SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NEW YORK 14853

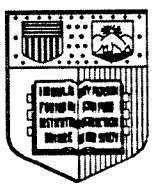
TECHNICAL REPORT NO. 831

March 1988
(Revised: July 1989)

PATH PLANNING IN $0/1/\infty$ WEIGHTED REGIONS WITH APPLICATIONS

By

Laxmi Gewali, Alex Meng[1]
Joseph S. B. Mitchell[2]
Simeon Ntafos[3]



A preliminary version of this paper appears in the *Proceedings of the Fourth Annual ACM Symposium on Computational Geometry*, Champaign-Urbana, IL, June 6-8, pp. 266-278.

[1] Texas Instruments, Inc.
[2] SORIE, Cornell University.
[3] Computer Science, The University of Texas at Dallas

PATH PLANNING IN $0/1/\infty$ WEIGHTED REGIONS WITH APPLICATIONS†

Laxmi Gewali and Alex Meng
Texas Instruments, Inc.
Dallas, TX 75265-5474

Joseph S. B. Mitchell[1]
School of Operations Research and Industrial Engineering
Cornell University
Ithaca, NY 14853

Simeon Ntafos[2]
Computer Science Program
The University of Texas at Dallas
Richardson, TX. 75083-0688

Abstract

We consider the terrain navigation problem in a two-dimensional polygonal subdivision consisting of obstacles, "free" regions (in which one can travel at no cost), and regions in which cost is proportional to distance traveled. This problem is a special case of the weighted region problem for which a polynomial-time algorithm was given by Mitchell and Papadimitriou, and a generalization of the shortest path problem in the presense of obstacles, for which a number of exact and efficient solutions are known. We present an $O(n^2)$ exact algorithm for this problem and faster algorithms for the cases of convex free regions and/or obstacles. Our algorithm is based on building a "critical graph", which is similar to a visibility graph in that it contains an optimal path between any pair of vertices. In addition, we present algorithms to solve a variety of important generalizations and applications: (1) an $O(n^2)$ algorithm for constructing a critical graph of size $O(n^2)$ (which can be searched in time $O(n^2 \log n)$ for shortest paths) for the case in which linear features (roads) are added, thereby allowing arbitrary weights on the edges of the subdivision; (2) similar time bounds for the case in which the linear features include a possible "cost of crossing"; (3) an $O(n^2W)$ algorithm for finding lexicographically shortest paths in weighted regions (with W different weights); (4) an $O(k^2n^2)$ algorithm for planning least-risk paths in a simple polygon that contains k line-of-sight threats (this becomes $O(k^4n^4)$ in polygons with holes); and (5) an $O(k^2n^3)$ algorithm for finding least-risk watchman routes in simple rectilinear polygons (a watchman route is such that each point in the polygon is visible from at least one point along the route).

Key Words: shortest paths, terrain navigation, watchman routes, threat avoidance, visibility graphs, Voronoi diagrams, duality, computational geometry

---

† A preliminary version of this paper appeared in the Proceedings of the Fourth Annual ACM Symposium on Computational Geometry, Urbana-Champaign, June 6-8, 1988.

[1] Partially supported by NSF Grants IRI-8710858 and ECSE-8857642 and by a grant from Hughes Research Laboratories.

[2] Partially supported by a grant from Texas Instruments, Inc.

1. Introduction

The shortest path problem for a point robot moving among obstacles in the plane has been addressed by many authors, and several efficient algorithms are known [AAGHI,GM,KM,Le1,Mi1,Mi6,RS,SS,We]. In three dimensions, the shortest path problem becomes much harder and it is very unlikely that polynomial-time algorithms exist, as the general problem is NP-hard [CR].

The *weighted region problem* (WRP) is a generalization of the standard shortest path problem in the plane in which an arbitrary weight representing the cost per unit distance of travel is assigned to each edge and each face of a polygonal subdivision of the plane. An objective in terrain navigation is to find shortest paths through such a weighted subdivision [Mi2]. An $O(n^7L)$ algorithm that finds ϵ-optimal paths is presented in [MP], where n is the size of the problem and L is the precision of the problem instance. The algorithm actually finds a decomposition of the edges of a triangulation into *intervals of optimality*, which provide the needed information for finding ϵ-optimal shortest paths from one or more source points to any point on an edge. The algorithm uses the continuous Dijkstra technique ([Mi1,MMP]) of sweeping the plane, and Snell's law of refraction as the local optimality criterion. It is the algebraic complexity of the local optimality criterion that appears to make it impossible to solve the weighted region problem exactly.

In this paper, by considering regions with weights in the set $\{0, 1, +\infty\}$, we take a first step towards determining the extent to which non-uniformity of the free space can be introduced into the terrain navigation problem and still allow for exact and efficient solutions. We call the resulting problem the $0/1/\infty$-WRP. The motivation for this set of weights comes from a number of important and interesting applications in which this set provides a reasonable simplification to the general case of arbitrary weights.

In section 2, we consider the $0/1/\infty$-WRP in which we must compute shortest paths among obstacles (infinite-weight regions) and zero-cost regions ("0-regions", where travel is very "cheap") on a background with a fixed cost per unit distance (these regions are assigned weight 1, without loss of generality, and are called "1-regions"). Clearly, a least-weight path from a source to a destination point in such a terrain is one that minimizes the distance traveled on the background (1-regions). We assume that the map is given to us in the form of a planar straight-line subdivision with weights of 0, 1, or $+\infty$ attached to each face and each edge. We present an algorithm that solves this problem exactly (in a model of computation that allows square roots) in time that is at most quadratic in the

size of the scene. The algorithm is based on the observation that shortest paths in this terrain must contain straight line segments that are themselves shortest paths connecting pairs of 0-regions and vertices of obstacles. We construct a special type of visibility graph (called a *critical graph*) which captures the local optimality properties of shortest paths and we reduce our problem to that of finding shortest paths in this graph. We also present somewhat faster algorithms for the cases in which the 0-regions and/or the obstacles are convex.

In section 3, we allow the weights on edges of the subdivision to be arbitrary. In this way, we can model "roads" and other "linear features" which may have weights less than the surrounding terrain. The key idea here is that a shortest path can enter and exit edges in known and limited ways. This allows us again to define a discrete graph which is guaranteed to contain a shortest path. By appealing to some results from geometric duality theory, we show that the complexity of the algorithm to build the critical graph is again $O(n^2)$. (Unfortunately, Dijkstra's algorithm requires time $O(n^2 \log n)$ to run on this graph.) We further generalize our results to allow there to be a cost ξ_e for *crossing* an edge e, and, again, our critical graph construction algorithm runs in worst-case time $O(n^2)$.

In section 4, we solve the *lexicographic* weighted region problem. Given a polygonal subdivision of the plane into regions with arbitrary weights assigned to each region, we want to find a path from a source to a destination point that minimizes the distance traveled within the regions with the largest weight and, subject to that optimization, within regions with the next largest weight recursively. The approach we use is to reduce this problem to a collection of $0/1/\infty$-WRP's. Lexicographically shortest paths may provide a good starting point for approximation algorithms for the general weighted region problem.

In the standard obstacle-avoidance problem we want to find shortest collision-free paths. An important class of problems results when visibility considerations are added. In the watchman route problem, introduced in [CN1], we want to find a route from a starting point back to itself and such that each point in a given space is visible from some point along the route. It has been shown that the problems of finding shortest watchman routes in polygons with holes or in simple (three-dimensional) polyhedra is NP-hard [CN1]. An almost-linear-time algorithm for the special case of simple rectilinear polygons is given in [CN1], and a polynomial-time algorithm for simple polygons is given in [CN2].

A more general version of the watchman route problem is the *robber route problem* ([Nt]) in which we want to find a shortest route from which certain areas are visible (sights)

while at the same time the route is not visible to a specified set of locations (threats). Clearly, this problem has no solution if we insist on seeing the whole polygon from the route since the threat locations can not be seen without exposing the route to them. If we restrict the areas that need to be seen from the route to some subset of the polygon's boundary, the existence of a solution depends on the layout of the threats and the sights. If a solution exists, the robber route problem reduces to the watchman route problem by removing the areas visible to the threats and appropriately modifying the boundary of the resulting polygon [Nt]. Note, however, that a solution may not exist in many cases. Thus, we would like to find "least-risk" watchman routes. In the general version of this problem, each threat will have a different weight associated with it (to reflect the likelihood of been detected or the lethality of the threat) and it may also be appropriate that the risk would vary within the visibility polygon of each threat (e.g., according to distance from the threat, interference with other threats in areas of overlap, etc.). Even without the visibility considerations, a very complex weighted region problem arises (due to the large number of regions that can be induced by the threats).

In sections 5 and 6, we consider least-risk paths and watchman routes in polygonal regions in which there is a specified set of point observers ("threats"). We distinguish two types of regions: those visible to at least one threat and those hidden from all threats. The risk associated with a path (route) is proportional to the distance traveled within view of at least one threat. (This reflects covert penetration situations where avoiding detection is important and the detection threshold is exceeded in the whole area visible to a threat).

The approach we use to find least-risk paths in a polygon with threats is to apply the $0/1/\infty$-WRP algorithm with the boundary of the polygon now playing the role of obstacles. Since the complexity of the $0/1/\infty$-WRP algorithm depends on the size of the scene, it is important to determine the complexity of the subdivision induced by k threats inside a polygon. In section 5, we present a number of results that place tight bounds on the number of the various types of regions induced by k threats inside simple polygons and inside polygons with holes. These results lead to an $O(k^2n^2)$ algorithm for the least-risk path problem in simple polygons (without holes) and an $O(k^4n^4)$ algorithm for polygons with holes (obstacles).

