# Euclidean Shortest Paths in the Presence of Rectilinear Barriers

**D. T. Lee**
*Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, Illinois 60201*

**F. P. Preparata**
*Coordinated Science Laboratory, Departments of Electrical Engineering and Computer Science, University of Illinois, Urbana, Illinois 61801*

In this paper we address the problem of constructing a Euclidean shortest path between two specified points (source, destination) in the plane, which avoids a given set of barriers. This problem had been solved earlier for polygonal obstacles with the aid of the visibility graph. This approach however, has an $\Omega(n^2)$ time lower bound, if $n$ is the total number of vertices of the obstacles. Our goal is to find interesting cases for which the solution can be obtained without the explicit construction of the entire visibility graph. The two cases are (i) the path must lie within an $n$-vertex simple polygon; (ii) the obstacles are $n$ disjoint and parallel line segments. In both instances greedy $O(n \log n)$ time algorithms can be developed which solve the problems by constructing the shortest-path tree from the source to all the vertices of the obstacles and to the destination.

## I. INTRODUCTION

The shortest-path problem in a graph $G = (V, E)$, where each edge is associated with a weight, consists of finding a path with the least total weight between two specified nodes in $V$. This is a celebrated problem in graph theory with significant applications, and there is an extensive literature on it. The classical solution of the single-source version of the problem when all edge weights are non-negative was discovered by Dijkstra [3]; it uses time $O(|V|^2)$ and has been shown to be essentially optimal [15] in the decision-tree model of computation.

More recently, interesting new applications have been described in the literature [7, 9, 12, 13, 17] which can be modeled as "constrained-path" problems, or as path problems with obstacles. Specifically, there is a set of polyhedral objects in space (polygons in the plane)—the "obstacles"—and the problem consists of finding a path (possibly, a shortest path in the applicable metric) between two distinguished points, a *source* and a *destination*. An early formulation of this problem, in the context of pipe routing, is due to Wangdahl, Pollock, and Woodward [17]. An analogous formulation was given by Lozano-Perez and Wesley [9] in connection with collision-avoidance path

NETWORKS, Vol. 14 (1984) 393–410

CCC 0028-3045/84/030393-18$04.00

problems in robotics. Both works are based on the so-called *visibility graph* (also called VGRAPH [9], viewability graph [14], and rediscovered several times in the literature), whose nodes are the vertices of the polyhedral obstacles and whose edges are the segments joining two nodes without intersecting the interior of any obstacle. It is interesting to note that, once the visibility graph is available and each edge is weighted with its length, Dijkstra's technique is applicable to the constrained problem. Larson and Li [7], in a paper that appeared after the original submission of this work, proposed a solution of the shortest-path-with-obstacles problem in the rectilinear ($L_1$) metric; their approach is based on the construction of a graph—basically analogous to the visibility graph in the Euclidean ($L_2$) metric—to which Dijkstra's algorithm is applicable. Related but more complex collision-avoidance path problems were studied by Reif [12] and Schwartz and Sharir [13].

A common feature of the solutions proposed in [7, 9, 17] is the use of the visibility graph. Unfortunately, the visibility graph on $n$ nodes has, in general, $\Omega(n^2)$ edges. So, even assuming that the visibility graph is available at no cost, $\Omega(n^2)$ time is a computational lower bound in this graph-theoretic approach.

The objective of this paper is to explore whether there are interesting special cases of constrained shortest-path problems which can be solved with $o(n^2)$ operations. We have considered the case in which the obstacles are $n$ line segments in the plane and distances refer to the Euclidean metric. It will be shown that $o(n^2)$-time algorithms exist in two significant instances:

P1. The given line segments form a simple polygon, to which $s$ and $t$ are internal.
P2. The given line segments are disjoint and parallel.

To be more specific, we shall show that problem P1 can be solved in $O(n \log n)$ time [and in $O(n)$ time if $O(n \log n)$-time preprocessing of the polygon is allowed], and that problem P2 can be solved in $O(n \log n)$ time; the latter is also shown to be optimal to within a constant factor. The reason for the higher efficiency is the nature of the problems, which allows us to obtain the solutions by constructing only a suitable subgraph of the visibility graph.

It must also be mentioned that recently Tompa [16] proposed an $O(n \log n)$ algorithm for solving a constrained-path problem arising in wire layout applications. Tompa's problem could be viewed as a special case of either P1 or P2, that is, the construction of the shortest path through a collection of parallel slots; his technique, however, is essentially different from those presented in this paper. This investigation reported in this paper sheds some light on the class of constrained shortest-path problems which admit of an efficient solution. The paper is organized as follows. In Section II, we discuss the problem of the shortest path within a simple polygon, and in Section III we consider the case in which the obstacles are parallel line segments. The general case (arbitrarily oriented line segments) and other open problems are briefly alluded to in the Concluding Remarks.

## II. SHORTEST PATHS WITHIN A POLYGON

In this section we consider the problem of finding a shortest path between two points $s$ and $t$ in the interior of the polygon, which does not cross the polygon boundary. We begin with some nomenclature.

**Definition 1.** A *polygonal chain* $\overline{q_1 q_2 \cdots q_k}$ is a sequence of points $q_i$ ($i = 1, 2, \ldots, k$) in which every pair of adjacent points $q_i$ and $q_{i+1}$ denotes a segment for $i = 1, \ldots, k-1$, and no two nonconsecutive segments intersect.

When the polygonal chain becomes a cycle we have:

**Definition 2.** An $n$-vertex *simple polygon* $P = (q_1, q_2, \ldots, q_n)$ is a polygonal chain $\overline{q_1 q_2 \cdots q_{n+1}}$ with $q_{n+1} = q_1$; i.e., $q_n$ and $q_1$ become adjacent. A diagonal of $P$ is a line segment $\overline{q_i q_j}$, $j \neq i+1$, which does not cross any edge of $P$. $P$ is said to be *triangulated* if its interior has been partitioned into $n-2$ triangles by $n-3$ diagonals.

Triangulated simple polygons have an interesting property. Referring to Figure 1, we may view the triangulated polygon as a planar graph $G$ embedded in the plane. The triangles of the triangulation are the interior faces of $G$. We may now construct the standard planar dual $G^D$ of $G$ [4], where each face of $G$ corresponds to a node of $G^D$ and each edge of $G^D$ connects two nodes if and only if their corresponding faces in $G$ share an edge. If we now remove from $G^D$ the node corresponding to the unbounded face of $G$, the dual graph $G^D$ becomes a tree whose vertices have degree at most 3. This is reflected in the following definition:

**Definition 3.** The *dual tree* of a triangulated simple polygon $P$ is a graph $T = (V, E)$ such that each node of $V$ corresponds to a triangle of the triangulation and each edge of $E$ connects two nodes of $V$ if and only if the corresponding two triangles share a diagonal of $P$. The diagonal of $P$ and the corresponding edge in $T$ are said to be *dual*.

Note that the triangulation of $P$ is entirely arbitrary; one suitable triangulation may be obtained—for example—in time $O(n \log n)$ using the algorithm of [5].

