hull size verification problem [3] gives an $\Omega(n \log n)$ lower bound (for fixed order algebraic decision trees) on the time required even to verify that a given path is optimal. Therefore, the question arises as to whether one can compute the shortest path from x to y in $o(n^2)$ time, without actually computing the whole arrangement. This is a special case of a more general question concerning the identification of problems defined on arrangements that can be solved more efficiently than by reduction to the associated graph. This particular instance was posed by Marc van Kreveld at the Fourth Dagstuhl Seminar on Computational Geometry [1].

Computing the shortest path without computing the arrangement seems difficult; in particular, we have not succeeded in answering van Kreveld's question. However, we show how to compute, in $O(n \log n)$ time, a path P from x to y such that the length of P is at most twice that of the actual shortest path. Even though we have relaxed the optimality constraint, our algorithm makes extensive use of the structure of shortest paths on arrangements (or sub-arrangements formed by lines of restricted slope). As a simple example, a geometric fact exploited by our algorithm in several places is that the shortest path between x and y never crosses a line twice. In Section 2, we outline the properties of shortest paths on arrangements. In Section 3, we describe the approximation algorithm.

2 Properties of Shortest Paths

Before investigating approximate shortest paths from s to t , it is helpful to look at the structure of actual shortest paths. While the length of an approximate path can be compared to the distance between s and t , the actual shortest path in the arrangement can be arbitrarily longer.

Let \mathcal{A} be an arrangement of n lines and let s and t be points on lines of \mathcal{A} . We will often refer to the closed line segment \overline{st} ; these references do not require or imply that \overline{st} belongs to \mathcal{A} . Throughout this paper, we assume that \overline{st} is horizontal as shown in figure 1.

Definition 1 *The lines of \mathcal{A} that intersect the closed segment \overline{st} are called the cross lines of \mathcal{A} relative to s and t (figure 1b).*

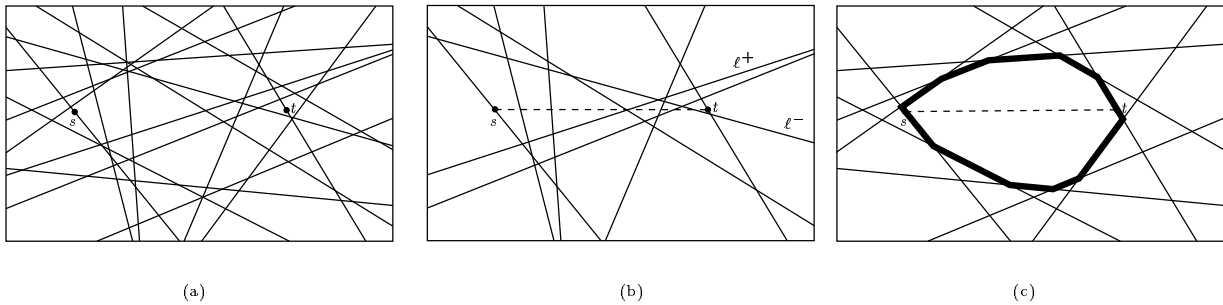


Figure 1: (a) An arrangement of lines, (b) cross lines with critical lines ℓ^+ and ℓ^- , and (c) cell $Cell(st)$.

Definition 2 *Let \mathcal{A}' be the arrangement \mathcal{A} without the cross lines. Include in \mathcal{A}' those cross lines that include s and t . The cell of s and t in \mathcal{A} , $Cell(st)$, is the boundary of the convex face of \mathcal{A}' that contains \overline{st} . A cell edge is an edge of the arrangement \mathcal{A} that lies on the boundary of $Cell(st)$. A cell vertex is an endpoint of a cell edge; it is an upper cell vertex if it is above the line \overline{st} and is a lower cell vertex if it is below the line \overline{st} ; vertices s and t are both upper and lower cell vertices. A cell line is a line of \mathcal{A} that contains a cell edge.*

The first property of shortest paths is a direct consequence of the triangle inequality. As stated the lemma applies to any path in the Euclidean plane. In particular, it applies to paths on any arrangement that includes the line ℓ .

