Discrete Comput Geom 4:41-53 (1989)

Discrete & Computational Geometry

Rectilinear Shortest Paths in the Presence of Rectangular Barriers*

P. J. de Rezende,[1]† D. T. Lee,[2]‡ and Y. F. Wu[3]‡

[1] College of Computer Science, Northeastern University, Boston, MA 02115, USA

[2] Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208, USA

[3] Microelectronics and Computer Technology Corporation, CAD, 3500 West Balcones Center Drive, Austin, TX 78759, USA

Abstract. In this paper we address the following shortest-path problem. Given a point in the plane and n disjoint isothetic rectangles (barriers), we want to construct a shortest L_1 path (not crossing any of the barriers) from the source point to any given query point. A restricted version of this problem (where the source and destination points are known *a priori*) had been solved earlier in $O(n^2)$ time. Our approach consists of preprocessing the source point and the barriers to obtain a planar subdivision where a query point can be located and a shortest path connecting it to the source point quickly transvered. By showing that any such path is monotone in at least one of x or y directions, we are able to apply a plane sweep technique to divide the plane into $O(n)$ rectangular regions. This leads to an algorithm whose complexity is $O(n \log n)$ preprocessing time, $O(n)$ space, and $O(\log n + k)$ query time, where k is the number of turns on the reported path. If only the length of the path is sought, $O(\log n)$ query time suffices. Furthermore, we show an $\Omega(n \log n)$ time lower bound for the case where the source and destination points are known in advance, which implies the optimality of our algorithm in this case.

1. Introduction

Consider the following problem: given a set of obstacles (which may be line segments, circles, or polygonal shapes) and two distinguished points, s and t,

* A preliminary version of this paper appeared in the *Proceedings of the First Symposium on Computational Geometry* (1985).

† Supported in part by CNPq-Conselho Nacional de Desenvolvimento Científico e Tecnológico (Brazil).

‡ Supported in part by the National Science Foundation under Grants MCS 8420814 and ECS 8340031.

one wants to find a shortest path from s to t without crossing any of the obstacles [7]–[9], [13]. The path may, however, intersect the boundaries of the obstacles. In this paper we study the case in which the obstacles are disjoint isothetic rectangles and the distance between two points is measured in the L_1 metric.

Shamos [12] studied the shortest-path problem between two points lying inside a simple polygon. In 1974 Wangdahl *et al.* [15] presented a variation of Dijkstra's algorithm [4] for finding the minimal Euclidean distance between two points in the plane in the presence of polygonal barriers. Basically, their algorithm constructs the visibility graph of the set consisting of the source point, the destination point, and the vertices of the barriers and then finds the shortest path on this graph, where the weight of an edge is the Euclidean distance between its end points. A similar approach was undertaken by Lozano-Perez and Wesley [10] for the particular case in which the obstacles are convex polygons. (If G is a graph with positive weights associated to its edges, the shortest-path problem on G amounts to finding a path of the least total weight between two specified vertices [1].)

For the case where the barriers are line segments, the visibility graph G consists of the following. The vertex set of G is composed of the two distinguished points together with the $2n$ endpoints of the n given line segments. A pair of vertices is connected by an edge if and only if the line segment joining them does not intersect any of the barriers except possibly at the endpoints. Vaccaro [14] presented an algorithm to construct this graph in $O(n^3)$ time. Later, Lee [8] reduced this time bound to $O(n^2 \log n)$. Furthermore, he addressed the shortest-path problem with parallel line segments as barriers using the Euclidean metric and obtained an $O(n \log n)$-time algorithm which is optimal under the algebraic computation tree model defined by Ben-Or [3]. Lee's approach can be generalized to the case in which the line segments are not parallel provided that there exists a line onto which they have disjoint orthogonal projections (see also [9]). However, to determine such a line is not a simple task and it may not even exist. Welzl [16] and Asano *et al.* [2] have independently shown that the visibility graph G can be computed in $O(n^2)$ time.

Very different applications motivated each of these efforts: minimum trajectory pipe routing through a ship [15], routing urban vehicles [14], planning robot motion [10].

More recently, Larson and Li [7] studied a similar problem of finding all minimal distance paths among a set of origin–destination nodes in a network with polygonal barriers and rectilinear distance. They present an $O(k \cdot (k^2 + n^2))$-time algorithm where k is the number of origin–destination nodes and n is the total number of vertices of the barriers. Their result can be applied to solve a restricted version of the problem studied here. Namely, the case where both the source and destination points are known *a priori*. The complexity of their algorithm in this case is $O(n^2)$ time where n is the number of rectangular obstacles.