In section 6, we present an $O(k^2n^3)$ algorithm for the least-risk watchman route problem inside simple rectilinear polygons with k threats. This algorithm is based on the fact that a set of edges that any watchman route must visit can be constructed inside the

polygon and an order in which these edges should be visited can be established. It combines ideas used in the standard watchman route algorithm in [CN1] with a modification of the $0/1/\infty$-WRP algorithm applied to appropriately connected copies of a graph obtained from an extended visibility graph inside the polygon.

Finally, in section 7, we discuss some generalizations and further applications of the results presented in this paper and propose problems and directions for further research.

The results in this paper are a combination of the technical reports [GMN] and [Mi3], and they appear in preliminary form in [GMMN].

2. The $0/1/\infty$ Weighted Region Problem

Consider the weighted region problem in which all weights are restricted to be in the set $\{0, 1, +\infty\}$. Specifically, we are given a straight-line planar subdivision (perhaps bounded inside a simple polygon), and we are given a mapping of weights α_f on the interior of each face and α_e on each edge. The subdivision has n vertices, $V = \{v_1, \ldots, v_n\}$, and hence also has $n_e = O(n)$ edges, $E = \{e_1, \ldots, e_{n_e}\}$. The special case we consider is that in which $\alpha_e, \alpha_f \in \{0, 1, +\infty\}$. (In section 3, we will relax the assumption on α_e, letting it be an arbitrary nonnegative integer.) The interpretation of a weight α on a region is that the cost per unit distance in that region is α. Thus, if time is our measure of cost, then α is the reciprocal of the maximum velocity in the region.

We are also given a starting point s and a destination point t. Without loss of generality, we can assume that points s and t are vertices of the given subdivision ($s, t \in V$), since we could think of there being a tiny "point" obstacle at s and at t. We define the weighted length of a path to be the path integral of the piecewise-constant weight function assigned to regions of the plane. Our objective, then, is to find a path from s to t which minimizes the weighted length of the path. We will refer to such a path as an *optimal path* or a *shortest path*.

An *α-region* is a maximal connected set of edges and faces of weight α. We refer to ∞-regions as *obstacles*, we think of 0-regions as places where we can travel at infinite speed, and we consider 1-regions to be the background through which we are able to travel, but we pay some constant amount (without loss of generality, 1) per unit distance. If there are no 0-regions, then we get the usual obstacle avoidance shortest path problem in the plane.

The algorithm described in [MP] for the weighted region problem does in fact solve the case with $\alpha \in \{0, 1, +\infty\}$. But the purpose of this paper is to show that this special case

has structure which allows an alternative approach which is faster and easier to implement. One approach to solving our problem would be to modify the algorithm of [MP] so that it runs more efficiently, exploiting the fact that the local optimality criterion takes on a special form in this case (in particular, the complication of applying Snell's law disappears). Instead, our approach here is to build a special kind of "visibility graph" (called a *critical graph*) which takes advantage of the local optimality criteria. The basic fact that we use is that locally optimal paths must enter a 0-region either at a vertex or perpendicularly at an edge.

A typical picture is shown in Figure 1, where we show obstacles as dark gray regions, 0-regions as white, and 1-regions as a light gray. Here, the entire map lives inside a simple polygon.

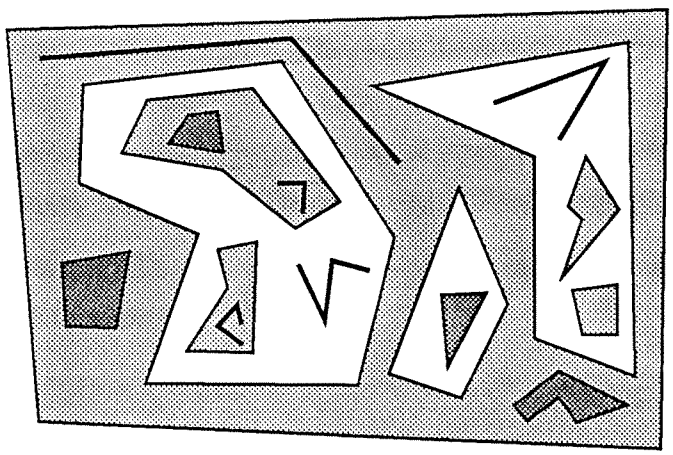


Figure 1. A map for the $\{0, 1, +\infty\}$ weighted region problem.

Note that, in general, the connected 0-regions will be *generalized* polygons with generalized polygonal holes, where the holes are occupied either by 1-regions or by obstacles. For the definition of generalized polygons, see [RS]. An example of such a polygon is shown in Figure 2. Similarly, 1-regions and obstacles will be generalized polygons with generalized polygonal holes. For brevity, we will refer to a generalized polygon with generalized polygonal holes as a *region*. (So a region is just a connected component of a planar straight-line subdivision.)

We begin by recalling some results that are known for shortest paths among obstacles without the presense of 0-regions. By the recent results of [GM,KM], we can build the

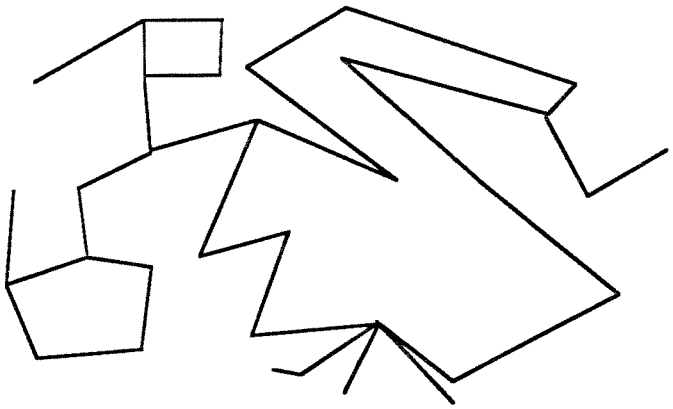


Figure 2. A generalized polygon.

visibility graph induced by a set of obstacles in time $O(n \log n + K)$, where K is the number of edges in the visibility graph. (Recall that the visibility graph is the graph whose nodes are vertices of obstacles and whose edges join vertices for which the line segment between them does not intersect the interior of any obstacle.) Once the visibility graph is constructed, we can use the results of [FT] to search it using Dijkstra's algorithm to find a shortest path in time $O(n \log n + K)$ (see [AAGHI,Le1,Mi1,SS,We]. (An alternative approach that does not rely on visibility graphs yields an $O(mn + n \log n)$ algorithm for shortest paths among m disjoint obstacles with a total of n vertices [RS].)

Consider first the following simple problem in the plane: Find a shortest path from a point v to a polygonal region R. (For the moment, we are not considering anything other than the one point v and the one region R; there are no 0-regions, and distances are Euclidean.) The following lemma characterizes the solution.

Lemma 2.1 *A shortest path from a vertex v to a region R not containing v is a line segment $\overline{vw}$ from v to a point w on the boundary of R such that the relative interior of $\overline{vw}$ does not intersect R, and w is either a vertex of R, or w lies interior to an edge of R such that $\overline{vw}$ is perpendicular to that edge. In case w is a vertex of R, all edges of R incident to w must lie in the closed halfplane H, where H is defined by the line through w perpendicular to $\overline{vw}$ and H does not contain v. (The line defining H is "locally supporting" at point w.) Refer to Figure 3.*

Next, consider the problem of finding a shortest path between two polygonal regions in the plane.

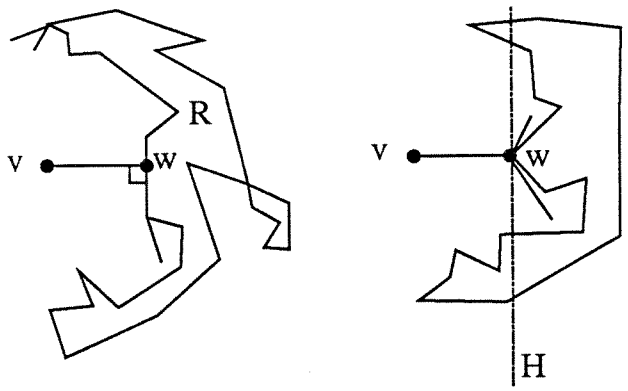


Figure 3. Shortest path from v to region R.

Lemma 2.2 *A shortest path from a region R to a region R' is given by a line segment $\overline{uu'}$ such that the relative interior of $\overline{uu'}$ does not intersect $R \cup R'$, and one of the following cases must hold (refer to Figure 4):*

(i). u and u' are vertices of R and R', respectively, in which case $\overline{uu'}$ makes an angle $\geq \pi/2$ with all edges incident to u or u';

(ii). u is a vertex of R such that $\overline{uu'}$ makes an angle $\geq \pi/2$ with all edges incident to u, and u' lies interior to an edge of R' such that $\overline{uu'}$ is perpendicular to that edge;

(iii). u' is a vertex of R' such that $\overline{uu'}$ makes an angle $\geq \pi/2$ with all edges incident to u', and u lies interior to an edge of R such that $\overline{uu'}$ is perpendicular to that edge; or

(iv). u and u' both lie interior to edges, and $\overline{uu'}$ is perpendicular to both of these edges.