Lemma 1 *Let P be a shortest path from s to t and let ℓ be any line. If P intersects ℓ at two points u and v then all points of P between u and v lie on ℓ .*

Proof: Let w be a point on P that lies between u and v . If w does not lie on ℓ then $|\overline{uw}| + |\overline{wv}| < |\overline{uv}|$ by the triangle inequality and we can find a shorter path than P from s to t by following ℓ , which contradicts the optimality of P . ■

Corollary 2 *If P is a shortest path from s to t in an arrangement \mathcal{A} then P is contained in $Cell(st)$.*

Proof: $Cell(st)$ is the intersection of halfspaces that contain both s and t . Any path that leaves such a halfspace can be shortened. ■

Define a total ordering \prec_U (resp. \prec_L) on the upper (resp. lower) cell vertices as $u \prec_U v$ (resp. $u \prec_L v$) if u occurs before v in the path from s to t along the upper (resp. lower) cell.

Corollary 3 *The order of vertices in a shortest path P from s to t agrees with \prec_U and \prec_L .*

Proof: Assume that $u \prec_U v$ but that P encounters v before u . Since P must remain within the face of $Cell(st)$ (Corollary 2), the portion of P from s to v separates u from t . The path P leads to t and cannot cross itself by Lemma 1, so P must leave v on the t side of the separation and does not visit vertex u , a contradiction. A similar argument applies to \prec_L . ■

Corollary 4 *Let \mathcal{A} be an arrangement of lines. Let k and m be two cross lines that intersect inside $Cell(st)$ with upper cell vertex u on k , upper cell vertex v on m , and $u \prec_U v$. Any shortest path starting at s that crosses $Cell(st)$ from one boundary to the other along k must start at u and descend to the lower boundary.*

Proof: Cross line m divides $Cell(st)$ into two regions: one containing s and u and the other containing t . Since m and k cross inside $Cell(st)$, the intersection points of k and $Cell(st)$ lie in separate regions. If a shortest path P uses k to traverse the cell from one boundary to another then it must start in the s side of m , namely at u . Otherwise, the path P would intersect m twice contradicting Lemma 1. ■

Our shortest path approximations will focus on two particular cross lines of extreme slope. Intuitively, these provide the fastest traversal across the face of $Cell(st)$.

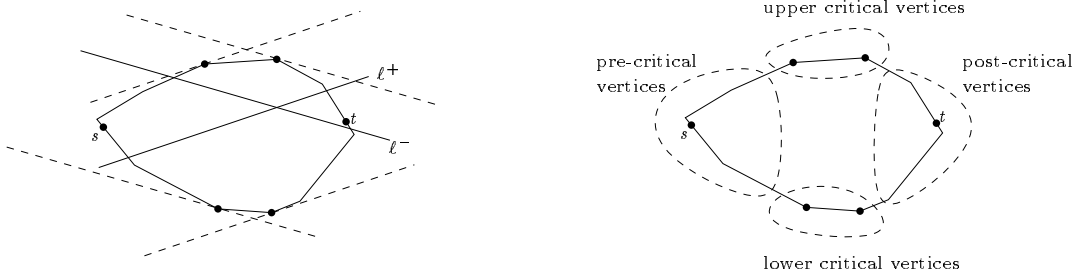


Figure 2: Division of $Cell(st)$ into pre-critical vertices, critical vertices, and post-critical vertices

Definition 3 *The extreme lines ℓ^- and ℓ^+ of \mathcal{A} for s and t are the cross lines with the largest negative slope and smallest positive slope respectively. If there is no cross line of negative slope then ℓ^- is the cross line with maximum positive slope. Similarly, if there is no cross line of positive slope then ℓ^+ is the cross line of with smallest negative slope.*

Since $Cell(st)$ is convex, the critical vertices are grouped into two contiguous chains and the cell $Cell(st)$ has supporting tangents parallel to ℓ^- and ℓ^+ at the ends of these chains (figure 2). By convention, point s is always pre-critical and point t is always post-critical.

Definition 4 *The cell vertices of $Cell(st)$ in the same chain as the point s are pre-critical vertices. The cell vertices in the same chain as t are post-critical vertices. The remaining vertices, including the vertices with the tangents parallel to ℓ^- and ℓ^+ , are critical vertices.*

Definition 5 *An edge of $Cell(st)$ is a critical edge if its two end vertices are critical vertices.*

Lemma 5 *The intersection vertex of extreme line ℓ^+ with the upper (lower) cell is not pre-critical (post-critical). The intersection vertex of ℓ^- with the upper (lower) cell is not post-critical (pre-critical).*

Proof: For every upper pre-critical vertex v , the line parallel to ℓ^+ through v does not intersect \overline{st} . The rest of the lemma follows from similar arguments. ■

We will want to consider shortest paths that only use some restricted set of lines from an arrangement. The restrictions are made by looking at *wedges*:

Definition 6 *Let k and m be distinct non-parallel lines, which divide the plane into four quadrants. The wedge defined by k , m , and a point r not on either k or m is the quadrant of the plane that contains the point r . The intersection point of k and m is the apex of the wedge. A line j respects a wedge W if the line parallel to j through the apex of W does not intersect the interior of W .*

The following pair of lemmas provide upper and lower bounds on the length of a shortest path that respects a wedge.