Our method is based on the following observation. Let $\Delta(s)$ and $\Delta(t)$ be the two triangles in (the triangulated) $P$ which contain $s$ and $t$, respectively. In $T$ there is a unique path $\pi$ between the vertices which are the duals of $\Delta(s)$ and $\Delta(t)$. The edges in $\pi$ are themselves duals of diagonals of $P$, and so the sequence of edges of $\pi$ corresponds to a sequence of diagonals $d_1, d_2, \ldots, d_p$ (ordered from $s$ to $t$). Since $d_i$ divides $P$ into two parts, which respectively contain $s$ and $t$, the shortest path from $s$

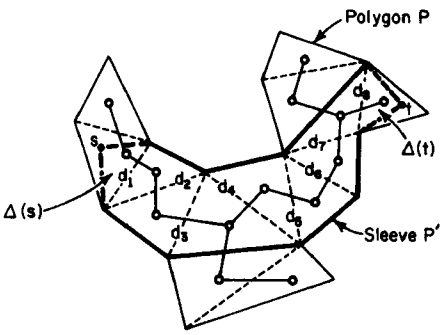


FIG. 1. A triangulated polygon and its dual tree.

to $t$ within $P$ crosses each and every $d_1, \ldots, d_p$ *exactly once*. This observation and a known property [9] of the visibility graph provide the basis for an efficient algorithm. This property is expressed by the following lemma. (For a proof, see [2].)

**Lemma 1.** Let $S$ be the set of endpoints of the rectilinear barriers. The vertices of the shortest path between $s$ and $t$ belong to the set $S \cup \{s, t\}$.

In our case $S$ is the vertex-set of $P$. Lemma 1 indicates that all we need to do is to construct, say, the shortest paths from $s$ to each of the endpoints of $d_1, d_2, \ldots, d_p$, and finally to $t$. The union of these paths is the usual *shortest-path tree with root s*. It is important to point out that, although the shortest-path tree with root $s$ is a subgraph of the visibility graph on $S \cup \{s, t\}$, we are not constructing the entire visibility graph [which may have $O(n^2)$ edges]. Indeed, since each path from the root $s$ crosses a diagonal exactly once, we can have a "greedy" algorithm, in which we sweep the triangles in the sequence specified by the path $\pi$, and *for each processed triangle we extend the shortest-path tree by one edge and one vertex.*

In addition, without loss of generality, we may restrict ourselves to the plane polygon $P'$ which dualizes to $\pi$, with the further condition that $s$ and $t$ be themselves vertices of the polygon [that is, we replace $\Delta(s)$ with the triangle having as its vertices $s$ and the extremes of $d_1$; similarly, $\Delta(t)$ is replaced by the triangle having as its vertices $t$ and the extremes of $d_p$]. This type of polygon is conveniently called a "sleeve" as in the following definition:

**Definition 4.** A triangulated polygon is called a *sleeve* if its dual tree is a chain.

In what follows we assume that the given polygon $P$ is a sleeve with $n$ vertices, including $s$ and $t$. The rest of this section is devoted to the description of the advancing mechanism, whereby the tree is extended edge by edge.

Let $v_i^{(1)}$ and $v_i^{(2)}$ be the two extreme points of diagonal $d_i$, $1 \leqslant i \leqslant n-3$, and let $D(s, v_i^{(j)})$ be the shortest path from $s$ to $v_i^{(j)}$, $j = 1, 2$, within the polygon $P$. From Lemma 1, $D(s, v_i^{(1)})$ is a polygonal chain whose points are vertices of $P$. Let $D_i = D(s, v_i^{(1)}) \cup D(s, v_i^{(2)})$. In the graph $D_i$ there is a unique vertex $v$ which is common to both $D(s, v_i^{(1)})$ and $D(s, v_i^{(2)})$ and is furthest from $s$ on either chain; we say that the two chains diverge at $v$ and, obviously, $D(v, v_i^{(1)})$ and $D(v, v_i^{(2)})$ have no edge in common.

Assume at first that neither of the latter subchains is empty; then we claim that $D(v, v_i^{(j)})$ $(j = 1, 2)$ is an *inward-convex* polygonal chain; i.e., it is convex with convexity facing toward the interior of $P$. To prove this, we first show that the region $R_i$ delimited by $D(v, v_i^{(1)})$, $D(v, v_i^{(2)})$, and $d_i$ (briefly called a *funnel*) is entirely contained in $P$. Let $d_s, d_{s+1}, \ldots, d_{i-1}$ be the diagonals crossed by $D(v, v_i^{(1)})$ and $D(v, v_i^{(2)})$. Clearly the triangle $(v, v_s^{(1)}, v_s^{(2)}) = R_s$ is contained in $P$; assuming inductively that $R_{i-1} \subset P$, we see that $R_i$ is obtained by adjoining to $R_{i-1}$ all or part of a triangle contained in $P$, thus showing that $R_i \subset P$. Next if $D(v, v_i^{(j)})$ is not inward-convex, then, by the triangle inequality, there is a shorter path from $v$ to $v_i^{(j)}$, entirely contained in $P$, thereby violating the hypothesis that $D(s, v_i^{(j)})$ is a shortest path from $s$ to $v_i^{(j)}$ (see Fig. 2). This convexity property also proves that $D(s, v_i^{(1)})$ and $D(s, v_i^{(2)})$ may diverge at most at one vertex $v$, for if they diverge at some other vertex $u_1$, then they must

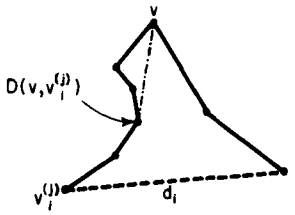


FIG. 2. Illustration of inward-convexity of $D(s, v_i^{(j)})$.

reconverge at some vertex $u_2$, and the two *distinct* subchains from $u_1$ and $u_2$ must both be inward-convex, a clear inconsistency.

In general $D_i$ is a (possibly empty) chain branching at some vertex $v$, called a *cusp* into two inward-convex chains, which delimit a (possibly degenerate) funnel. Note that either of these two chains could be empty (but not both, since $v_i^{(1)} \neq v_i^{(2)}$). If, say, $D(v, v_i^{(1)})$ is empty, then clearly $D(v, v_i^{(2)}) = d_i$; in this case the funnel degenerates to a single diagonal, $R_i$ has no interior, and $D_i$ becomes a single chain.

The algorithm successively constructs $D_1, D_2, \ldots, D_p$ and finally $D(s, t)$. In detail we have:

*Initial Step.* Construct $D_1$ by connecting $s$ to $v_1^{(1)}$ and $v_1^{(2)}$.

*General Step.* (Construct $D_{i+1}$ from $D_i$.) Let $v$ be the cusp of $D_i$, at which the two subchains $\overline{u_a u_{a+1} \cdots u_b}$ and $\overline{u_a u_{a-1} \cdots u_0}$ diverge, where $v = u_a$, $v_i^{(1)} = u_b$, $v_i^{(2)} = u_0$. Without loss of generality, let $v_i^{(1)} = v_{i+1}^{(1)}$ (see Fig. 3). Starting from $u_0$, scan the sequence $u_0, u_1, \ldots, u_b$ and let $j$ be the smallest integer for which $\overline{v_{i+1}^{(2)} u_j}$ becomes a supporting* segment of the boundary of $R_i$. We distinguish two cases

(i) $j \leqslant a$ [Fig. 3(a)]. Delete all edges $\overline{u_l u_{l+1}}$ for $0 \leqslant l \leqslant j-1$ and add edge $\overline{u_j v_{i+1}^{(2)}}$.

(ii) $j > a$ [Fig. 3(b)]. Delete all edges $\overline{u_l u_{l+1}}$ for $0 \leqslant l \leqslant j-1$ and add $\overline{u_j v_{i+1}^{(2)}}$; $u_j$ becomes a cusp of $R_{i+1}$.