Furthermore, there always exists a shortest path which satisfies one of the first three cases.

Returning now to the context of the $0/1/\infty$-WRP, we now define some terms with respect to our given subdivision. We define a line segment to be *locally optimal* if its interior is disjoint from the obstacles and 0-regions and it is either (i). a shortest path between two 0-regions (with at least one endpoint being a vertex), or (ii). a shortest path from an obstacle vertex to a 0-region such that it is locally tangent to the obstacle at that vertex, or (iii). a segment joining two obstacle vertices such that it is *locally tangent* at each of its endpoints (meaning that the line containing the segment is supporting within a small ball about each of its endpoints [Mi1]). Six cases of locally optimal segments are illustrated in Figure 5, where obstacles are gray and 0-regions are white. We will denote the set of all locally optimal line segments by the letter $\mathcal{S}$.

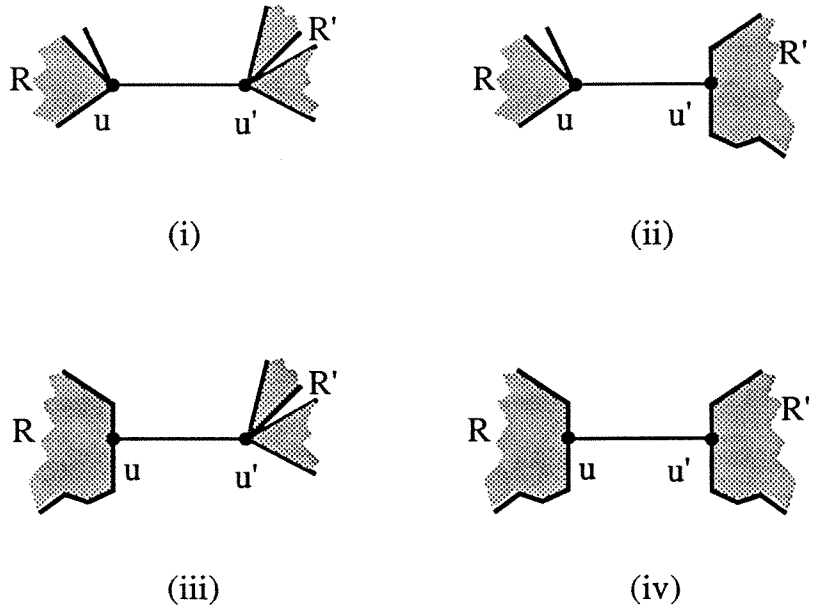


Figure 4. Cases of shortest paths between regions R and R'.

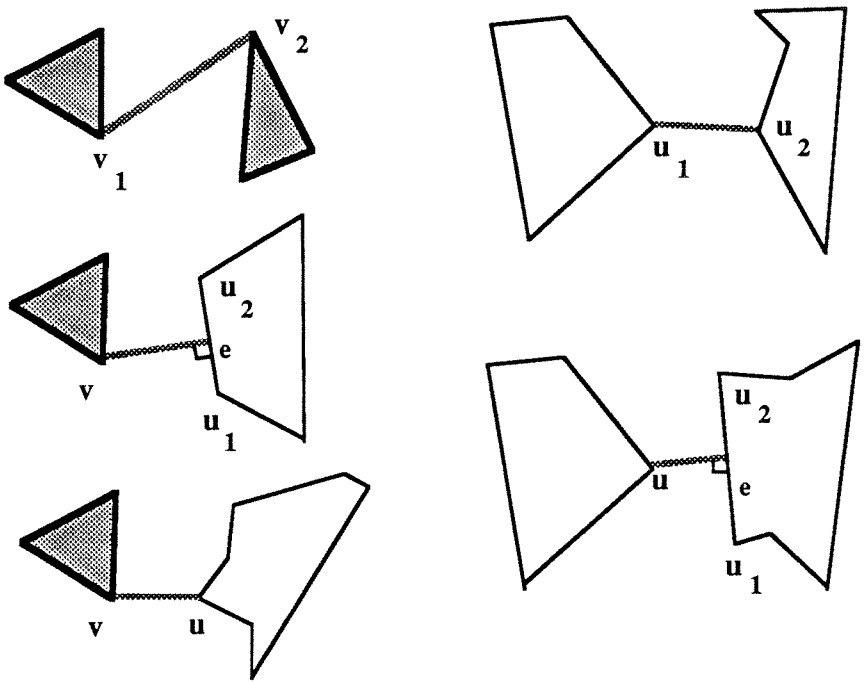


Figure 5. The various types of locally optimal segments $\mathcal{S}$.

We define the set W of *0-entrance points* to be the set of endpoints w of locally optimal segments $\overline{vw}$ for which w is an interior point of a boundary edge of a 0-region (and, hence, $\overline{vw}$ is perpendicular to the edge at point w).

Lemma 2.3 *There are at most $O(n^2)$ locally optimal line segments, and hence also only $O(n^2)$ 0-entrance points.*

Proof: There is at most one locally optimal line segment per vertex/vertex pair or per vertex/edge pair. ∎

Note that our definition of locally optimal segments rules out segments which cross 0-regions. In effect, we are saying that we can treat 0-regions as if they were "obstacles" for purposes of local optimality. The reason for this is that a shortest path need not "cut across" a 0-region: it can enter a 0-region at a vertex of the region or at a normal to an edge of the region. (Once in a 0-region, a path needs to get from one point to another, but an arbitrary path, for instance a path along the boundary of the 0-region, will suffice.) Thus, we have no need for segments which pass through 0-regions, and we can treat the 0-regions as if they were obstacles. (Later, when we consider lexicographic minimization in section 4, we will want to get through 0-regions in the shortest possible way.)

Lemma 2.4 *In searching for an optimal path from s to t, we can ignore line segments whose interior intersects 0-regions.*

Proof: Consider a segment $e = \overline{pq}$ whose interior intersects 0-region R (but does not intersect any obstacle). Let $p' \in R$ be the point of $int(e) \cap R$ that is closest to p. Similarly, let $q' \in R$ be the point of $int(e) \cap R$ closest to q. If $p' = q'$, then p' is a vertex of R, so the segments from p to p' and from p' to q would be considered separately as locally optimal segments. So, assume that $p' \neq q'$. Refer to Figure 6. Then, letting $d(p, q)$ indicate the weighted Euclidean length of segment $\overline{pq}$, we have $d(p, q) \leq d(p, p') + d(p', q') + d(q', q) = d(p, p') + d(q', q)$, since $p', q' \in R$. Thus, $d(p, q) \leq |\overline{pp'}| + |\overline{q'q}| < |\overline{pq}|$, showing that we need not consider segment e, whose Euclidean length is $|\overline{pq}|$. ∎

Lemma 2.5 *If there exists an optimal path from s to t whose length is finite, then there exists an optimal path from s to t that consists of a union of locally optimal line segments through the 1-regions, together with arbitrary subpaths through 0-regions.*

To find shortest paths, we define the *critical graph*, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, as follows:

$$\mathcal{V} = V \cup \{s, t\}$$

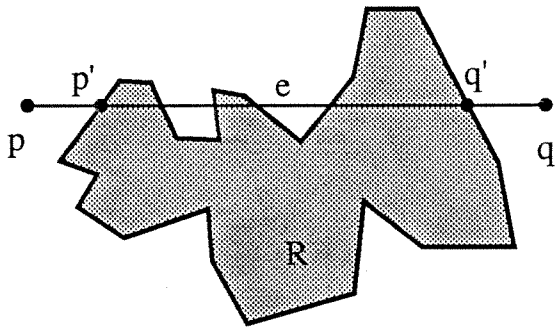


Figure 6. Proof of Lemma 2.4.

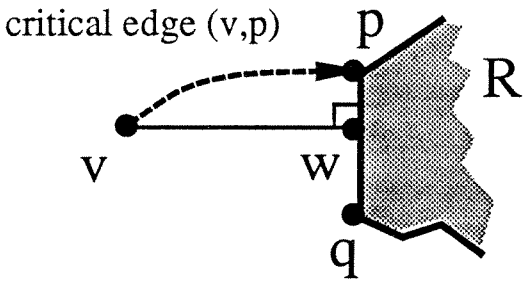


Figure 7. Critical graph edge of type (v,p).

is the set of nodes, and

$$\begin{aligned}\mathcal{E} = E \cup \{(v,v') | v,v' \in V, \overline{vv'} \in \mathcal{S}\} \\ \cup \{(v,p) | v \in V, p \in V, \\ \exists w \in W, \overline{pq} \in E : \overline{vw} \in \mathcal{S}, w \in int(\overline{pq})\}\end{aligned}$$

is the set of edges. In words, edges of the critical graph correspond to three types of line segments: (i). edges E of the subdivision (so it includes the boundaries of obstacles and 0-regions); (ii). locally optimal line segments $\overline{vv'}$ that join two vertices; and (iii). locally optimal segments $\overline{vw}$ that join a vertex v to an edge at a 0-entrance point w (in which case, we consider the critical graph edge to be between the vertex$v$ and one of the endpoints of the edge containing the 0-entrance point w). For edges of the third type, (v,p), we keep a pointer to the 0-entrance point w, so that the path from v to w to p can be reconstructed should the edge (v,p) appear on a shortest path through $\mathcal{G}$. Figure 7 illustrates an edge

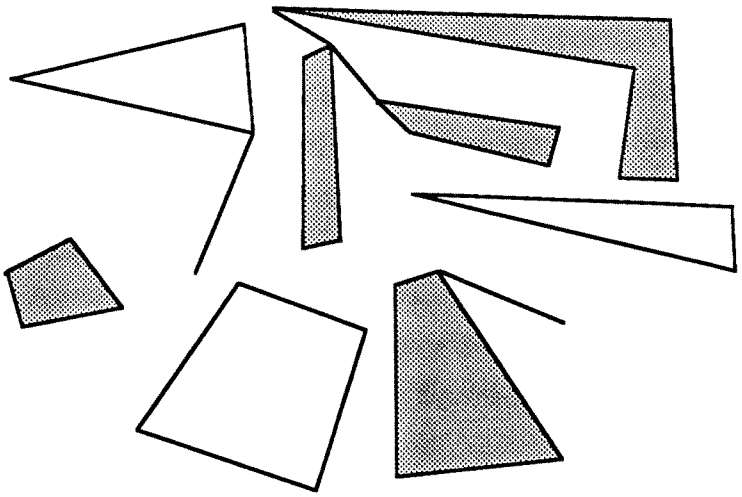


Figure 8. A collection of obstacles and zero-cost regions.