In addition to solving this case in $O(n \log n)$ time, the algorithm presented here actually solves the more general query version of this problem, namely, given a point in the plane and n disjoint isothetic rectangles (barriers), we want

to construct a shortest L_1 path (not crossing any of the barriers) from the source point to any given query point.

This paper is organized as follows. Section 2 presents some preliminary definitions. In Section 3 we show that *all* shortest paths starting at the source point are monotone in one of $-x$, $+x$, $-y$, or $+y$ directions. Next, in Section 4 an algorithm is presented and shown to achieve $O(n \log n)$ preprocessing time, $O(n)$ space, and $O(\log n + k)$ query time, where k is the number of turns on the path. In Section 5 we show an $\Omega(n \log n)$-time lower bound for the restricted version of a single destination point. Lastly, some concluding remarks and remaining open problems are mentioned in Section 6.

2. Definitions

Definition 1. Given points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ in the plane, the L_1 (or *rectilinear*) *distance* from p_1 to p_2 is defined as $d(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|$.

Definition 2. A simple polygon is called *isothetic* if all its sides are parallel to the coordinate axes. Given a set of isothetic polygonal obstacles and a point s in the plane, let P be a path starting at s and not crossing any of the obstacles. P is called *monotone in the x direction* (resp. y), referred to as an *x-path* (resp. y-path), if:

(a) the intersection of P with any line parallel to the y axis (resp. x axis is either a point or a line segment;
(b) for all $p = (p_x, p_y)$ on P, $p_x \geq s_x$ (resp. $p_y \geq s_y$).

To define monotonicity in the $-x$ direction (resp. $-y$), replace $\leq$ for $\geq$ in (b) above.

Definition 3. An xy-path P starting at a point s is called a *y-direction-preferred xy-path* (or simply *y-preferred xy-path*) if:

(a) P is the vertical line $x = s_x$ for $y \geq s_y$ (not crossing any obstacle); or
(b) P follows the $+y$ direction up to the boundary of an obstacle O, then follows the $+x$ direction up to the lower right corner p of O, then concatenates with a y-preferred xy-path starting at p.

Similarly, we define the following terms: *$(-x)$-path* as a path monotone in the $-x$ direction; *$(-y)$-path*; *$(-x)y$-path*; *$x(-y)$-path*; *$(-x)(-y)$-path*; *x-preferred*; *$(-x)$-preferred* and *$(-y)$-preferred.*

It follows immediately from Definition 3 that the y-preferred xy-path is uniquely determined by the location of the obstacles and of the point s. (Similarly, the following are unique: the χ-preferred $\chi\psi$-path and the ψ-preferred $\chi\psi$-path for $\chi \in \{+x, -x\}$, $\psi \in \{+y, -y\}$.)

The Rectilinear Shortest-Path Problem in the Presence of Rectangular Barriers. Given a point s in the plane and n disjoint isothetic rectangles (barriers),

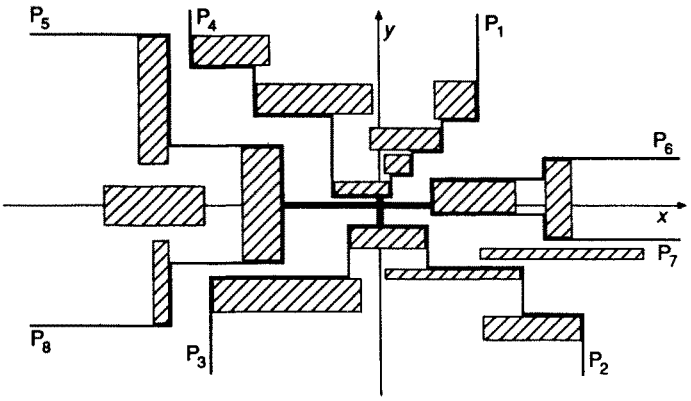

Fig. 1. The initial subdivision of the plane.

we want to construct a shortest L_1 path (not crossing any of the barriers) from the source point s to any given query point.

We will begin by dividing the plane into four rectilinear regions determined by the location of the point s and the barriers. Without loss of generality, regard s as the origin of the coordinate axes.