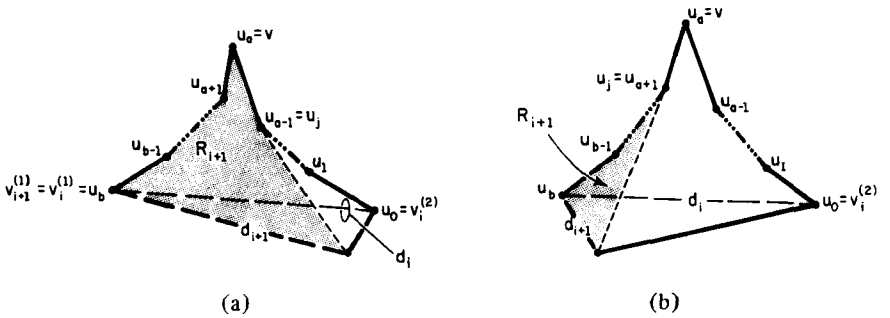


FIG. 3. Illustration of the general step. In (a), $u_j$ belongs to $\overline{u_s \cdots u_a}$; in (b) $u_j$ belongs to $\overline{u_a \cdots u_b}$. $R_{i+1}$ is shown crosshatched.

*A line $l$ is a supporting line of a convex open curve $C$ if it has at least one point in common with $C$ and $C$ lies all on one side of $l$, with its convexity facing $l$.

*Final Step.* Once $D_{n-3}$ has been constructed, one of the two sides of $P$ incident on $t$ is treated as a diagonal $d_{n-2}$ and the general step is applied to this case, yielding $D(s, t)$.

In Figure 4 we illustrate with an example the actions of the algorithm constructing the shortest-path tree from $s$. The diagonals are consecutively numbered, and for each $i = 1, \ldots, 13$, we indicate the cusp of $D_i$.

The correctness of the algorithm depends on the following claim: For any point $u$ in the triangle $R_{i+1}$ defined by two diagonals $d_i$ and $d_{i+1}$, a shortest path from $s$ to $u$ passes through $v$, the cusp of $D_i$. This is obviously true for $i = 1$, where $v = s$. Assume inductively that it is true for $1, \ldots, i-1$. In particular, the shortest paths from $s$ to $v_i^{(1)}$ and to $v_i^{(2)}$ pass through $v$. Consider the two subchains $D(v, v_i^{(1)})$ and $D(v, v_i^{(2)})$. If either of these two subchains is empty, then, as we saw earlier, the other subchain will consist of the diagonal $d_i$ with $v$ being a vertex of $d_i$. In this case the shortest path from $s$ to $u$ must be the concatenation of $D(s, v)$ and the segment $\overline{vu}$, and we are done. We therefore assume that both subchains are nonempty. Consider the edge incident on $v$ on either of these subchains: since $P$ is a sleeve, at least one of them is a diagonal of $P$ (although not necessarily an original diagonal of the triangulated $P$). Let $\overline{vv'}$ be this diagonal and $\overline{vv''}$ be the other edge (Fig. 5). Because of the inward convexity of these subchains, the unbounded wedge defined by $\overline{vv'}$ and $\overline{vv''}$ intersects $d_i$. Point $u$ in triangle $R_{i+1}$ belongs to one of the three regions in which the wedge partitions $R_{i+1}$ (refer to Fig. 5); all three cases, however, are treated analogously. Suppose now that the shortest path from $s$ to $u$, a polygonal chain $l(s, u)$, does not pass through $v$ but instead crosses $\overline{vv'}$ at some point $p \neq v$. Further assume that $l(s, u)$ crosses the ray containing $\overline{vv''}$ at a point $p_1$ (case shown in Fig. 5). By definition of the shortest path, the distance from $s$ to $p_1$ on $l(s, u)$ is shorter than the distance of the path obtained by concatenating $D(s, v)$ and the segment $\overline{vp_1}$, i.e.,

$$\text{length}(l(s, p)) + \text{length}(l(p, p_1)) < \text{length}(D(s, v)) + \text{length}(\overline{vp_1}),$$

where $l(p, p_1)$ denotes the subchain of $l(s, u)$ from $p$ to $p_1$. But, by the triangle inequality, $\text{length}(\overline{vp_1}) \leq \text{length}(\overline{vp}) + \text{length}(l(p, p_1))$, whence

$$\text{length}(D(s, v)) + \text{length}(\overline{vp}) - \text{length}(l(s, p)) > 0.$$

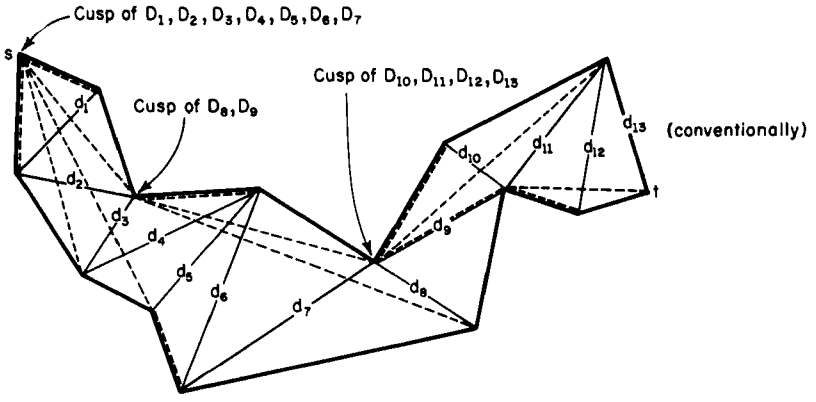


FIG. 4. Illustration of the shortest-path algorithm.

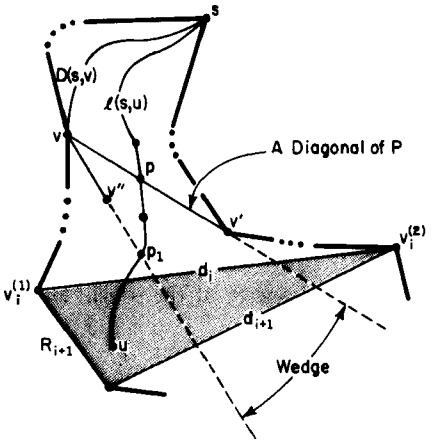


FIG. 5. Illustration for the proof that a shortest path between $s$ and $u$ passes through $v$.

Therefore length$(D(s, v))$ + length$(\overline{vv'})$ > length$(l(s, p))$ + length$(\overline{pv'})$, contradicting the known fact that the shortest path from $s$ to $v'$ passes through $v$.

We now analyze the running time of the algorithm. Case (i) of the general step takes constant time; case (ii) may involve scanning a large number of vertices; however, once a vertex has been scanned and the corresponding angle has been found to require continuation of the scanning process, that vertex is definitively eliminated from consideration. Since in $P$ there are $n - 2$ vertices besides $s$ and $t$, the entire algorithm runs in time $O(n)$. The shortest-path algorithm, however, assumes that $P$ be a sleeve. To transform an arbitrary simple $n$-vertex polygon into a sleeve, we first triangulate it in time $O(n \log n)$ [5], a suggested above; the dual $T$ of the given polygon is obtained in time $O(n)$ and, still in linear time, the path $\pi$ is obtained. This completes the transformation of the polygon into the required sleeve. Thus the entire procedure runs in time $O(n \log n)$, the triangulation task being asymptotically dominant. However, if preprocessing is allowed, the shortest-path problem can be solved in optimal $O(n)$ time for every pair of points $s$ and $t$. We summarize the results as a theorem below.