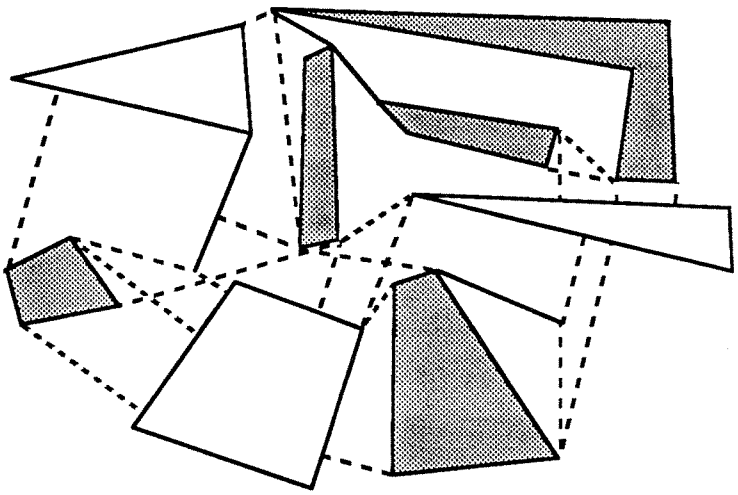


Figure 9. A critical graph.

of type (v, p). For the set of (gray) obstacles and (white) 0-regions of Figure 8, we show the critical graph in Figure 9, where we draw $\overline{vw}$ to represent edges of the type (v, p).

Lengths are assigned to edges of $\mathcal{G}$ as follows. For $e \in E$, we assign the length 0 if $\alpha_e = 0$ or if e borders a 0-region; otherwise, we simply assign the Euclidean length of e. Edges of the form (v, v') get assigned the Euclidean length $|vv'|$. Edges of the form (v, p) get assigned $|vw|$.

Note that, in effect, we have shrunk each 0-region to a single node, since all vertices

of a 0-region are at zero distance from one another.

> *Remark:* We could have simply defined a graph whose nodes are $V \cup W$ and whose edges are $\mathcal{S}$, but this would potentially give us $\Omega(n^2)$ nodes (the worst-case size of W), leading to an $O(n^2 \log n)$ shortest path algorithm, while our goal here is to get a bound of $O(n^2)$. Since each 0-entrance point $w \in W$ is at distance zero from some vertex (an endpoint of the edge on which it lies), we avoid instantiating nodes at points w by making connections directly to these vertices (with edges of the form (v, p)).

The fact that the critical graph will enable us to solve our $0/1/\infty$-WRP is the statement of the following lemma. In effect, we are claiming that the critical graph is shortest paths preserving.

Lemma 2.6 *If there exists an optimal path from s to t whose length L^* is finite, then there exists a path from s to t of length L^* in the critical graph $\mathcal{G}$. Furthermore, if there exists a path from s to t of length L in the critical graph $\mathcal{G}$, then there exists a piecewise-linear path of length L from s to t through the planar subdivision, constructed as follows: for each edge of E or of the form (v, v'), draw the corresponding line segment, and for each edge of the form (v, p), replace it by the two segments $\overline{vw}$ and $\overline{wp}$ (where w is the 0-entrance point associated with (v, p)).*

The critical graph $\mathcal{G}$ can be constructed in much the same way as standard visibility graphs. We appeal to the results of [AAGHI, We] to get a quadratic time bound and to the recent results of [GM, KM] to get an output-sensitive bound.

Lemma 2.7 *The critical graph $\mathcal{G}$ can be constructed in time $O(n^2)$. In fact, if the visibility graph among the set of obstacles and 0-regions contains K edges, then we can build $\mathcal{G}$ in time $O(n \log n + K)$.*

Proof: We apply the visibility graph techniques of [AAGHI, We] or of [GM,KM] to construct the visibility graph among the set of obstacles and 0-regions. This is done in time $O(n^2)$ by the methods of [AAGHI, We], or in time $O(n \log n + K)$ by the new results of [GM, KM], where K is the size of the resulting visibility graph. Now, around each vertex v, we can output the visibility polygon (within the stated complexities). For each edge of the visibility polygon which bounds a 0-region, we check if the foot of the perpendicular, w, from v to the line containing the edge lies on the visible portion of the edge, in which

case we establish an edge of type (v, p) from v to an endpoint p of the edge and we record the 0-entrance point w and distance $|\overline{vw}|$ along with this edge. One can check that all of this additional work can be done in total time $O(K)$ for all vertices v. An example of this construction is shown in Figures 10(a,b,c). ∎

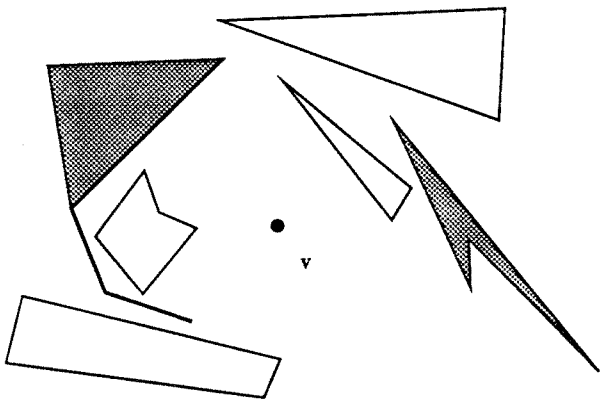


Figure 10(a). Obstacles and 0-regions about vertex v.

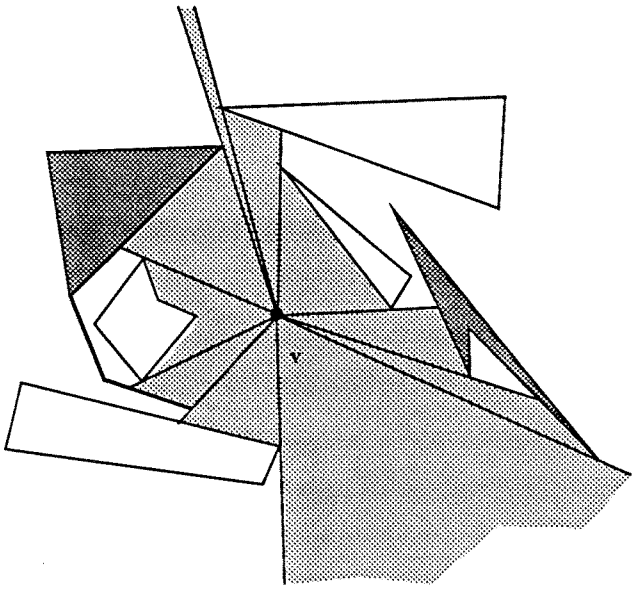


Figure 10(b). The visibility polygon about v.

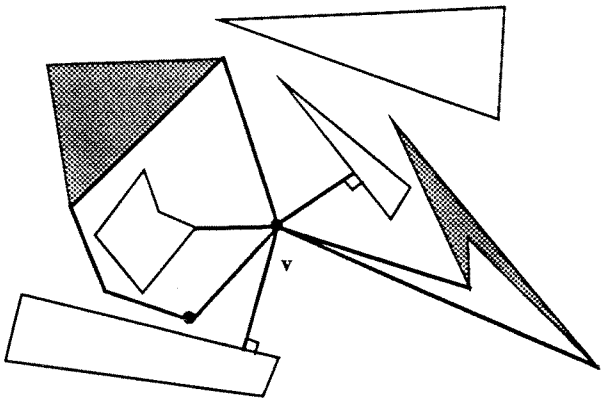


Figure 10(c). Locally optimal segments out of v.

Remark: The number of locally optimal segments in the critical graph may be considerably less than the number of edges in the visibility graph obtained by treating all 0-regions and ∞-regions as obstacles. This raises the interesting open question of whether or not there is an algorithm to build the critical graph in time proportional to *its* size (plus $O(n \log n)$). Note that [KM] provide an output-sensitive bound on building a subgraph of the visibility graph that is relevant to finding shortest paths among obstacles (i.e., that includes only those visibility graph edges that are locally tangent at their endpoints). It is likely that a similar approach may be possible to build our critical graph in time proportional to its size.