Definition 4. The region containing the $+x$ semiaxis and bounded by the y-preferred xy-path from s and the $(-y)$-preferred $x(-y)$-path from s is called the x-region. Similarly, we define the $(-x)$-region as the region containing the $-x$ semiaxis and bounded by the y-preferred $(-x)y$-path from s and the $(-y)$-preferred $(-x)$ $(-y)$-path from s.

Exchanging x and y above, we get the corresponding definitions for *y-region* and *$(-y)$-region.* Note that these four regions cover the entire plane and that the $\pm x$-regions always interest the $\pm y$-regions See Fig. 1.

Note that starting from the same point, the y-preferred xy-path P is always above or on the x-preferred xy-path P', since for P' to be above P there would have to be an intersection point of P and P' at which P' follows the $+y$ direction while P does not, but this would contradict the definition of P.

3. Monotonicity of the Shortest Paths

In this section we show that to each one of the regions defined above corresponds a direct $\delta \in \{-x, x, -y, y\}$ so that all shortest L_1 paths from s to any point in that region are monotone in the direction δ.

Given the symmetry of the four regions, all results will be shown without loss of generality just for the x-region.

From here on, consider all paths to be rectilinear. (This is clearly sufficient.) Given a path π and two points a and b along π we denote by π_{ab} the portion of π from a to b.

Lemma 1. *Every shortest path from s to a point in the x-region must lie entirely in that region.*

Proof. By contradiction. Let t be a point in the x-region. Let π be a shortest path from s to t and suppose that π goes outside the x-region. Since s and t are in the x-region, π must intersect its boundary at least twice. Let $\pi = (s, \ldots, p_1, p_2, \ldots, p_{i-1}, p_i, \ldots, t)$ where p_1 and p_i lie inside the x-region and $\pi_{p_2 p_{i-1}}$ lies outside. The line segment $\overline{p_1 p_2}$ intersects the boundary of the x-region at a point q_1 (possibly $q_1 = p_1$) and the line segment $\overline{p_{i-1} p_i}$ intersects the boundary of the x-region at a point q_2 (possibly $q_2 = p_i$). Let P_1 be the y-preferred xy-path from s and P_2 be the $(-y)$-preferred $x(-y)$-path from s. The boundary of the x-region consists of the union of P_1 and P_2. By the construction of P_1 and P_2 any vertical segment intersecting a horizontal portion of either of them must cross (or at least enter) an obstacle. Therefore, $\overline{p_1 p_2}$ and $\overline{p_{i-1} p_i}$ must be horizontal. Furthermore, $\overline{p_1 p_2}$ must point west and $\overline{p_{i-1} p_i}$ must point east. Hence π_{sq_2} is not an x-path. Assume without loss of generality that the segment $\overline{p_{i-1} p_i}$ intersects P_1. Let ρ be the path from s to q_2 along P_1. Since ρ is an xy-path while π_{sq_2} is not an x-path, we can construct a path shorter than π_{st} by concatenating ρ and $\pi_{q_2 t}$, contradicting the minimality of π. □

As a consequence of this lemma, note that given a point t in the intersection of two of the regions, all shortest paths from s to t must lie entirely in each of the regions and hence *in their intersection.*

Theorem 1. *Every shortest path from s to a point in the x-region must be an x-path.*

Proof. By induction on the number k of segments in the path. Without loss of generality, let t be a point in the x-region. Let π be a shortest path from s to t. Let P_1 be the y-preferred xy-path from s.

- If $k = 1$ or $k = 2$, π_{st} is certainly an x-path.
- Suppose the claim is true for all paths with less than k segments and suppose that π has k segments.
- Let qt be the last segment along π. If the point q is to the north, south, or west of t, by induction hypothesis and the previous lemma, π_{sq} is an x-path and therefore so is π_{st}. We now show that q cannot be to the east of t. Suppose otherwise. Starting at t, construct a y-preferred $(-x)y$-path ρ and a $(-y)$-preferred $(-x)(-y)$-path ρ'. It follows from t being in the x-region and q being to the east of t that π is intersected either by ρ or ρ'. Say ρ intersects π. (A symmetrical argument applies in the case of ρ' intersecting π). Let r be the intersection point of ρ and π. Since π_{rt} is not an x-path while ρ_{rt} is an xy-path, the latter must be shorter than the former. Hence, by concatenating π_{sr} and ρ_{rt} we obtain a shorter path than π from s to t, contradicting its optimality. □

It follows from the above theorem that given a point t in the intersection of, say, the x-region and the y-region, all shortest paths from s to t are xy-paths and therefore have length $d(s, t)$.