**Theorem 1.** Given a simple polygon $P$ with $n$ vertices and two points $s$ and $t$ in the interior of $P$, a shortest path between $s$ and $t$ lying entirely within $P$ can be found in $O(n \log n)$ time. If preprocessing of the polygon $P$ is allowed with preprocessing time $O(n \log n)$, then the problem can be solved in optimal $O(n)$ time for any two points $s$ and $t$ in the interior of $P$.

## III. SHORTEST PATHS WITH OBSTACLES REPRESENTED BY PARALLEL LINE SEGMENTS

In this section we consider the shortest-path problem in which the obstacles are $n$ disjoint and parallel line segments (assumed to be vertical, i.e., parallel to the $y$ axis, without loss of generality).

Denoting again by $S$ the set of $2n$ segment endpoints, by Lemma 1, the problem can be solved by constructing the shortest-path tree on $S \cup \{s, t\}$ with root in, say, $s$. As

for problem P1, our objective is to obtain this tree without constructing the entire visibility graph, of which it is a subgraph. To gain some intuition into how this might be attainable, we must establish some important properties of the shortest-path tree.

We begin with some nomenclature.

**Definition 5.** A polygonal chain $C = \overline{q_1 q_2 \cdots q_k}$ is said to be *monotone* with respect to a straight line $l$ if the perpendicular projections $l(q_1), l(q_2), \ldots, l(q_k)$ on $l$ of the vertices of $C$ are ordered on $l$ as $(q_1, \ldots, q_k)$.

Monotonicity is an important property of the sought shortest path, as shown in the following lemma.

**Lemma 2.** The shortest path SP between $s$ and $t$ is a polygonal chain monotone with respect to the $x$ axis.

*Proof.* From Lemma 1 we know that SP is a polygonal chain. Suppose for contradiction that SP is not monotone with respect to the $x$ axis; i.e., there are three consecutive vertices $v_{j-1}$, $v_j$, and $v_{j+1}$ of SP such that $x(v_{j-1}) < x(v_j)$ and $x(v_{j+1}) < x(v_j)$ (Fig. 6). This implies two facts: (i) there is at least one obstacle intersecting $\overline{v_{j-1} v_{j+1}}$, otherwise the latter segment would be on SP; (ii) none of the obstacles intersecting $\overline{v_{j-1} v_{j+1}}$ intersects $\overline{v_{j-1} v_j}$. Thus all obstacles intersecting $\overline{v_{j-1} v_{j+1}}$ have one endpoint inside the triangle $\Delta(v_{j-1} v_j v_{j+1})$. Let $A$ be the set of segment endpoints inside the triangle [clearly, by (i), $A \neq \phi$], and let $u_1$ be an element of $A$ such that the angle $\measuredangle(u_1 v_{j-1} v_j)$ is minimum. We now let $u$ be the point of the line containing $\overline{v_{j-1} u_1}$ such that $x(u) = x(v_{j+1})$. Clearly $(v_{j-1}, u)$ and $(u, v_{j+1})$ are both visible pairs and the length of chain $\overline{v_{j-1} u v_{j+1}}$ is smaller than that of $\overline{v_{j-1} v_j v_{j+1}}$, a contradiction. ∎

Lemma 2 can be rephrased by saying that the shortest path "never turns backwards." This holds, of course, for any pair of vertices $(s, v)$, where $v \in S \cup \{t\}$. Therefore, assuming without loss of generality that all points in $S \cup \{t\}$ have abscissae not smaller than $s$, we recognize that the shortest-path tree from $s$ to all points in $S$ with abscissa not exceeding a chosen value $x$ is *not influenced* by the segments lying to the right of $x$. This suggests that a left-to-right plane-sweep technique (see, e.g., [10]) is the natural approach to the solution of our problem. Specifically, we propose to scan the segments from left to right. If $\overline{pq}$ is the currently scanned segment, our algorithm will

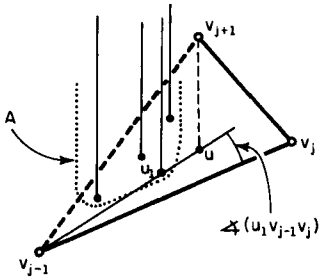


FIG. 6. Illustration for the proof of Lemma 2.

extend the shortest-path trees from $s$ to the points $p$ and $q$. We must now develop an efficient advancing mechanism.

The mechanism we propose is based on the notion of geometric *locus* which we now illustrate. Notationally, given a collection $\{l_1, l_2, \ldots, l_r\}$ of vertical segments, $p_{2i-1}$ and $p_{2i}$ denote respectively, the lower and upper endpoints of $l_i$. Thus, letting $p_0 \triangleq s$ and $S^* \triangleq \{p_0, p_1, \ldots, p_{2r}\}$, we seek a partition of the plane according to the following definition:

**Definition 6.** The *shortest-path map* SPM($S^*$) for a given set $S^*$ is a partition of the plane into regions $R(0), \ldots, R(2r)$. Region $R(i)$ $(i = 0, 1, \ldots, 2r)$ is associated with point $p_i \in S^*$ and is the locus of the points $q$ (visible from $p_i$) such that the shortest path from $s$ to $q$ passes by $p_i$ (i.e., the last edge of this shortest path is $\overline{p_i q}$).

The SPM has, obviously, the structure of a planar graph embedded in the plane. The use of the SPM in extending the shortest-path tree is now apparent. Suppose that point $s$ and segments $l_1, \ldots, l_r$ are all to the left of the current segment $l_{r+1} = \overline{p_{2r+1} p_{2r+2}}$ and the SPM($S^*$) is available. We may now "locate" $p_{2r+1}$ and $p_{2r+2}$ in SPM($S^*$), i.e., denote the regions to which each of them belongs; if $p_{2r+1} \in R(h)$ and $p_{2r+2} \in R(k)$, then the tree is extended by adding edges $\overline{p_h p_{2r+1}}$ and $\overline{p_k p_{2r+2}}$.

Before describing the algorithm in detail it is necessary to gain adequate insight into the structure of the SPM. The next subsection is devoted to this task.

### A. The Shortest-Path Map

We begin by considering the SPM when the segment set has just one member $l_1 = \overline{p_1 p_2}$, and $S^* = \{p_0, p_1, p_2\}$. The map is shown in Figure 7, where broken lines are used to represent edges of the shortest-path tree. With each $p \in S^*$ we associate a real number $W(p)$—the *weight* of $p$—which equals the length of the shortest path between $s$ and $p$. Note that for any point $q$ that lies on the curve separating regions $R(i)$ and $R(j)$ $(0 \leqslant i, j \leqslant 2)$ the lengths of the shortest paths from $s$ to $q$ via $p_i$ and via $p_j$ are

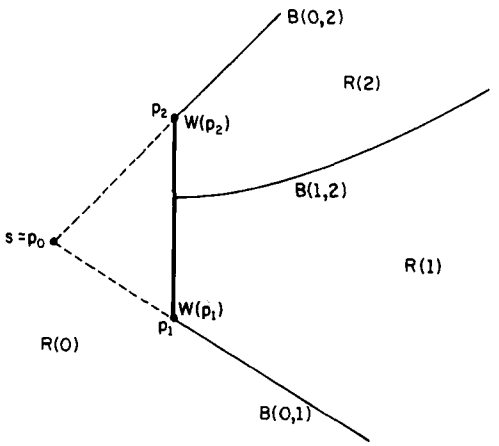


FIG. 7. Shortest-path map when the obstacle set consists of just one segment.