Lemma 6 *Let W be a wedge defined by lines k and m , and point r . Let p be the apex of W . Let P be the shortest path from r to p in an arrangement of lines that respect the wedge W . Then the length of P is greater than or equal to half the perimeter of the parallelogram with edges parallel to k and m and diagonal \overline{pr} .*

Proof: Proceed by induction on the number of edges in the path. Let R be the parallelogram with edges parallel to k and m and diagonal \overline{pr} .

Since all edges in P respect the wedge, P must have at least two edges. If P has exactly two edges then the path cannot intersect the interior of R and half the perimeter of R is a lower bound on the length of P .

Assume that all shortest paths of at most d edges from r to p have length greater than or equal to half the perimeter of R . Let P' be a shortest path from r to p with at most $d+1$ edges. If all vertices of P' lie outside of R then the result follows. Otherwise, let v be a vertex along P' inside R and divide P' into two paths: path P_1 from r to v and path P_2 from v to p . By the induction hypothesis, P_1 and P_2 have lengths greater than or equal to half the perimeter of the parallelograms with diagonals \overline{rv} and \overline{vp} respectively and edges parallel to k and m . If v is inside R then the sum of these perimeters is exactly the perimeter of R . Otherwise, the sum of these perimeters is greater than the perimeter of R . ■

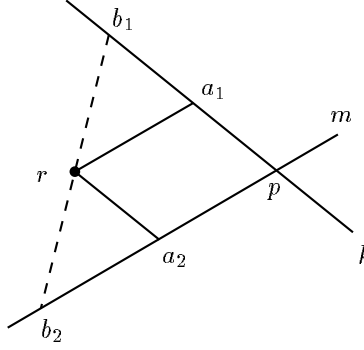


Figure 3: The length of a shortest path from r to p in the arrangement of lemma 7 is bounded by the perimeter of parallelogram ra_1pa_2 .

Lemma 7 *Let W be a wedge defined by lines k and m and point r . Let p denote the apex of W . If ℓ is any line through r , then the length of the shortest path from r to p in the arrangement formed by lines ℓ , k and m , is at most the perimeter of the parallelogram with edges parallel to k and m and diagonal \overline{rp} .*

Proof: Let a_1 (resp. a_2) denote the point on line k (resp. line m) intersected by the line parallel to m (resp. k) through point r . Let λ_1 (resp. λ_2) denote the length of segment $\overline{pa_1}$ (resp. $\overline{pa_2}$). Let b_1 (resp. b_2) denote the point on k (resp. m) at distance $\lambda_1 + \sqrt{\lambda_1\lambda_2}$ (resp. $\lambda_2 + \sqrt{\lambda_1\lambda_2}$) from p (figure 3). Then, b_1 , r and b_2 are colinear (by similar triangles). It follows that any line ℓ through p intersects either segment $\overline{rb_1}$ or segment $\overline{rb_2}$. Denote by z such an intersection point. By the triangle inequality, the path P_ℓ consisting of segments \overline{pz} and \overline{zr} has length at most $\min\{|\overline{pb_1}| + |\overline{b_1r}|, |\overline{pb_2}| + |\overline{b_2r}|\}$. But, $|\overline{pb_1}| + |\overline{b_1r}| \leq |\overline{pb_1}| + |\overline{b_1a_1}| + |\overline{a_1r}| = (\lambda_1 + \sqrt{\lambda_1\lambda_2}) + \sqrt{\lambda_1\lambda_2} + \lambda_2 = (\sqrt{\lambda_1} + \sqrt{\lambda_2})^2$. By a symmetric argument $|\overline{pb_2}| + |\overline{b_2r}| \leq (\sqrt{\lambda_1} + \sqrt{\lambda_2})^2$. Thus P_ℓ has length at most $(\sqrt{\lambda_1} + \sqrt{\lambda_2})^2$. But $(\sqrt{\lambda_1} + \sqrt{\lambda_2})^2$ is never greater than $2(\lambda_1 + \lambda_2)$, which is just the perimeter of the parallelogram with edges parallel to k and m and diagonal \overline{rp} . ■

It follows immediately from the preceding lemmas that the shortest path connecting a wedge point to its wedge apex using lines that respect the wedge can always be approximated to within a factor of two (which we abbreviate by *well-approximated*) by a path in a subarrangement of constant size. This observation, summarized in the lemma below, forms the building block for our more involved approximation algorithm.