4. Construction of a Shortest Path

We now design the algorithm that actually constructs a shortest path. The approach begins by partitioning the plane, in the preprocessing phase, into the four regions of Definition 4. Each one of these regions in turn is divided into $O(n)$ (possibly semi-infinite) rectangular subregions. By precomputing shortest paths from the source point to the vertices of these subregions, a shortest-path query can be quickly answered by simply performing a point location.

4.1. *Constructing the Boundary of a Region*

Let s be a point in the plane and C be a collection of n disjoint isothetic rectangles. Let U be the set consisting of s, the vertices of all rectangles and the intersection points of the rectangles with the lines $x = s_x$ and $y = s_y$. Let X be the set of the x coordinates of all points in U and Y be the set of the y coordinates of all points in U.

The following algorithm describes a generic procedure to construct any of the eight paths that form the boundaries of the four relevant regions. Let $\chi \in \{+x, -x\}$ and $\psi \in \{+y, -y\}$. Refer to Fig. 1.

Algorithm *Boundary* (χ, ψ)

1. Sort X and Y.
2. To build a ψ-preferred $\chi\psi$-path start the path at s.
3. Move in the ψ direction until the path either encounters a χ-edge[1] of an obstacle or goes beyond the furthest obstacle (in the ψ direction). In the second case the path is extended indefinitely in the ψ direction and the construction is complete.
4. If a χ-edge of an obstacle is encountered, the path follows that edge in the χ direction up to its endpoint. Continue with step 3.

The correctness of this algorithm is immediate from the definition of the paths that it constructs. It follows from the finiteness of the number of obstacles that the algorithm always terminates.

The complexity analysis is equally simple. Step 1 takes $O(n \log n)$ time to compute and step 2 takes constant time. The loop of steps 3 and 4 can certainly be done in $O(n \log n)$ time.

4.2. *Subdivision of the Plane*

We apply the above algorithm to construct the boundaries of the four regions, $\pm x$ and $\pm y$, that divide the plane. We then proceed to further subdivide each of these regions into $O(n)$ rectangular subregions. These subregions will be used in the following way. Given a point t in the plane, we first determine in which

[1] A χ-edge is an edge parallel to the χ-axis.

of the $O(n)$ subregions t lies. This can be done in $O(\log n)$ time using an algorithm of Kirkpatrick [6], Edelsbrunner *et al.* [5], or Sarnak and Tarjan [11]. Denote this subregion by $R(t)$. It will be shown that there is a shortest path from s to t through one of the vertices of $R(t)$. The length of this path can be computed in additional constant time and the path can be traversed in additional $O(k)$ time where k is the number of turns in the path.

The subdivision of the four regions is essentially the same for each region and is constructed independently of each other. To avoid a cumbersome notation, we will describe the procedure specifically for the x-region.

Let V be the set consisting of the *pairs* of vertices of the left edges and the right edges of all obstacles contained in the x-region. The reason to consider the vertices in pairs is that the forthcoming scan process will analyze the vertical edges rather than the individual vertices. Note from the construction of the boundary of the x-region that a vertical edge of an obstacle is either completely inside or completely outside the x-region.

Remark 1. The vertex pairs in V will be lexicographically sorted by x and y coordinates. Let R and R' be two rectangles. If the x coordinate of the right edge R_r of R and of the left edge R'_ℓ of R' are equal, we will regard R_r as being to the left of R'_ℓ. This will guarantee that the vertex pairs appearing on the vertical scan line are either all from left edges or all from right edges. The same implied order applies to the lower and upper edges of rectangles with common y-coordinates.

Remark 2. Given a point p in the x-region, if π is a shortest path from s to p we denote by $D(p)$ the length of π and by $v_\pi(p)$ the last vertex before p along π, or simply $v(p)$ if the path is understood. Calculating and storing these quantities for an appropriate (finite) collection of points p will enable us to retrieve the length of a shortest path from s to any point in $O(1)$ time (after point location) or actually to traverse the path. (Actually, since we will be dealing only with the x-region, it would suffice to compute $D(p)$ as the sum of the lengths of the vertical segments of π since the sum of the horizontal ones is simply $|s_x - p_x|$ by Theorem 1.)