Using the results of [FT] to run Dijkstra's algorithm, we get

Theorem 2.8 *The shortest path problem in the presence of $\{0, 1, +\infty\}$ weighted regions can be solved in time $O(n \log n + K)$ (and space $O(K)$), where K is the number of edges in the visibility graph of the n edges defining the subdivision. In fact, within these bounds, one can obtain the shortest path tree from the source point s to all other vertices of the subdivision.*

Remark: Once the shortest path tree is known from s to all other vertices $v \in V$, one can extend the tree to be a *shortest path map* subdivision of the plane by computing the (additive) weighted generalized Voronoi diagram in which the set of Voronoi "sources" consists of the

0-regions together with the obstacle vertices, each source being labeled with its distance from s in the shortest path tree. It is easy to see that the resulting subdivison will consist of straight line segments and parabolic and hyperbolic arcs. Similarly, if we are interested in allowing multiple source points $(s_1, \ldots, s_k)$, then we can build a Voronoi diagram according to the $0/1/\infty$ weighted region distance function by first building a Voronoi "forest" based on the sources $s_1, \ldots, s_k$ (do this by creating a "super source" node σ that is attached by a zero-length edge to each source point s_i, and then find the shortest path tree from σ), and then constructing a weighted generalized Voronoi diagram as before. These subdivisions and their application to maximum flows are discussed in [Mi5].

Special Cases

In some applications, we know that the 0-regions are convex polygons. For example, in the case of the maximum concealment problem with multiple observers, the shadow regions will either be convex or they will be "bays" of the obstacles that cast shadows. Both types of 0-regions can be considered to be convex (since they are convex except for where they contact the boundary of an obstacles, and one will never enter a bay through the obstacle adjoining it).

Assume that we are given a set, $\mathcal{R}_0 = \{R_1, \ldots, R_{m_0}\}$, of m_0 convex 0-regions with a total of n_0 vertices, and a collection of m_∞ obstacles with a total of n_∞ vertices. We call the set of obstacle vertices V_∞, and assume again, without loss of generality, that s and t are obstacle vertices. Then, in the case that $m_0 << n_0$, we can get an improvement as follows.

We define a new type of critical graph, $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$, where

$$\mathcal{V}^* = V_\infty \cup \mathcal{R}_0$$

is the set of nodes, and

$$E^* = E \cup \{(v, R) : v \in V_\infty, R \in \mathcal{R}_0\}$$
$$\cup \{(R, R') : R, R' \in \mathcal{R}_0 \ R \neq R'\}$$

is the set of edges. Basically, we have shrunk the zero-cost regions to single nodes, and connected them to obstacle vertices and other zero-cost regions. The reason that the

simplification is possible here is that an optimal $0/1/\infty$-WRP path will enter a 0-region R either along a segment representing the minimum distance between R and some other 0-region R', or along a segment joining an obstacle vertex $v \in V_\infty$ to the point of R closest to v.

Lengths are assigned to edges as follows: edges in E get their usual Euclidean length; edges of the form (v, R) are assigned the distance from point v to polygon R (that is, the length of the shortest line segment from v to a point of R), if the shortest segment from v to R is obstacle-free and does not cross a zero-cost region, otherwise the edge is assigned a length of $+\infty$; and edges of the form (R, R') are assigned the distance between R and R' (that is, the length of the shortest line segment from a point of R to a point of R'), if the shortest segment is obstacle-free and does not cross a zero-cost region, otherwise the edge is assigned a length of $+\infty$. With these length assignments, it is easy to see that we have captured the appropriate distance information within the critical graph $\mathcal{G}^*$ so that optimal $0/1/\infty$-WRP paths can be obtained by searching $\mathcal{G}^*$:

Lemma 2.9 *Optimal $0/1/\infty$-WRP paths correspond to optimal paths in the critical graph $\mathcal{G}^*$, where connections within 0-regions can be made arbitrarily.*

Lemma 2.10 *The construction of $\mathcal{G}^*$ in the case of convex 0-regions can be done in time $g(m_0, n_0, n_\infty) = O(K_\infty + n_\infty \log n_\infty + (n_\infty + m_0)[\min\{m_0 \log \frac{n_0}{m_0}, n_0\} + m_0 \log(n + m_0)]) = O(K_\infty + n_\infty \log n_\infty + (n_\infty + m_0) m_0 \log(n_0 + \frac{n_0 n_\infty}{m_0}))$, where K_∞ is the size of the visibility graph among (just) the obstacles.*

Proof: The visibility graph among the obstacles can be found in time $O(n_\infty^2)$ ([AAGHI, We]), or in output-sensitive time $O(K_\infty + n_\infty \log n_\infty)$ ([GM,KM]), where K_∞ is the size of the visibility graph among (just) the obstacles. Between each pair of convex 0-regions we must find the minimal-distance segment. This can be done in logorithmic cost per pair of regions, at a total of $O(m_0^2 \log \frac{n_0}{m_0})$, or can be done in $O(n_0)$ per region, at a total cost of $O(m_0 n_0)$. Perform a similar task to compute the segments of tangency between every pair of 0-regions. (These will be needed to determine visibility among the 0-regions.) The shortest distance from each obstacle vertex to each 0-region can be found in total time $O(n_\infty m_0 \log \frac{n_0}{m_0})$ or $O(n_\infty n_0)$, whichever is smaller. Now, to determine which of the segments is feasible (ie, does not intersect the interior of any 0-region or any obstacle), we sort the segments (including the segments from obstacle vertices to 0-regions, segments between 0-regions, and segments of tangency between 0-regions) by slope around each convex 0-region, and perform a standard visibility algorithm on the result.

(This is similar to the technique used in [Mi1,Roh] to compute the necessary part of the visibility graph among a collection of convex obstacles.) The sorting step will require $O(m_0(n_\infty + m_0)\log(n_\infty + m_0))$. ∎

Once we have constructed $\mathcal{G}^*$, we can find shortest paths from s by running Dijkstra's algorithm in time $O((m_0 + n_\infty)^2 + (m_0 + n_\infty)\log(m_0 + n)) = O((m_0 + n_\infty)^2)$ (since $|\mathcal{V}^*| = O(m_0 + n_\infty)$ and $|\mathcal{E}^*| = O(n^2 + n_\infty m_0 + m_0^2)$.

Theorem 2.11 *The shortest path problem in the presence of obstacles and convex zero-cost regions can be solved in time* $O((m_0 + n_\infty)^2 + g(m_0, n_0, n_\infty))$.

Note that this is an improvement over the general case handled by Theorem 2.8 when $m_0 << n_0$ and $K_\infty << K$ (where K is the size of the visibility graph among both obstacles and 0-regions).

Sometimes, in addition to knowing that the 0-regions are convex, we also know that the obstacles are convex. We can get a further improvement here, if the number of obstacles is very small in comparison with the number of vertices defining them (ie, $m_\infty << n_\infty$).

We build the relevant critical graph $\mathcal{G}^*$ in time $O(m_\infty^2 \log \frac{n_\infty}{m_\infty} + m_0 m_\infty \log \frac{n_\infty + n_0}{m_\infty + m_0} + m_0^2 \log \frac{n_0}{m_0})$, where the first term corresponds to finding the relevant subgraph of the visibility graph among the obstacles, the second corresponds to finding the "projection tangents" between obstacles and 0-regions (described below), and the third term corresponds to finding the shortest distance between every pair of 0-regions. The size of $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$ will be

$$|\mathcal{V}^*| = O(m + m_0 + \min\{m^2, n\} + \min\{mm_0, n\})$$
$$|\mathcal{E}^*| = O((m + m_0)^2),$$

and Dijkstra's algorithm can be run in time $O(|\mathcal{E}^*| + |\mathcal{V}^*|\log|\mathcal{V}^*|)$. Again, the basic ideas used to get this improvement are those that [Mi1,Roh] used to improve the visibility-graph-based shortest path algorithm in the presence of convex obstacles.

The result depends on computing the "projection tangents" between obstacles (convex polygons) and 0-regions (also convex polygons). The notion is illustrated in Figure 11, where $\overline{rR}$ is the "right" projection tangent and $\overline{lL}$ is the "left" projection tangent. More formally, projection tangents can be defined as follows. From each vertex of the obstacle, we draw the (unique) segment joining it to the 0-region that minimizes the distance from the vertex to the 0-region. It is easy to see that there will be precisely two vertices (R and L) on the obstacle for which the line containing the segment will be supporting at the vertices. We call the two segments $\overline{rR}$ and $\overline{lL}$ the (*right* and *left*) *projection tangents*

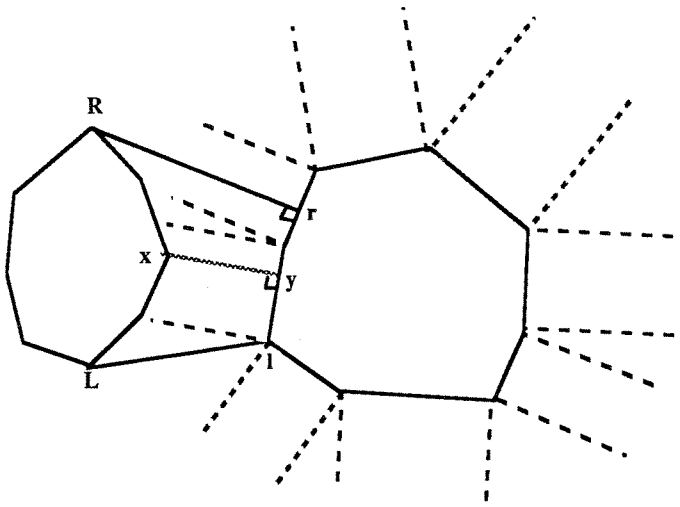


Figure 11. The projection tangents between two convex polygons.