Lemma 8 *Using the preceding notation, the shortest path from r to p in the arrangement formed by lines ℓ , k and m , well-approximates the path from r to p all of whose edges respect W .*

Lemma 9 *If P is a shortest path from s to t which does not contain a critical vertex then the shortest path from s to t on the arrangement of ℓ^+ , ℓ^- , and the lines which contain s and t well-approximates P .*

Proof: Let c be the intersection point of ℓ^+ and ℓ^- and let W_c^s (resp. W_c^t) denote the wedge formed by ℓ^+ and ℓ^- containing s (resp. t). Note that since ℓ^+ and ℓ^- both separate s and t , the wedges W_c^s and W_c^t are opposite quadrants and hence, any line that respects one respects the other.

It follows from Lemma 6 that a shortest path P from s to t that visits no critical vertex (and hence all its edges respect both W_c^s and W_c^t) can be assumed to lie entirely within the $W_c^s \cup W_c^t$ (and hence P must pass through c). Thus by Lemma 7, P can be well-approximated by the concatenation of the paths that well-approximate the subpath of P from s to c and the subpath of P from c to t . ■

Lemma 10 *Let P be a shortest path from s to t . Either P contains no critical vertices or P contains at least one critical vertex between the last pre-critical vertex a and the first post-critical vertex b in P .*

Proof: Assume that there is a shortest path P that contradicts the theorem; this implies, the subpath of P from s to a contains a critical vertex or the subpath from b to t contains a critical vertex or they both contain critical vertices. We may assume without loss of generality that the subpath from b to t contains a critical vertex (otherwise switch the roles of s and t). Let z be the first critical vertex in P after b . We may also assume that b is a lower cell vertex (otherwise reflect the arrangement about the line \overleftrightarrow{st}). Thus, z is on the upper cell (otherwise P would violate the lower cell ordering).

Since ℓ^+ is a cross line it separates s and t . From lemma 5, b lies on the same side of ℓ^+ as t . Once the path P crosses ℓ^+ from s to b , it does not re-cross ℓ^+ . Thus, the vertex z that follows b in P is on the same side of ℓ^+ as b and t .

Since ℓ^+ separates s from z and (by lemma 5) ℓ^+ does not intersect the upper pre-critical region, ℓ^+ intersects the upper critical region at a vertex y that precedes z in the upper cell ordering. Let x be the vertex where the path P and the line ℓ^+ first intersect.

If the path P does not visit a lower critical vertex between x and z then we can shorten P by following ℓ^+ from x to y and then following cell edges from y to z as we show in the next paragraph (see figure 4).

Let w be the last upper critical vertex between x and b in P (if no critical vertex lies between x and b then let $w = x$). Construct the wedge W_z with apex z , containing w , and with one bounding line ℓ being the critical cell line at z and the other bounding line parallel to ℓ^+ . Since the lines used by P between w and z are cross lines, they respect the wedge W_z . Thus, by lemma 6, the path, $Q(w, z)$, parallel to ℓ^+ from w to ℓ and then along ℓ to z is shorter than P 's path from w to z . If $w = x$, then w lies on ℓ^+ and the path $Q(w, z)$ exists in the arrangement. If $w \neq x$ then w is an upper cell vertex which may not lie on ℓ^+ . In this case, the path along the upper cell from w to z is no longer than $Q(w, z)$. This follows from the fact that the minimum perimeter convex set containing a set of points is the convex hull of those points (see for example [4]). In our case, $Q(w, z)$ and the segment \overline{wz} form a triangle which contains the path along the upper cell from w to z which, along with \overline{wz} , is the convex hull of the upper cell vertices from w to z .

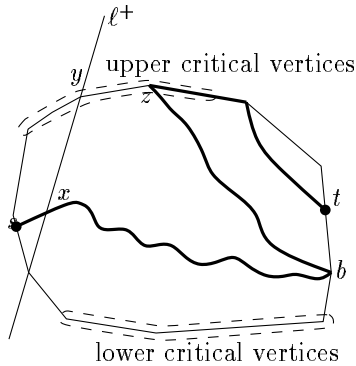


Figure 4: Illustration of the analysis of the structural properties of a shortest path.