Remark 3. Let P_1 be the y-preferred xy-path from s and P_2 be the $(-y)$-preferred $x(-y)$-path from s. Let I be the set of vertical line segments of $P_1 \cup \{a_1\}$ and of $P_2 \cup \{a_k\}$, where a_1 and a_k are nominal endpoints of the paths P_1 and P_2. A set A of *active* (*vertical*) *line* segments will be manipulated by the algorithm. Generally speaking, A will contain vertical line segments with nonoverlapping y projections. The set A will be initialized to I. The segments in A will be maintained in sorted order by the y-coordinate. A balanced binary search tree may be used in an actual implementation. As we will see later, the following holds as the scan line sweeps the vertical edges of the obstacles in the $+x$ direction. The segments of A to the left of the scan line L are fully visible in the horizontal direction from L.

Algorithm *Subdivision*

1. Construct the set V and sort the vertex pairs according to Remark 1.
2. Initialize the set of active segments A to I as described in Remark 3. Sort the segments in A by y coordinates. Let $D(s)=0$, $v(s)=s$. For each vertex p of P_1 (resp. P_2), let $D(p)=d(s,p)$ and $v(p)=q$ where q is the vertex that immediately precedes p along P_1 (resp. P_2). Initialize the vertical scan line L on s_x.
3. Advance the scan line L to the next x value in V.
4. Let the set A be denoted $\{(a_1, a_2), (a_3, a_4), \ldots, (a_{k-1}, a_k)\}$ where the segments having nonoverlapping y-projections are listed in decreasing y-order. Refer to Fig. 2. For each vertex pair (p_1, p_2) of V on the scan line L $(p_{1y} > p_{2y})$, determine by binary search the two segments (a_i, a_{i+1}) and (a_{j-1}, a_j) in A such that $a_{iy} > p_{1y} \geq a_{i+1y}$ and $a_{j-1y} \geq p_{2y} > a_{jy}$. Let $D(p_1) = \min\{D(a_i)+d(a_i, p_1), D(a_{i+1})+d(a_{i+1}, p_1)\}$ and let $v(p_2)$ be the point a_i or a_{i+1} where this minimum is achieved. Similarly, let $D(p_2) = \min\{D(a_{j-1})+d(a_{j-1}, p_2), D(a_j)+d(a_j, p_2)\}$ and let $v(p_2)$ be the point a_{j-1} or a_j where this minimum is achieved. Let a and b denote the intersections of the lines $y=a_{iy}$ and $y=a_{jy}$, respectively, with L. Insert (a, p_1), (p_1, p_2), and (p_2, b) into A and delete (a_i, a_{i+1}), $(a_{i+2}, a_{i+3}), \ldots, (a_{j-1}, a_j)$ from A. Add to the subdivision the rectangular areas with left boundary (a_l, a_{l+1}) for $l=1$, $i+2, \ldots, j-1$ and right boundary at the current scan line.
5. If there are vertex pairs in V not yet scanned, continue with step 3. Otherwise, for each segment (a_l, a_{l+1}) in A, add to the subdivision the semi-infinite rectangular region with left boundary (a_l, a_{l+1}) and STOP.

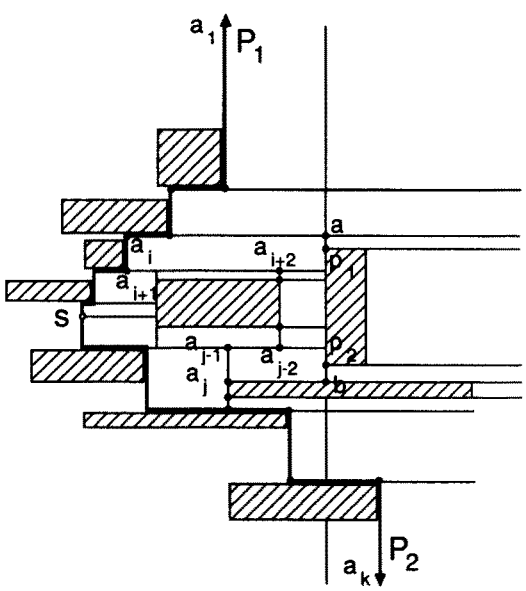

Fig. 2. The scan process.