identical; that is, $d(q, p_i) + W(p_i) = d(q, p_j) + W(p_j)$. Therefore this curve, which is formally the geometric locus $\{q \mid d(q, p_i) + W(p_i) = d(q, p_j) + W(p_j)\}$, is called the *bisector* $B(i, j)$. Since the difference $d(q, p_i) - d(q, p_j) = W(p_j) - W(p_i)$ is a constant, the bisector $B(i, j)$ belongs to one of the two branches of the hyperbola with points $p_i$ and $p_j$ as foci and eccentricity $d(p_i, p_j)/|W(p_i) - W(p_j)|$. Specifically, $B(i, j)$ belongs to the branch closer to $p_i$ if $W(p_i) > W(p_j)$ or to the branch closer to $p_j$ otherwise. We also note (i) if $W(p_i) = W(p_j)$, then $B(i, j)$ becomes the perpendicular bisector of the line segment $\overline{p_i p_j}$. (ii) If $d(p_i, p_j) = |W(p_i) - W(p_j)|$, i.e., the eccentricity is equal to 1, then each branch of hyperbola degenerates to a half-line [see the bisectors $B(0, 1)$ and $B(0, 2)$ in Fig. 7].

We now establish some general properties of the SPM.

**Lemma 3.** For any point $z$ in region $R(i)$, the line segment $\overline{zp_i}$ lies entirely in $R(i)$.

*Proof.* Suppose, for a contradiction, that there exists a point $z'$ on $\overline{zp_i}$ which belongs to $R(j)$ with $j \neq i$. Then by the definition of the SPM, $d(z', p_j) + W(p_j) < d(z', p_i) + W(p_i)$. Since $d(z, p_j) \leq d(z, z') + d(z', p_j)$, we have $d(z, p_j) + W(p_j) \leq d(z, z') + d(z', p_j) + W(p_j) < d(z, z') + d(z', p_i) + W(p_i)$. Since $d(z, z') + d(z', p_i) = d(z, p_i)$, the above inequality implies that $d(z, p_j) + W(p_j) < d(z, p_i) + W(p_i)$, which contradicts the assumption that $z \in R(i)$. ∎

**Lemma 4.** In SPM($S^*$) each bisector is *monotone* with respect to the $x$ axis, i.e., any vertical line intersects a bisector at most once.

*Proof.* By contradiction. Suppose that there exists a vertical line $l$ which intersects a bisector, say $B(i, j)$, in two points $z_1$ and $z_2$ (refer to Fig. 8). From Lemma 3, for any $z \in B(i, j)$ segment $\overline{zp_j}$ lies entirely in $R(j)$, whence point $p_j$ must lie to the right of the vertical line $l$. But the shortest path from $s$ to, say, $z_1$ must pass through $p_j$, and is therefore not monotone with respect to the $x$ axis, contradicting Lemma 2. ∎

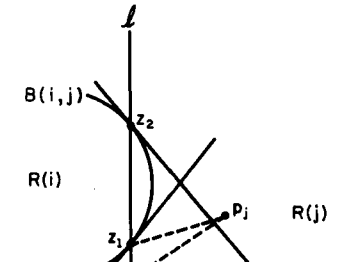


FIG. 8. Illustration of the proof of Lemma 4.

We define an *SPM-point* as a vertex of the SPM where bisectors meet. We prove the following property of SPM-points:

**Lemma 5.** Consider an SPM-point $q$ and the vertical line $l$ through $q$. There is one and only one bisector issuing from $q$ to the right of $l$.

*Proof.* Consider the counterclockwise sequence $(B_1, \ldots, B_s)$ of bisectors through $q$ to the left of $l$ (see Fig. 9) and let $B_j = B(j, j+1), j = 1, \ldots, s$. (Typically this bisector sequence consists of just two terms.) The region $R(j+1)$ to the left of $l$ between $B_j$ and $B_{j+1}$ $(j = 1, \ldots, s-2)$ is the locus of the points from which the shortest path passes through $p_{j+1}$. Similarly the region $R(1)$ above $B_1$ and the one $R(s+1)$ below $B_s$ are, respectively, the loci for $p_1$ and $p_{s+1}$. (Since each shortest path is monotone with respect to the $x$ axis by Lemma 3, $p_1, \ldots, p_{s+1}$ lie to the left of $l$.) If we now prolong each $B_j$ to the right of $l$, in the immediate proximity of $l$ we have $s+1$ regions. Of these, the top and bottom ones belong, respectively, to $R(1)$ and $R(s+1)$, while for any point in the intermediate ones the shortest path to it may pass through either $p_1$ or $p_{s+1}$. By continuity, any vertical line $l'$ to the immediate right of $l$ contains a point of $B(1, s+1)$, which is the only bisector issuing from $q$ to the right of $l$. ∎

We can now analyze the structure of the SPM viewed as a planar graph. Edges are curves joining two vertices and they are either portions of given vertical segments (obstacles) or portions of bisectors (note that several arcs of the same hyperbola may appear as edges of the SPM). Vertices of the SPM are basically of two types:

(i) Where bisectors meet. These vertices, previously called SPM-points, have—barring degeneracy—two bisectors incident from the left and one incident from the right. Their degree is at least 3.

(ii) Where bisectors meet a segment. Barring degeneracies, one bisector meets one segment. If the vertex is internal to the segment, then its degree is at least 3. If it coincides with the endpoint of a segment (and the bisector is rectilinear), then the vertex degree is 2.

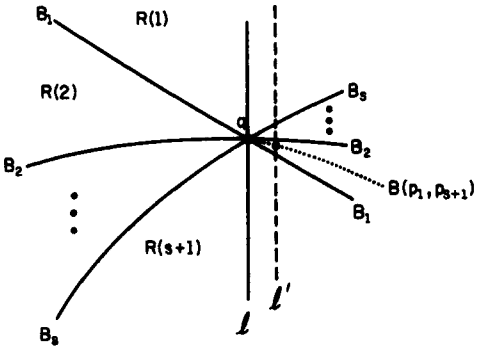


FIG. 9. Illustration for the proof of Lemma 5.

Let $f$, $e$, and $v$ be, respectively, the numbers of faces, edges, and vertices of SPM($S^*$), with $|S^*| = 2n + 1$. By Euler's theorem on planar graphs, we have

$$v - e + f = 2.$$

By its definition, SPM($S^*$) has $2n + 1$ regions. We have just seen that SPM($S^*$) has $2n$ vertices of degree 2: suppose we bypass each vertex, i.e., remove the vertex and replace its two incident edges with a single edge. The resulting graph $G'$ is characterized by numbers $f'$, $e'$, and $v'$ given by

$$f' = f = 2n + 1, \qquad e' = e - 4n + 2n = e - 2n, \qquad v' = v - 2n.$$

Each vertex of $G'$ has degree at least 3; therefore we have the straightforward relationship

$$2e' \geqslant 3v',$$

i.e., $v' \leqslant \frac{2}{3}e'$ or $e' \geqslant \frac{3}{2}v'$. As a consequence, applying Euler's formula to $G'$, we obtain

$$2 = v' - e' + f' \leqslant \tfrac{2}{3}e' - e' + f'$$

or

$$e' \leqslant 3f' - 6 = 6n - 3.$$

Analogously, one can show $v' \leqslant 4n - 2$, whence $e \leqslant 8n - 3$ and $v \leqslant 6n - 2$, i.e., both $f$ and $e$ are $O(n)$. Since SPM-points are a subset of the vertices of the SPM, and bisectors are a subset of the edges, we have proved

**Lemma 6.** The sets of the SPM-points and of the bisectors of an SPM on $n$ segments both have cardinalities $O(n)$.

### B. An Algorithm for Constructing the SPM and the Shortest-Path Tree

We have $n$ vertical segments $l_1, l_2, \ldots, l_n$, indexed according to the left-to-right order of their abscissae [this indexing can obviously be obtained in time $O(n \log n)$]. Segment endpoints are labeled according to the convention established above, with $p_0 = s$. After scanning the first $j$ segments $l_1, l_2, \ldots, l_j$, we have the shortest-path tree with root $p_0$ on the point set $S^* = \{p_0, p_1, \ldots, p_{2j}\}$. For a generic point $p$, we let $x(p)$ denote its abscissa. Finally, $x(p_i) \geqslant x(p_0)$ for $0 \leqslant i \leqslant 2n$.