Therefore, the path P must visit a lower critical vertex u between x and z . The path P cannot visit an upper precritical vertex after u because they are all on the other side of ℓ^+ from u and t . It cannot visit a lower pre-critical vertex after u since this would violate the lower cell ordering. ■

3 Algorithm

In this section, we describe an algorithm that takes a set of n lines and two points s and t that lie on the arrangement of these lines and determines a path on the arrangement from s to t that is at most twice as long as the shortest path on the arrangement from s to t . The running time of the algorithm is $O(n \log n)$.

The algorithm first determines some of the structural properties of the arrangement, and intuitively, orders the cell vertices “from s to t .” By considering cell vertices in this order, the algorithm builds up paths from s to z for each cell vertex z . By considering cell vertices in the opposite order, the algorithm finds paths from z to t . For each cell vertex z , the algorithm forms the path P_z that is the concatenation of the path from s to z and the path from z to t . Finally, the algorithm outputs the shorter of

- A. the shortest path from s to t using the cell lines at s and t and the extreme lines (a total of at most four lines).
- B. the shortest path from the set $\{P_z | z \text{ is a cell vertex}\}$

3.1 Cell Vertex Order

The first step of the algorithm is to determine the necessary structure in the arrangement of lines. The algorithm determines the cross lines (the lines that cross the segment \overline{st}) in $O(n)$ time. It determines $Cell(st)$ in $O(n \log n)$ time by intersecting the halfplanes (defined by the input lines)

that contain both s and t . It determines the cell vertices and the upper and lower cell orders (\prec_U and \prec_L) by intersecting the cross lines with $Cell(st)$ which takes $O(n \log n)$ time.

The algorithm imposes a “cross” partial ordering \preceq_C between cell vertices by considering cross lines \overleftarrow{uv} (here u is the upper cell vertex and v the lower cell vertex on the cross line). If there is no cross line \overleftarrow{wx} with $w \prec_U u$ and $v \prec_L x$ then $u \preceq_C v$. If there is no cross line \overleftarrow{wx} with $u \prec_U w$ and $x \prec_L v$ then $v \preceq_C u$. The motivation for this definition comes from Corollary 4: If $u \not\preceq_C v$ then a shortest path cannot cross the cell from u to v .

“Naive” methods to calculate \preceq_C would take time quadratic in the number of cross lines. However, \preceq_C can be determined in time linear in the number of cross lines by exploiting the following observation. The set of cross edges $\{\overline{uv} \mid u \text{ is an upper cell vertex, } v \text{ is a lower cell vertex, and } u \preceq_C v\}$ do not intersect within the cell. Thus the algorithm considers upper cell vertices in \prec_U -order. At any step, the algorithm has considered all upper cell vertices $w \prec_U u$ and knows the \prec_L -maximum vertex x reached by a cross line \overleftarrow{wx} with $w \prec_U u$. If cross line \overleftarrow{uv} has $x \prec_L v$ then $u \preceq_C v$ (and x becomes v). Repeating the similar process for the lower cell vertices, completes the construction of \preceq_C .

The algorithm directs the cell and cross edges in the arrangement consistent with the orders \prec_U , \prec_L , and \preceq_C . A cell or cross edge \overline{uv} is directed from u to v ($u \rightarrow v$) if $u \prec_U v$ or $u \prec_L v$ or $u \preceq_C v$. It is possible that a cross edge \overline{uv} is directed both from u to v and from v to u ($u \leftrightarrow v$). In this case, the algorithm creates an artificial copy u' of u with cell edge $u' \rightarrow u$ of length 0, and cross edges $u' \rightarrow v$ and $v \rightarrow u$. The algorithm creates at most a linear number of artificial vertices. The algorithm then calculates a total order \prec on cell vertices consistent with the edge directions (if $u \rightarrow v$ then $u \prec v$).

3.2 Incremental Approximation

The algorithm considers the cell vertices in \prec -order and for every cell vertex z , chooses the shortest of three paths from s to z . These paths are:

1. The directed cell edge $y \rightarrow z$, preceded by the path (that the algorithm has already calculated) from s to y .
2. The directed cross edge $x \rightarrow z$, preceded by the path (already calculated) from s to x . If z is not formed by the intersection of a cross line and a cell line (x does not exist) or $x \not\rightarrow z$ then this path is not considered.
3. The shortest path from s to z using the cell line at z , the cross line at z , and the cell line at s . This possibility is considered only when z is formed by a cross edge.