Note that when the right edge of a rectangular obstacle is scanned in step 4, that rectangle becomes a subregion of the subdivision. In this way, by distinguishing these subregions we are able to determine whether a given query point is inside any of the obstacles (and therefore unreachable from s).

We say that a point q is x-visible from the scan line L if the straight line that extends from q in $+x$ direction does not intersect any obstacle before intersecting L.

Lemma 2. *At any given instance, the segments of A to the left of the scan line L are fully x-visible from L. Furthermore, if q_1 is a vertex (from some vertex pair) of V x-visible from L, then there must exist a point a in some vertex pair of A such that $a_y = q_{1y}$ and $a_x \geq q_{1x}$.*

Proof. All segments (in the x-region) to the left of the scan line L are eventually inserted into the set A when scanned by L in step 4. Whenever some portion of a currently active segment becomes no longer x-visible from L, that segment is removed from A. For the second part of the lemma, let (q_1, q_2) be a segment in A and suppose without loss of generality that q_1 remains x-visible from L while (q_1, q_2) is removed from A. Note that another vertex pair (a, r) is then inserted into A with $a_y = q_{1y}$ and $a_x \geq q_{1x}$. □

Lemma 3. *The total number of subregions in the subdivision is $O(n)$, where n is the number of obstacles.*

Proof. This follows from the fact that scanning a vertex pair in V can open at most three new subregions (possibly terminating others). □

Lemma 4. *The value $D(p)$ computed by the algorithm for each point p is equal to the length of the shortest path from s to p.*

Proof. The claim is clearly true for the turn points of the boundary of the x-region. Let p be a point of V encountered in step 4. Let (a_i, a_{i+1}) be the segment in A used to compute $D(p)$ (in step 4). First note that $a_{iy} > p_y \geq a_{i+1y}$ and that there is an xy-path $\pi_{a_{i+1}p}$ (hence of minimum length) from a_{i+1} to p and an $x(-y)$-path $\rho_{a_i p}$ (also of minimum length) from a_i to p. It suffices to show that there is a shortest path from s to p through either a_i or a_{i+1}. Let P be a shortest path from s to p. We will distinguish three cases:

1. a_{i+1} is above P;
2. a_i is below P;
3. a_i is above P and a_{i+1} is below P.

As before, denote by P_1 the y-preferred xy-path from s and consider each of the cases below.

1. Starting at a_{i+1} construct a $(-y)$-preferred $(-x)(-y)$-path Q. This path will either intersect P_1 or intersect P at some point q (possibly with $q = s$). In the first case, by concatenating P_{1sq}, $Q_{qa_{i+1}}$, and $\pi_{a_{i+1}p}$ we have an xy-path

from s to p through a_{i+1}. In the second case, we concatenate P_{sq}, $Q_{qa_{i+1}}$, and $P_{a_{i+1}p}$ obtaining a path from s to p through a_{i+1} which is no longer than P_{sp}.

2. Similar analysis as in case 1 for a y-preferred $(-x)y$-path from a_i.
3. *Claim.* The path P must intersect either the line l_i that extends from a_i in $(-x)$ direction or the line l_{i+1} that extends from a_{i+1} in $(-x)$ direction. Recall that the path P enters the rectangular region bounded by l_i, l_{i+1} and the line segment $\overline{a_i, a_{i+1}}$ through an intersection with $\overline{a_i, a_{i+1}}$. If s is above l_i or below l_{i+1}, the claim follows from Lemma 2. Suppose now that $a_{iy} > s_y > a_{i+1y}$. By Lemma 2 s must not be x-visible from L and, furthermore, s must be blocked (from L) by some obstacle which intersects both l_i and l_{i+1}.

   Suppose without loss of generality that the path P intersects l_{i+1} and let q be the intersection point closest to a_{i+1}. By concatenating P_{sq} with the line segment from q to a_{i+1} and $\pi_{a_{i+1}p}$ we obtain a shortest path from s to p through a_{i+1}.

This concludes the proof. □

Lemma 5. *The algorithm Subdivision takes $O(n \log n)$ time and linear space.*

Proof. Let us analyze the algorithm one step at a time:

1. The set V can be built in linear time and sorted in $O(n \log n)$ time.
2. The initialization of $D(p)$ and $v(p)$ for all turn points on the boundary of the region takes linear time.
3. Step 3 takes constant time and is executed $O(n)$ times.
4. In step 4 the only nonconstant time operations are:
   (i) two binary searches in A for each of the $O(n)$ points in V which accounts for $O(n \log n)$ total time;
   (ii) the insertion of points into A at a cost of $O(\log n)$ for each of the $O(n)$ points;
   (iii) the insertion of a total of $O(n)$ regions (Lemma 3) into the subdivision which takes at most $O(n \log n)$ total time.