The algorithm which constructs the SPM is a plane sweep. This technique is characterized by a line $l$ (sweep line) which sweeps the plane in one direction. It is applicable to those problems where the status of the geometric structure on the sweep line (at its current position) contains all the relevant information for continuing the sweep. This is our case, since both the shortest path (Lemma 2) and any bisector (Lemma 4) are monotone with respect to the $x$ axis. That is, by choosing line $l$ as vertical, and sweeping from $s$ to $t$ (left to right), the knowledge of the intersections of $l$ and the SPM is all

that is needed to proceed in the construction. In other words, the constructed portion of the SPM lying to the left of $l$ is final.

The status of the SPM on the sweep line is the sequence of the intersected bisectors (i.e., the ordered sequence of their intercepts on $l$). In the sweep this order can be modified only in two circumstances: either (i) when a vertical segment is reached, or (ii) when an SPM-point is reached. We call the abscissae of (i) and (ii) the *event points* of the sweep; if we order the event points from left to right, the order of the bisectors intercepted by the sweep line is unaltered between two successive event points. In conclusion, the proposed plane sweep needs two basic data structures:

(i) The *vertical status structure* $T$, which is a dictionary [1] of bisectors, arranged according to their vertical ordering from bottom to top. With each bisector $B$ in $T$ we associate an abscissa $X(B)$, denoting its rightmost extreme point [that is, the equation of $B$ holds only for $x \leqslant X(B)$].

(ii) The *future event structure* $Q$, which is a priority queue [1] of abscissae. Each abscissa in $Q$ is associated with a record which has one of two forms: either (a) $(x; y_1, y_2)$, if $x$ is the abscissa of an obstacle segment whose lower and upper endpoints have respective ordinates $y_1$ and $y_2$, or (b) $(x; y, B', B'')$ if $x$ is the abscissa of an SPM-point $(x, y)$ which is the intersection of bisectors $B'$ and $B''$.

Both $T$ and $Q$ may be implemented as height-balanced trees and support each of the necessary operations in time logarithmic in their sizes.

Data structure $T$ is initialized as empty, while $Q$ is initialized with the abscissae of the vertical segments $l_1, \ldots, l_n$. Figure 7–and its discussion–could be viewed as describing the *initial step* of the plane-sweep. The *general step* of the algorithm moves the sweep line from the current event point to the next event point, and consists of obtaining MIN($Q$) (the next event point), extending the shortest-path tree and updating the SPM (finalizing it to the left of the sweep line). We next describe this general step and accompany the description with an example illustrating the various situations.

In Figure 10 the sweep line is positioned at segment $l_3$, whose processing has just been completed. (As an exercise, the reader is advised to apply the general step to verify the construction of the current partial SPM displayed in Fig. 10.)

*General Step.* MIN($Q$) is extracted from the queue $Q$. We distinguish whether MIN($Q$) is the abscissa of a segment $l_i$ or of an SPM-point. [*Comment:* in Fig. 11, MIN($Q$) is the abscissa of $l_4$.]

(a) *MIN($Q$) is the abscissa of a segment* $l_i = \overline{p_{2i-1}p_{2i}}$.

(a1) *Extend shortest-path tree.* By searches in $T$, locate $p_{2i-1}$ in some region $R(j)$ and $p_{2i}$ in some region $R(k)$. The shortest-path tree is extended by linking $p_{2i-1}$ to $p_j$ and $p_{2i}$ to $p_k$. [*Comment:* in Fig. 11 $p_7$ is in $R(5)$ and $p_8$ in $R(3)$.]

(a2) *Construct bisectors.* Generate the equations of bisectors $B(j, 2i-1)$, $B(2i-1, 2i)$, and $B(2i, k)$. Set $X(B(j, 2i-1)) = X(B(2i-1, 2i)) = X(B(2i, k)) = \infty$. [*Comment:* In Fig. 11 the new bisectors are shown with heavy lines. Note that only $B(2i-1, 2i)$ is, normally, a branch of non-degenerate hyperbola.]

(a3) *Update T.* Insert $B(j, 2i-1)$ into $T$, delete all bisectors whose intersection with $l$ have ordinates comprised in those of $p_{2i-1}$ and $p_{2i}$, and finally

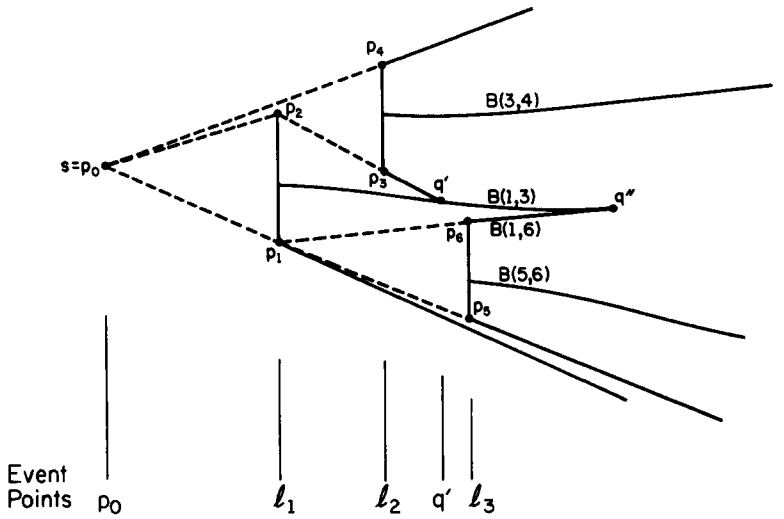


FIG. 10. The current partial SPM.

insert $B(2i-1, 2i)$ and $B(2i, k)$. [*Comment:* In Fig. 11, $B(5, 6)$, $B(1, 6)$, and $B(1, 3)$ are deleted from $T$.]