We say a cell vertex z is *well-approximated* if the length of the path assigned to it in the incremental approximation stage is at most twice the length of the path from s to z in any shortest path from s to t .

Lemma 11 *Let P be any shortest path from s to t . Each cell vertex in P up to (but not including) the first post-critical vertex is well-approximated.*

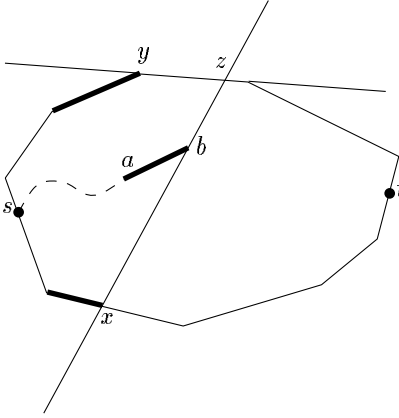


Figure 5: Analysis of the incremental approximation step.

Proof: The proof is by induction on the \prec -order of the cell vertices.

Let z be a cell vertex that occurs before the first post-critical vertex in P . If z has no adjacent cross edge then any path through z uses cell edge $y \rightarrow z$. By induction, y is well-approximated since $y \prec z$ and y precedes the first post-critical vertex. Thus z is well-approximated.

If z has an adjacent cross edge $x \rightarrow z$ then let W_z be the wedge formed by the cell and cross lines at z that contains s . Again, if P visits z via cell edge $y \rightarrow z$ or cross edge $x \rightarrow z$ then z is well-approximated by induction. Otherwise, consider the edges in P that precede z . We claim that these edges respect the wedge W_z . If they do not then consider the last edge \overline{ab} in P that does not respect the wedge.

The vertex b lies on one of the lines forming the wedge W_z , otherwise, by lemma 6 the path from b to z , which only uses edges that respect W_z , would be longer than the path from b to b' (the intersection of the line \overleftarrow{ab} with the boundary of W_z) to z .

If b lies on the cell line at z then P visits z via cell edge $y \rightarrow z$ which violates our assumption. So b lies on the cross line at z (see figure 5). Since $b \neq x$, \overline{ab} must be part of a cross edge. There are now two cases to consider depending on the line \overleftarrow{ab} . In the first case, \overleftarrow{ab} separates s and z from t . This is impossible since P must then re-cross \overleftarrow{ab} in going from z to t . In the second case, \overleftarrow{ab} separates s from z and t . In this case, since z is not a post-critical vertex, the line \overleftarrow{ab} respects the wedge W_z .

Since every line respects W_z and by lemma 8, the shortest path from s to z has length at most twice the length of the third path considered by the algorithm ■

3.3 Combining Incremental Approximations

The algorithm performs the incremental approximation from s to t and then from t to s . Notice that the “post-critical” vertices from the perspective of t are pre-critical from the perspective of s . Thus the incremental approximation algorithm well-approximates from t all vertices that follow the last pre-critical vertex in P .

For each cell vertex z , the algorithm concatenates the path from s to z with the path from z to t to obtain a path P_z from s to t through z . The algorithm outputs the shorter of

- A. the shortest path from s to t using the cell lines at s and t and the extreme lines (a total of at most four lines).
- B. the shortest path from the set $\{P_z | z \text{ is a cell vertex}\}$

Theorem 12 *The algorithm outputs a path which well-approximates any shortest path P from s to t .*

Proof: By Lemma 10, P either contains no critical vertex or visits a critical vertex after the last pre-critical vertex and before the first post-critical vertex. If P contains no critical vertex then the path describes in item A well-approximates P by lemma 9.

Otherwise, let z be a critical vertex after the last pre-critical vertex and before the first post-critical vertex in P . Since z precedes the first post-critical vertex, z is well-approximated from s . Since z follows the last pre-critical vertex, z is well-approximated from t . Thus P_z is at most twice the length of P , and the path output by the algorithm is at most as long as P_z . ■

References

- [1] H. Alt, B. Chazelle, and R. Seidel (editors). Computational geometry. Dagstuhl-Seminar-Report; 109, 1995.
- [2] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
- [3] D. G. Kirkpatrick and R. Seidel. The ultimate planar convex hull algorithm? *SIAM J. Comput.*, 15:287–299, 1986.
- [4] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1994.