   Therefore step 4 takes $O(n \log n)$ total time.
5. It is equally easy to see that step 5 can be performed in $O(n)$ time.

Clearly, no steps require more than linear space. □

4.3. Generating a Shortest Path

In summary, the following algorithm solves the rectilinear shortest-path problem in the presence of rectangular barriers.

Algorithm *Construction*

1. Apply the algorithm *Boundary* to divide the plane into the four regions $\pm x$ and $\pm y$.

2. Preprocess each of these regions independently by applying the algorithm *Subdivision* to them.
3. Given a query point t, determine which of the four primal regions contains t. If t lies in the intersection of two such regions, then the shortest path to t is a $(\pm x)(\pm y)$-path. Otherwise, suppose without loss of generality that t lies in the x-region. Locate t in the subdivision generated in step 2. Let $R(t)$ be the subregion containing t and let a_1 and a_2 be the endpoints of its left edge.
4. Let $D(t) = \min\{D(a_1) + d(a_1, t), D(a_2) + d(a_2, t)\}$ and let $v(t)$ be the endpoint where this minimum is achieved. Output $D(t)$ as the length of the shortest path from s to t.
5. If an actual path is to be reported, then:

```
output (t);
let p := t;
while p ≠ s do
        let p := v(p);
        output (p)
od
```

Theorem 2. *Let s be a point in the plane and C be a collection of n disjoint isothetic rectangles (barriers). After $O(n \log n)$ time for preprocessing, given a point t in the plane, the length of a shortest L_1 path from s to t (not crossing any of the barriers) can be determined in $O(\log n)$ time. Furthermore, an actual path can be reported in additional time proportional to the number of turns in that path.*

Proof.

- The correctness and complexity of steps 1 and 2 have been shown in previous lemmas.
- The location of a point in a planar subdivision of n regions can be done in $O(\log n)$ time (see [5], [6], and [11]).
- Arguments analogous to those of Lemma 4 show that the reported path is indeed a shortest path from s to t. □

5. Optimality of the Algorithm

Consider now the restricted case in which given *two* points s and t in the plane and n disjoint isothetic rectangles, a shortest L_1 path from s to t is to be constructed. This is referred to as the *Single Destination Rectilinear Shortest-Path Problem in the Presence of Rectangular Barriers.* By regarding this as a single query of the general case, we can apply the algorithm described in the previous section to solve this problem in $O(n \log n)$ total time. We will now show that this is optimal under the algebraic computational tree model (see [3]).

Theorem 3. *$\Omega(n \log n)$ time is a lower bound for the single destination rectilinear shortest-path problem in the presence of rectangular barriers.*

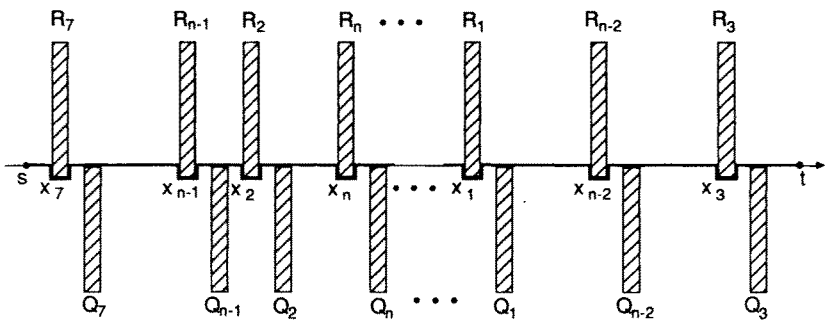

Fig. 3. Illustrating the $\Omega(n \log n)$ lower bound.