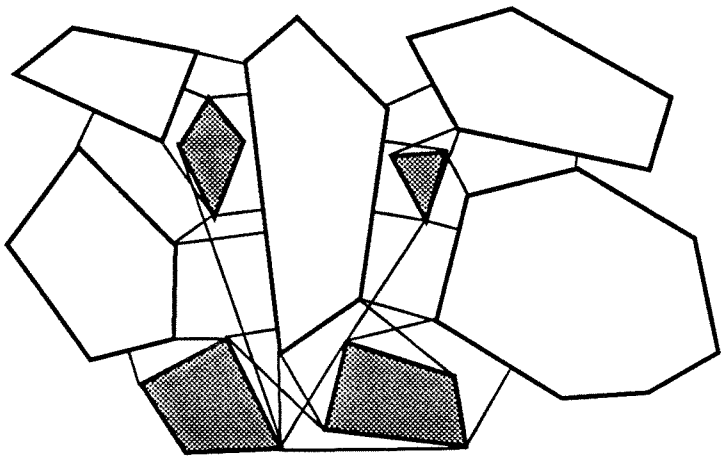


Figure 12. Critical graph for a set of convex 0-regions and convex obstacles.

between the two convex polygons. It is also not hard to show that the projection tangents can be found by doing a binary search in time $O(\log(m+n)$, for polygons of sizes m and n. (This is done in a similar fashion to finding the minimum distance between two convex polygons in logorithmic time. See [PS].)

Figure 12 shows the construction of a critical graph from a set of convex obstacles (shown gray) and convex 0-regions (shown white).

3. Roads

We describe now the case of path planning among a set of linear features. We refer to these one-dimensional features as "roads", although many other interpretations are possible. Linear features may be roads, rivers, ditches, power lines, very skinny weighted regions, etc. Applications include optimal path planning for a mobile robot or optimal layout of pipelines, power lines or phone lines. The model is that the weight attached to the feature indicates a cost of motion or of construction (per unit length) along the feature.

We will begin our discussion by considering only the effect of linear features; thus, there are no obstacles or 0-regions, but only linear features and the background. In the context of the weighted region problem, we are assuming that $\alpha_f = 1$ for all faces f, and that $\alpha_e \in [0, +\infty]$ is arbitrary on the edges. Additionally, we allow there to be a cost $\xi_e \in [0, +\infty]$ for *crossing* edge e. (This models the problem in which it may be possible to go through a fence or cross a river, but only by paying a fixed charge for doing so. If $\xi_e = \infty$, then it is not possible to cross edge e.) We conclude this section by noting that our method generalizes to the case in which there may be obstacles and/or 0-regions present as well as roads and to the case in which there is a cost $\xi_e \in [0, +\infty]$ for *crossing* edge e.

We are given a planar straight-line graph consisting of n straight edges, as shown in Figure 13. Each edge e has a weight $\alpha_e \in [0, +\infty]$ attached to it. (In the figure, roads with greater weight are drawn with a bolder line.) The weight α_e indicates the cost per unit distance of motion along the edge. We shall furthermore assume that there is a cost $\xi_e \in [0, +\infty]$ for *crossing* edge e. (Such a cost models, for instance, the cost of crossing a fence, ditch, or road.) The cost ξ_e is added to the length of any path which crosses the interior of e. Note that if $\xi_e = \infty$, then edge e acts as an obstacle. We are also given a starting point s and a destination point t, which we can assume without loss of generality are vertices of the set of edges.

If a vertex v has more than one edge incident to it, then we must be careful how we define the cost of a path passing through v. For example, v may be a fencepost between two sections of fence, each of which has a nonzero crossing cost ξ, in which case we may want to include ξ in the cost of a path passing through v in such a way as to cross the fence. Thus, depending on the manner in which a path enters and leaves a vertex, there may be a fixed charge associated with it. If k edges are incident to vertex v, these edges may belong to several different connected components of the network, with the components separated

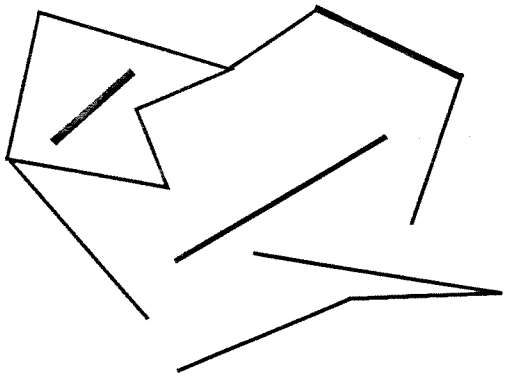


Figure 13. A network of linear features.

by a very small "gap" at v. Refer to Figure 14. There may be $O(k^2)$ different crossing costs at v, depending on which of the k sectors has the path entering and which has the path leaving. (For example, in Figure 14 the crossing cost associated with the path from s to t would be $\xi_1 + \xi_2 + \xi_3$.) We will not want to precompute all of these possible crossing costs, as this would be too time-consuming. Instead, we will take into account the vertex crossing costs when we run Dijkstra's algorithm, being careful to include the cost when we expand a vertex.

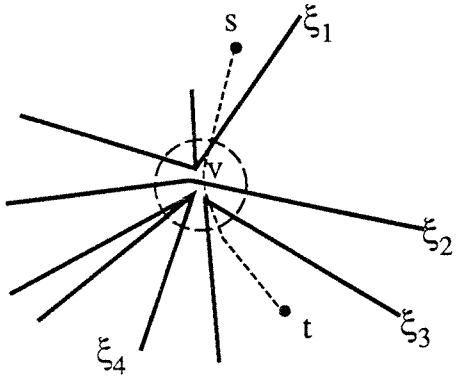


Figure 14. Crossing costs at vertex v.

Note that any edge e for which $\xi_e = 0$ and $\alpha_e \geq 1$ can be eliminated from consideration, since we will never want to travel along such an edge.

Our technique for finding shortest paths is to reduce the continuous problem to a

discrete graph search problem. The key idea is that shortest paths will get onto the road network in a known way: either by entering at a vertex of a road segment or by hitting the interior of a road segment at the "critical angle". These facts are shown in [MP], but we repeat them here for completeness.

For a road edge e, we define the *critical angle* as $\theta_c(e) = \sin^{-1}(\alpha_e)$. The angle that a path makes with an edge e at a bend point $w \in e$ of the path is defined to be the angle between the path and the normal line to the edge at the point w. We call w a critical point of entrance of a path on edge e if the path enters e at point w at the critical angle $\theta_c(e)$. We similarly define a critical point of exit of a path on an edge.

For an endpoint p of an edge $e = \overline{pq}$, we define the *critical cone* at p to be the cone pointed at p of all points r for which $\angle rpq \geq \pi/2 + \theta_c(e)$. Refer to Figure 15.

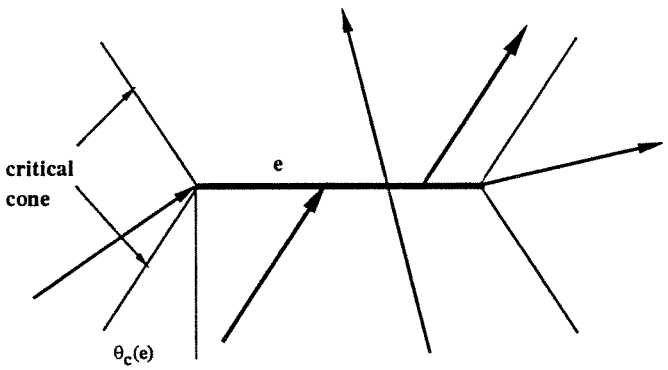


Figure 15. Local optimality conditions at a road edge.

Optimal paths will be piecewise linear and will satisfy the following local optimality criteria, which are illustrated in Figure 15 and proven in [MP].

Lemma 3.1 *An optimal path that enters or leaves a road edge at a point interior to that edge must do so at the critical angle, $\theta_c(e)$, defined by the edge.*

Lemma 3.2 *An optimal path that enters or leaves a road edge at an endpoint of the edge must lie locally within the critical cone at that endpoint.*

Lemma 3.3 *An optimal path which crosses a road edge must do so at an angle of incidence less than the critical angle.*

We also appeal to the result of [MP] which states that between a critical point of exit and a critical point of entry there must be a vertex. This assures that we will not go from

road segment to road segment (entering and leaving at critical angles), and therefore that we will not have to deal with too many critical points of entrance and exit. (Note that in order to go from one edge to another at critical angles would require that the edges be in a degenerate position, as it specifies an exact angle relationship between the two edges.)

Lemma 3.4 *There always exists an optimal path which does not contain segments that join a critical point of exit of edge e to a critical point of entrance of another edge e'. In other words, in our search for an optimal path, we can assume that between a critical point of exit and a critical point of entrance, there must exist a vertex (an endpoint of an edge).*

In the setting of linear features, we can once again define a notion of locally optimal segments and a critical graph. Line segments which join vertices to vertices or vertices to critical points and which additionally satisfy the local optimality conditions of Lemmas 3.1–3.3 are called *locally optimal segments*. Our algorithm constructs the *critical graph* $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$, defined as follows. $\mathcal{V}^*$ is the set of all endpoints of locally optimal segments (these include vertices and critical points), and $\mathcal{E}^*$ is the set of locally optimal segments, together with segments joining all pairs of critical points that are on the same edge and are adjacent. Segments that lie on road edges are assigned a length equal to the product of the edge weight and the segment's Euclidean length; segments that pass through the background are assigned their Euclidean length, plus the sum of the crossing costs of all of the segments each crosses.