(a4) *Update* Q *and the SPM.* [Given any two bisectors $B'$ and $B''$, suppose that they intersect to the right of the abscissa MIN($Q$), and let $q$ be their intersection with smallest abscissa. This intersection is said to be *valid* if $x(q) \leqslant \min\{X(B'), X(B'')\}$.] Consider the sequence of bisectors $\mathfrak{B} = (B_1, B_2, B_3, B_4, B_5)$, where $B_2 = B(j, 2i-1)$, $B_3 = B(2i-1, 2i)$, $B_4 = B(2i, k)$, and $B_1$ and $B_5$, if not empty, are, respectively, the predecessor of $B_2$ and the successor of $B_4$ in $T$, and let $q_{l,l+1}$ be the valid intersection of $B_l$ and $B_{l+1}$ $(l = 1, 2, 3, 4)$ [$q_{l,l+1} = \Lambda$ if $B_l$ and $B_{l+1}$ do not intersect to the right of MIN($Q$)].† (*Comment:* In Fig. 11 all five bisectors are present and $q_{23}$ and $q_{45}$ are valid intersections.) If there are valid intersections, select first the one with smallest abscissa. Let it be $q_{k,k+1}$. Set $X(B_k) = X(B_{k+1}) = x(q_{k,k+1})$ and insert into $Q$ the SPM-point record $(x(q_{k,k+1}); y(q_{k,k+1}), B_k, B_{k+1})$. Next disregard, if they exist, $q_{k-1,k}$ and $q_{k+1,k+2}$ and among the remaining valid intersections, if any, select the one with smallest abscissa (let it be $q_{l,l+1}$); $q_{l,l+1}$ is treated exactly as $q_{k,k+1}$ before. [*Comment: At most two* SPM-points are generated by this step, and in our example they are $q_{45}$ and $q_{23}$. Note that once a record $(x^*; y^*, B', B'')$ has been inserted into $Q$ it will not be deleted until the sweep line reaches abscissa $x^*$, even if either $B'$ or $B''$ or both have been previously deleted. This indeed is the case of the intersection $q''$ of $B(1, 3)$ and $B(1, 6)$ in Figure 11. For this reason, a point inserted into $Q$ at this step is called a *tentative SPM-point* until proven *actual* in (b) below.]

(b) *MIN*(Q) *is the abscissa of a tentative SPM-point.* The SPM-point record is $(x^*; y^*, B', B'')$.

†When more than two bisectors meet at a valid intersection (degenerate case), only the two extreme intersecting bisectors are considered. In the present discussion, for the sake of clarity, we deliberately ignore such degeneracies.

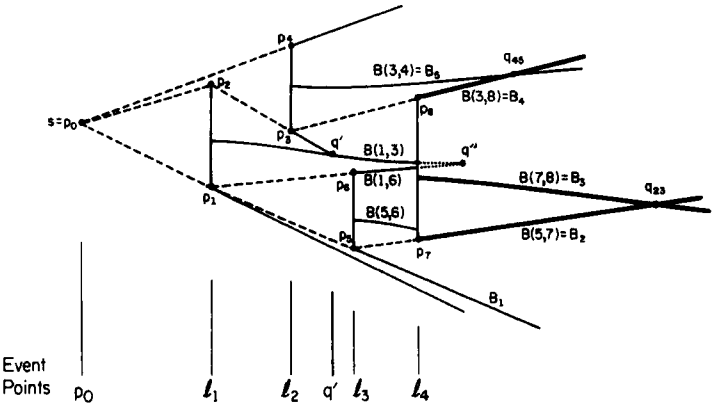


FIG. 11. Action of the algorithm at the insertion of segment $l_4$.

(b1) *Test SPM-record and update* T. To decide if $(x^*, y^*)$ is an actual SPM-point (i.e., a vertex of the SPM), locate $y^*$ in $T$. We have three cases:

1. *Neither* B′ *nor* B″ *is in* T. $(x^*, y^*)$ is not an SPM-point and there is no further action. [*Comment:* $B'$ and $B''$ are not in $T$ owing to deletions effected in (a3) above. This is the case of point $q''$ in Fig. 11, the next position reached by the sweep line.]
2. *Only one of* B′ *and* B″ *is in* T. [*Comment:* Again $(x^*, y^*)$ is not an SPM-point. However, the surviving bisector must be prolonged to the right. This case does not appear in Fig. 11.] Call $B$ the surviving bisector and set $X(B) = \infty$.
3. *Both* B′ *and* B″ *are in* T. Delete $B'$ and $B''$ from $T$. Then if $B(i, j) = B'$ and $B(j, k) = B''$, construct $B(i, k)$, with $X(B(i, k)) = \infty$, and insert it into $T$. $B(i, k)$ is referred to as $B$ in Step (b2) below. [*Comment:* Referring to Fig. 12, the sweep line is first moved to $q_{45}$, where $B(3, 4)$ and $B(3, 8)$ are deleted and $B(4, 8)$ is inserted; subsequently, the sweep line will reach $q_{23}$, where $B(5, 7)$ and $B(7, 8)$ are deleted and $B(5, 8)$ is inserted.]

(b2) *Update* Q *and the SPM.* Consider the sequence $\mathfrak{B} = (B_1, B, B_2)$ where, again, $B_1$ and $B_2$ are, respectively, predecessor and successor of $B$ in $T$. We perform on $\mathfrak{B}$ the same operation described in (a4). (*Comment:* In our example, processing of $q_{45}$ and $q_{23}$ generates no valid intersection.)

The algorithm terminates when $Q$ becomes empty. (*Comment:* The SPM is completed only when $Q$ is emptied. Referring, for example to Fig. 10, assuming $n = 3$—i.e., there are no more segments—still SPM-point $q''$ must be processed to complete the construction.)

### C. Proof of Correctness of the Algorithm

To prove the correctness of the algorithm, we must verify that, after each execution of the general step, $T$ contains the correct sequence of bisectors and that each actual SPM-point in SPM($S^*$) is visited.

That $T$ contains the correct bisector sequence is ensured by the straightforward actions of Steps (a3) and (b1) of the general step.

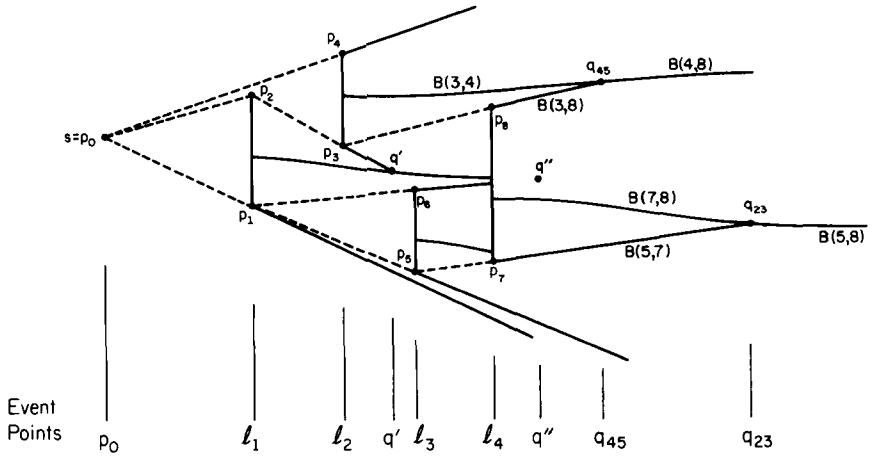


FIG. 12. Further actions of the algorithm when the sweep line is successively moved to $q''$, $q_{45}$, and $q_{23}$.

To show that each actual SPM-point is visited by the algorithm, recall that an actual SPM-point is by definition the intersection of two bisectors and that two bisectors may intersect only if they become adjacent in the vertical ordering maintained by $T$. We have seen that the ordering maintained by $T$ can be modified only at a segment abscissa [as effected by (a3)] or at an SPM-point abscissa (as effected by (b1)]. Any time the ordering is modified, each newly created adjacency of bisectors is tested for intersection, i.e., for the creation of an SPM-point. This shows that all actual SPM-points will be visited by the algorithm.