Proof. We show that integer sorting can be reduced to an appropriate instance of the problem. Given integers $x_1, x_2, \ldots, x_n$ to be sorted, let $s = (s_x, 0)$ and $t = (t_x, 0)$ be points such that s_x and t_x are integers respectively smaller and larger than the given n integers. These can be determined in linear time. For $i = 1, 2, \ldots, n$, let R_i be the isothetic rectangle with the southwest corner at $(x_i, -1/n)$ and the northeast corner at $(x_i + \frac{1}{4}, 2)$ and let Q_i be the isothetic rectangle with the southwest corner at $(x_i + \frac{1}{2}, -2)$ and the northeast corner at $(x_i + \frac{3}{4}, 0)$. Refer to Fig. 3. These are the barriers. Clearly, the length of any shortest path from s to t is $(t_x - s_x + 2)$ and any such path must turn at the southwest corners $(x_i, -1/n)$ of the barriers R_i's. Therefore, any shortest path (which must be monotone in the x-direction) sorts the given integers since it visits the points $(x_i, -1/n)$ in increasing order. □

Note that Theorem 3 holds provided that the actual path has to be produced. We conjecture that the same is true if just its length has to be reported.

6. Concluding Remarks

We have described an algorithm for finding a shortest L_1 path from a source point to any given query point not crossing any of n given isothetic rectangular barriers. The complexity of this algorithm is $O(n \log n)$ preprocessing time, $O(n)$ space, and $O(k + \log n)$ query time where k is the number of turns in the reported path.

We have also shown that this is essentially optimal for the case of a single destination point provided the actual path has to be produced.

A few natural generalizations of this problem are worth studying. One is the case in which the obstacles are (simple) polygons with sides parallel to the coordinate axes. Note that in this case, as in the one studied here, we can consider the paths to be restricted to the rectangular grid defined by s, t, and the vertices of the polygons. A second problem is to design an efficient algorithm when we are given a set of (more than two) origin-destination points. We may want to determine: (1) the shortest paths between every pair of points; or (2) a minimum

weighted rectilinear tree that interconnects the points; or (3) which of the points minimizes the sum of distances to the other points; or (4) for each point which other point has the shortest rectilinear path that avoids the obstacles. Note that (2) is the geometric Steiner tree problem and is known to be NP-hard. Hence, we look for good heuristics for it.

Acknowledgment

We would like to thank one of the referees whose suggestions greatly simplified the proof of Theorem 1.

References

1. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1984.
2. T. Asano, L. Guibas, J. Hershberger, and H. Imai, Visibility of disjoint polygons, *Algorithmica* **1** (198h6), 49-63.
3. M. Ben-Or, Lower bounds for algebraic computation trees, *Proceedings of the 15th ACM Annual Symposium on Theory of Computing*, 80-86, Boston, MA, 1983.
4. E. W. Dijkstra, A note on two problems in connection with graphs, *Numer. Math.* **1** (1959), 269-271.
5. H. Edelsbrunner, L. J. Guibas, and J. Stolfi, Optimal point location in a monotone subdivision, *SIAM J. Comput.* **15** (1986), 317-340.
6. D. G. Kirkpatrick, Optimal search in planar subdivision, *SIAM J. Comput.* **12** (1983), 28-35.
7. R. C. Larson and V. O. Li, Finding minimum rectilinear distance paths in the presence of barriers, *Networks* **11** (1981), 285-304.
8. D. T. Lee, Proximity and Reachability in the Plane, Ph.D.Thesis, University of Illinois, 1978.
9. D. T. Lee and F. P. Preparata, Euclidean shortest paths in the presence of rectilinear barriers, *Networks* **14** (1984), 393-410.
10. T. Lozano-Perez and M. A. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, *Comm. ACM* **22** (1979), 560-570.
11. N. Sarnak and R. Tarjan, Planar point location using persistent search trees, *Comm. ACM* **29** (1986), 669-679.
12. M. I. Shamos, Computational Geometry, Ph.D. Thesis, Yale University, New Haven, CT, 1978.
13. M. Sharir and A. Schorr, On shortest paths in polyhedral spaces, *SIAM J. Comput.* **15** (1986), 193-215.
14. H. Vaccaro, Alternative Techniques for Modeling Travel Distance, Thesis in Civil Engineering, Massachusetts Institute of Technology, 1974.
15. G. E. Wangdahl, S. M. Pollack, and J. B. Woodward, Minimum-trajectory pipe routing, *J. Ship Res.* **18** (1974), 46-49.
16. E. Welzl, Constructing the visibility graph for n line segments in $O(n^2)$ time, *Inform. Process. Lett.* **18** (1985), 167-171.

Received November 27, 1985, and in revised form November 3, 1986, and May 3, 1987.