Theorem 3.5 *The critical graph $\mathcal{G}^*$ will include an optimal path between every pair of vertices of the subdivision.*

Proof: This theorem follows from Lemmas 3.1-3.4 above and from the definition of the critical graph, which was defined to include all locally optimal segments. ∎

Algorithmically, we can compute the critical graph as follows. For each vertex $v \in V$ and for each edge e, we determine the two critical points on the line through e with respect to v. Portions of the edge e which lie beyond the critical points (for which the angle of incidence from v would be greater than $\theta_c(e)$) can now be treated as obstacles from v, by Lemma 3.3. Compute the visibility polygon from v among these $O(n)$ line segments. This requires $O(n \log n)$ per vertex v. Now, instantiate a node of $\mathcal{G}^*$ at the critical points which are visible from v, and connect v to these nodes via an edge. Also, connect v to any other vertex v' visible from v for which the segment $\overline{vv'}$ lies within the critical cone at v'. Assign lengths to these connections according to the distance from v to the point, plus

the sum of the fixed costs ξ_e of any edges e that are crossed by the segment. (The sum of the crossing costs can be known from the $O(n \log n)$ angular sweep about v which allowed us to compute the visibility polygon.) All of this can be done within an $O(n \log n)$ time bound for each vertex v. An example is shown in Figure 16. The result of making such connections for every vertex v is an $O(n^2 \log n)$ algorithm for making the critical graph, an example of which is shown in Figure 17.

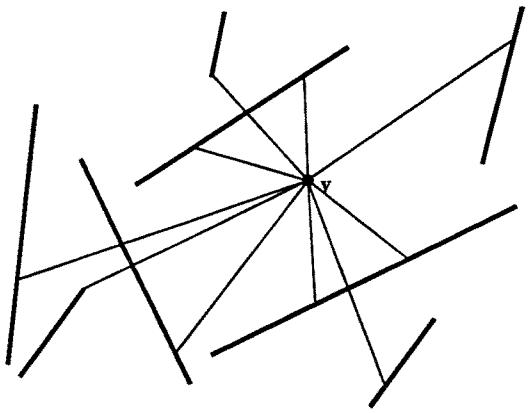


Figure 16. Making connections between a vertex and road edges.

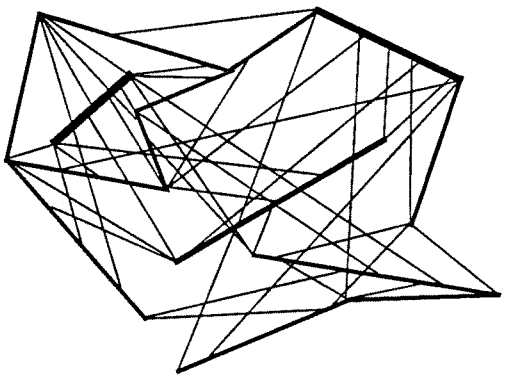


Figure 17. The critical graph.

Assume now that $\xi_e = 0$ for every edge e. Then, the above algorithm can be improved to yield an $O(n^2)$ time bound on building the critical graph as follows. First, note that we cannot directly apply the $O(n^2)$ visibility graph algorithms of [AAGHI,We] since what

is considered to be obstacle depends on the vantage point v; that is, we get a potentially different set of obstacle segments corresponding to each vertex v. However, it is important to notice that the obstacle segments with respect to v are bounded either by one of the original n vertices or by a critical point of entrance to a road edge. The line joining v to a critical point must be oriented in one of $O(n)$ directions, since it intersects the edge at the critical angle for that edge (and each edge defines only two critical directions).

Thus, by the techniques of [LC], we can apply duality to preprocess our original data in time $O(n^2)$ such that the vertices can be given in sorted angular order about each v in time $O(n)$ (rather than $O(n \log n)$). Additionally, we can sort the $O(n)$ critical directions once in advance. Then, to get the directions to endpoints of the obstacle segments in sorted order about v, we have only to do a linear-time merge of these two sorted lists. Thus, in total time $O(n^2)$, we can get the obstacle endpoints (with respect to v) in sorted order about v, for each v. Then, applying the results of [AAGHI], we can determine the visibility from each v in linear time, since the set of important directions is already sorted about v. This allows us to construct the critical graph in $O(n^2)$ time overall.

> *Remark:* Note that we do not really need to worry about treating sections of road edges as obstacles. While it is true that local optimality guarantees that we will never want to cross a road edge at greater than the critical angle, if we ignore this condition in the construction of the graph, it will automatically hold in any shortest path. But our discussion above shows that it does not hurt us in terms of worst-case time to take into account visibility, and it may help us by producing a smaller critical graph. Having a smaller critical graph will make queries run faster if we solve shortest path problems between many different pairs of points.

We have now seen that the critical graph can be constructed in quadratic time. If one is to obtain an $O(n^2)$ shortest path algorithm, the following issue must be addressed: the critical graph may contain $\Omega(n^2)$ nodes, due to the fact that there may be $\Omega(n)$ critical points established on each road edge. This implies an $O(n^2 \log n)$ bound for running Dijkstra's algorithm to find a shortest path in $\mathcal{G}^*$. By applying a duality argument similar to the one previously described, we can get the critical points in *sorted* order along each edge. We originally thought that this allowed us to find shortest paths in time $O(n^2)$ [GMMN], but we discovered a flaw in our original approach. It remains open at this time to improve

the running time of Dijkstra's algorithm on the critical graph we construct. We suspect that the special structure present in our problem will allow a quadratic-time algorithm. Note, however, that the size of the shortest path map (see [Mi1,Mi6] for definitions) can in fact be $\Omega(n^2)$, since, as shown in Figure 18, there can be $\Omega(n^2)$ distinct topologies of shortest paths from s to other points in the plane.

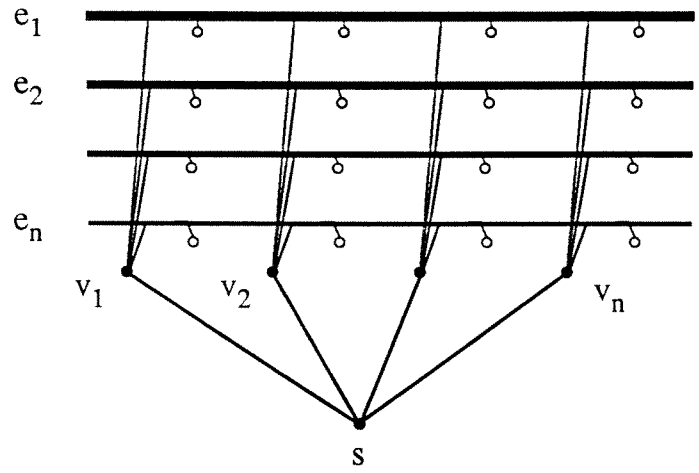


Figure 18. A case in which there are $\Omega(n^2)$ distinct shortest paths.

Another issue that deserves a remark is that of handling crossing costs at vertices when running Dijkstra's algorithm on the critical graph. When a node v is permanently labeled in Dijkstra's algorithm, we must next examine its neighbors and assign to them temporary labels. When doing so, we are careful to add in any relevant crossing costs: we know v's predecessor v' (the node from which we just permanently labeled v), and we look at each other vertex u adjacent to v. The temporary label assigned to u (assuming it is less than any existing label on u) will be v's label plus the (possibly weighted) distance from v to u, plus any relevant crossing costs. We can determine the crossing costs simply by walking around v, visiting its neighbors in order and comparing our position in its neighbor list with our position in the list of road edges incident on v. Since we can assume that both lists (the list of neighbors in the critical graph and the list of incident road edges) are available in sorted order, it is simple to do this angular scan in time proportional to the degree of v. Thus, the consideration of vertex crossing costs does not increase the running time.

Theorem 3.6 *Given a piecewise-linear network of roads of size n without crossing costs ($\xi_e \in \{0, \infty\}$), the critical graph (which contains an optimal path between any pair of vertices) can be found in time $O(n^2)$ (and space $O(n^2)$), after which Dijkstra's algorithm can be used to compute the shortest path tree from any source vertex in time $O(n^2 \log n)$.*

Consider now the case in which we allow a possibly nonzero crossing cost $\xi_e \in [0, \infty]$ on edges. The main new feature of our algorithm is that we need to consider line segments that cross several road edges (as long as they cross while obeying the local optimality criteria), and we need to know for each such segment the sum of the crossing costs for every road edge crossed. The naive means of doing this would be to sweep about each vertex in time $O(n \log n)$. Instead, we see below that the computation can be done in total time $O(n^2)$.