### D. Performance Analysis

For each created bisector, the algorithm uses a fixed time to generate its equation [Steps (a2) and (b1)] and $O(\log n)$ time to insert it into $T$. Similarly, for each created tentative SPM-point, the algorithm uses a fixed time to construct it [Steps (a4) and (b2)] and $O(\log n)$ time to insert it into $Q$. Since we have shown that the number of bisectors is $O(n)$, the corresponding computational work is $O(n \log n)$. On the other hand, although we know that the number of actual SPM-points is $O(n)$, we may suspect that the number of tentative SPM-points, as created in Steps (a4) and (b2), may be much larger. However, a tentative SPM-point $q$ fails to become actual because at least one, say $B$, of the bisectors meeting at $q$ had been previously truncated by the algorithm [Step (a3)]; thus we may associate $B$ with $q$ and "charge" the computational work of visiting $q$ to the deleted $B$. (Note that no other tentative SPM-point can be charged to $B$.) Since at most $O(n)$ bisectors are truncated, the number of tentative SPM-points which fail to become actual is also $O(n)$, whence the corresponding handling work is $O(n \log n)$. Finally, we recall that also $O(n \log n)$ time was expended to initially sort the set of segments. Combining this time analysis with the remark that the storage requirements of both $T$ and $Q$ are $O(n)$, we have

**Theorem 2.** Given $n$ parallel line segments and a source $s$, the shortest-path map and the shortest-path tree with root $s$ can be obtained in time $O(n \log n)$ and space $O(n)$.

Once the map is available, we can perform a point-location task to locate the destination $t$ in $O(\log n)$ time using one of the known techniques, such as those of [11] and [6]. That is, for any destination $t$, the shortest path between $s$ and $t$ can be found in time $O(\log n) + k$, where $k$ is the number of edges in the shortest path, with $O(n \log n)$ preprocessing time for constructing the shortest-path map and reconstructuring of the map to facilitate point location. Alternatively, once the shortest-path tree is available, we may compute in time $O(n)$ the distance $d(t, p_i)$ between $t$ and each vertex of the tree and choose the minimum of $d(t, p_i) + W(p_i)$ to construct the shortest path from $s$ to $t$. Finally, as a third possibility, we can solve the problem by initially including the destination $t$ in the priority queue $Q$, and when the sweep line reaches $t$, the algorithm will terminate after $t$ is found in some SPM-region. We summarize this in the following theorem.

**Theorem 3.** Given $n$ parallel line segments and two points $s$ and $t$, the shortest path between $s$ and $t$ not crossing any of the line segments can be found in $O(n \log n)$ time, which is optimal to within a constant factor in the decision-tree computation model.

*Proof.* The upper bound follows immediately from the previous discussion. To prove the optimality of the algorithm, we shall reduce set sorting of $n$ numbers to the shortest-path problem. Given $n$ distinct numbers $x_1, x_2, \ldots, x_n$, we shall transform each number into a short vertical line segment with length $\epsilon$, i.e., $x_i$ is mapped to $l_i$ whose upper and lower endpoints are $(x_i, +\frac{1}{2}\epsilon)$ and $(x_i, -\frac{1}{2}\epsilon)$, respectively. Let the source $s$ and the destination $t$ be at positions $(x', 0)$ and $(x'', 0)$ where $x' = \min\{x_1, x_2, \ldots, x_n\} - c$ and $x'' = \max\{x_1, x_2, \ldots, x_n\} + c$ for some constant $c > 0$. For $\epsilon > 0$, the shortest path between $s$ and $t$ must visit each vertical line segment in order. Hence the shortest-path algorithm can sort. We therefore conclude that $\Omega(n \log n)$ is a lower bound for the problem. ∎

## IV. CONCLUDING REMARKS

We have presented algorithms for two instances of the shortest-path-with-obstacles problem, in which the obstacles are line segments in the plane, either parallel or forming a simple polygon.

Generalizations of the parallel-segment case suggest themselves almost naturally, first by allowing the segments to be arbitrarily oriented and second by replacing the segments with more complicated figures, such as rectangles, general polygons, and circles. The case in which the obstacles are $n$ arbitrarily oriented segments was considered in [8] and solved in $O(n^2 \log n)$ time, which is far from being optimal. To obtain a significant improvement over the quoted result seems to be a difficult problem.

On the other hand, it is worth examining the extent to which the technique developed in Section III is applicable. Indeed, the existence of an efficient algorithm crucially rests on the property that "the shortest path never turns backward" (Lemma 2). A proposition analogous to Lemma 2 could be proved for a class of objects with "disjoint shadows," i.e., such that there exists a line $l$ on which they have disjoint projections. This observation, which applies to line segments as well as more complicated figures, could be instrumental for techniques designed to handle the more general cases.

This work was supported by the National Science Foundation under Grants MCS 78-13642, MCS 79-16847, ECS 81-06939, and by the Joint Services Electronics Program under Contract N00014-79-C-0424. A preliminary and partial version of this paper was presented at the conference on Graph-Theoretic Concepts in Computer Sciences, Linz, Austria, 1981.

**References**

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Analysis and Design of Computer Algorithms.* Addison-Wesley, Reading, MA (1974).
[2] O. Chein and L. Steinberg, Routing past unions of disjoint rectilinear barriers. *Networks* **13** (1983) 389–398.
[3] E. W. Dijkstra, A note on two problems in connection with graphs. *Numer. Math.* **1** (1959) 269–271.
[4] S. Even, *Graph Algorithms.* Computer Science, Potomac, MD (1979).
[5] M. R. Garey, D. S. Johnson, F. P. Preparata, and R. E. Tarjan, Triangulating a simple polygon. *Information Processing Lett.* **7** (1978) 175–179.
[6] D. G. Kirkpatrick, Optimal search in planar subdivisions. Manuscript, Department of Computer Science, University of British Columbia (1979).
[7] R. C. Larson and V.O.K. Li, Finding minimum rectilinear distance paths in the paths in the presence of barriers. *Networks* **11** (1981) 285–304.
[8] D. T. Lee, Proximity and reachability in the plane. Ph.D. thesis, Technical Report ACT-12, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign (1978).
[9] T. Lozano-Perez and M. A. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles. *Comm. ACM* **22** (1979) 560–570.
[10] J. Nievergelt and F. P. Preparata, Plane sweep algorithms for intersecting geometric figures. *Comm. ACM* **25** (1982) 739–747.
[11] F. P. Preparata, A new approach to planar point location. *SIAM J. Comput.* **10** (1981) 473–482.
[12] J. Reif, Complexity of the mover's problem and generalizations. In *Proceedings 20th IEEE Symposium on the Foundations of Computer Science*, San Juan, 1979. IEEE, New York (1980), pp. 421–427.
[13] J. T. Schwartz and M. Sharir, On the piano mover's problem. Technical Report No. 41, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University (1982).
[14] M. I. Shamos, Problems in computational geometry. Technical Report, Department of Computer Science, Carnegie-Mellon University (1977).
[15] P. M. Spira and A. Pan, On finding and updating shortest paths and spanning trees. *Proceedings IEEE 14th Annual Symposium on Switching and Automata Theory.* IEEE, New York (1973), pp. 82–84.
[16] M. Tompa, An optimal solution to a wire-routing problem. In *Proceedings 12th ACM Symposium on Theory of Computing*, Los Angeles, 1980. American Chemical Society, Washington, DC (1980), pp. 161–176.
[17] G. E. Wangdahl, S. M. Pollock, and J. B. Woodward, Minimum-trajector pipe routing. *J. Ship Res.* **18** (1974) 46–49.

Received August 28, 1981
Accepted September 23, 1983