Lemma 3.7 *Given a collection of n line segments in the plane that are disjoint except possibly at their endpoints, and given a weight ξ_e attached to each segment e, we can compute in time $\Theta(n^2)$ the set of all numbers $c(v, v')$, where $c(v, v')$ is defined to be the sum of the weights of all edges that are crossed by the line segment joining endpoint v to endpoint v'.*

Proof: (Adapted from Edelsbrunner [Ed].) The algorithm follows closely the method of Welzl [We]. For each vertex v we keep track of a function $C(v, \theta)$, which gives the sum of the weights of all edges crossed by the ray that leaves vertex v in direction θ. (In comparison, Welzl's algorithm keeps track of the first segment hit by a ray leaving vertex v in direction θ.) It should be clear that the function $C(v, \theta)$ changes only at the $2n - 1$ critical values of θ for which the ray eminating from v passes through another vertex. (Assume for simplicity that no three vertices are colinear.) Now, by the duality result of [LC], we can sort the vertices angularly about every other vertex in total time $O(n^2)$. This gives us a sorting of the critical values of θ about each vertex v, but does not give a a sorting of all $O(n^2)$ critical values of θ taken together. Instead, following Welzl, we do a topological sort of these $O(n^2)$ values of θ. The topological sort allows us to guarantee that at the instant we need to update the value of the function $C(v, \theta)$ at the critical value $\theta = \theta_{v,u}$ that points in the direction of vertex u, we already know the value of $C(u, \theta_{v,u})$, so the update is simple. Edelsbrunner [Ed] has shown that the use of topological sweep ([EG]) can reduce the working space in the above algorithm to $O(n)$. ∎

Thus, we see that Theorem 3.6 can be generalized to include the case of nonzero

crossing costs within the same $O(n^2)$ time bound.

Corollary 3.8 *Theorem 3.6 generalizes to the case in which we allow crossing costs ξ_e.*

The general problem of handling roads, 0-regions, and obstacles can be solved by combining the results of this section and the last. We build a composite critical graph which takes into account the local optimality criteria of how optimal paths behave at obstacle vertices, how they behave at 0-region vertices and edges, and how they behave at road edges and vertices. The overall result is the following theorem.

Theorem 3.9 *Consider the weighted region problem in the case that $\alpha_f \in \{0, 1, +\infty\}$ for each face f and $\alpha_e \in [0, +\infty]$ for each edge e. Additionally, there may be a crossing cost ξ_e for crossing an edge e. Then, we can in time $O(n^2)$ build a discrete (critical) graph of size $O(n^2)$ such that the graph contains an optimal path between any pair of vertices. Dijkstra's algorithm can be applied to search the graph in time $O(n^2 \log n)$. Here, n is the total number of vertices describing the planar subdivision.*

It is straightforward to generalize our results to the case in which edges may be directed, so that there is a different cost of motion in each direction along an edge. We could also consider edges to have two "sides", with a possibly different speed on each side, and the sides are not accessible to each other. For example, we may be modeling a divided highway whose median is not traversable. This can be modeled in our framework by sandwiching an obstacle edge (with $\xi = +\infty$) between two edges of possibly different speeds.

4. The Lexicographic Weighted Region Problem

An important application of our techniques is to solve the "lexicographic" weighted region problem. We want to minimize the length of path in the most expensive region, then subject to this being minimized, we want to minimize the length of path in the second most costly region, then subject to this being minimized, etc. More formally, let the region weights be $0 < \alpha_1 < \alpha_2 < \ldots < \alpha_W < +\infty$. Then, for a path $\mathcal{P}$ let $(d_W(\mathcal{P}), d_{W-1}(\mathcal{P}), \ldots, d_1(\mathcal{P}))$ be the vector of subpath lengths, where $d_i(\mathcal{P})$ is the length of the path $\mathcal{P}$ which passes through regions of weight α_i. Our goal is to minimize, lexicographically, the vector $(d_W(\mathcal{P}), d_{W-1}(\mathcal{P}), \ldots, d_1(\mathcal{P}))$ over all paths $\mathcal{P}$ from s to t.

Our solution proceeds as follows. Rescale the weights so that $\alpha_W = 1$. Treat all regions with weights less than α_W as if their weights were zero, and run the $0/1/\infty$-WRP

algorithm to find the shortest path in time $O(n^2)$. Now take the resulting solution path and look at its subpaths through regions with weight less than α_W. Solve each subpath problem by rescaling the second-highest weight to be 1, and then continue recursively until we reach the lowest non-zero weight. (Note that, while there may be $O(n)$ subpath problems, they can all be solved in total time $O(n^2)$, as the sum of the sizes of the problems is at most $O(n^2)$.) The result is a path which is lexicographically optimal. The overall complexity will be $O(n^2W)$.

Theorem 4.1 *The complexity of finding the lexicographically shortest path through weighted regions is $O(n^2W)$, where n is the number of vertices in the planar subdivision and W is the number of differently weighted regions.*

As a corollary, we get the fact that we can solve the lexicographic shortest path problem in the $\{0, 1, +\infty\}$ weighted region problem in time $O(n^2)$ (since $W = 1$ in this case).

This approach is potentially very valuable to region planners, as it quickly (in polynomial time) finds a "good" path which can be further improved by applying Snell's Law of Refraction (the local optimality condition) at each region boundary. This gives a fast heuristic for solving the weighted region problem. Note, however, that the weighted length of a lexicographically minimal path may be arbitrarily bad in comparison with the shortest weighted length path. The problem is that in order to save a few inches in a weight α region, we are willing to go arbitrarily many miles in a weight $\alpha' < \alpha$ region, and α' may be very close to α.

An idea which improves the performance of the heuristic is to subdivide the weight α' regions into smaller pieces (with, perhaps a bounded ratio of length to width) and to put fixed costs on the edges used for the subdivision. Then, the very long paths through the weight α' regions are not "free" anymore. This approach is similar to the technique of using a "region graph", as explained in [MP,Mi2].

5. Least-Risk Paths in a Polygon With Threats

In this section we consider the problem of finding a least-risk path in a polygon P with k threats when the risk associated with a path is proportional to the distance traveled within view of at least one threat. The threats are assumed to be line-of-sight and have unlimited range, i.e., every point inside the visibility polygon of a threat is at risk. The problem of finding a least-risk path from a source to a destination point inside P reduces

to the $0/1/\infty$-WRP by treating the portions of the interior of P that are not visible to any threat as 0-regions, those visible to at least one threat as 1-regions and treating the exterior of P as an obstacle. Since the complexity of the $0/1/\infty$-WRP algorithms depends on the size of the scene and especially on the number of 0-regions, we need to determine the complexity of the subdivision induced by k threats inside the polygon.

Since $\lfloor n/3 \rfloor$ threats are sufficient to cover any simple polygon completely [O'R], it is reasonable to assume that k is at most $O(n)$.

The region inside P that is visible to a threat T_i is described by the visibility polygon VP_i for the point T_i. Visibility polygons inside P can be constructed in time $O(n)$ if P is a simple polygon [EA,Le2] and $O(n \log n)$ if P contains holes [Le1] (or time $O(n)$, given $O(n^2)$ preprocessing [AAGHI]). The boundary of each visibility polygon consists of portions of the boundary of P and straight line segments, called *visibility edges* that are associated with reflex vertices inside P and define the boundary between the area seen by the threat and the area where the threat is blocked by the reflex vertex. It is the set of visibility edges for the various threats and their intersections inside P that partition P into regions.

First we determine the number of regions induced by k threats when each threat is distinct, i.e., each (connected) region is defined by the subset of threats to which it is visible.

Lemma 5.1 *Each threat generates $O(n)$ visibility edges.*

Proof: There is a one-to-one correspondence between the visibility edges generated by a threat and the reflex vertices on the boundary of P that are the endpoints of the edges and also lie closest to the threat. Since there can be $O(n)$ reflex vertices in a polygon with n sides, the lemma follows. ∎

Thus, the total number of visibility edges generated by k threats is $O(kn)$. Now, m line segments in the plane can produce an arrangement of size $\Omega(m^2)$. However, in our case, the segments (visibility edges) are not arbitrary, and the boundary of the polygon prevents certain intersections. Consider the set E_i of visibility edges generated by threat T_i.

Lemma 5.2 *A visibility edge in E_j can intersect at most two visibility edges in E_i.*

Proof: The visibility edges in E_i are ordered angularly about the threat T_i. A straight line inside P can intersect one visibility edge entering VP_i and another leaving it. Clearly, any additional intersections would imply that the line intersects the exterior of P. ∎

Lemma 5.3 *The number of distinct regions inside P induced by k threats of different weights is $O(k^2 n)$.*

Proof: Each intersection of visibility edges inside P can create at most one additional region. The set E_i can have at most $O(n)$ intersections with the set E_j by Lemma 5.2. Since there are $O(k^2)$ possible threat pairs, the lemma follows. ∎

The general least-risk path planning problem allows a different weight to be assigned to each threat T_i. Thus, we get a weighted region problem with $O(k^2 n)$ $(= O(n^3))$ regions. Using the algorithm in [MP], we could obtain an ϵ-optimal solution in $O(n^{21} L)$, which is clearly impractical.

In many cases, it is reasonable to assume that the areas visible to one or more threats have *equal* risk. Then the risk faced by a path is proportional to the distance traveled within view of a threat and the regions that are not visible to any threat become very important as they allow travel at no cost. These regions will become 0-regions in a $0/1/\infty$-WRP formulation of the problem. Note that the portions of the boundaries of these 0-regions that are formed by the visibility edges must be convex. For this case, the method described in section 2 will construct a graph whose vertices correspond to the reflex vertices in the boundary of P and the 0-regions in the interior of P. Thus, it is important to determine the number of 0-regions induced by k threats inside P.

Lemma 5.4 *The number of risk-free regions induced by k threats is $O(k(k+n))$. Furthermore, the total number of edges on the boundaries of all of the risk-free regions is also $O(k(k+n))$.*

Proof: Consider the line segments produced by the intersecting visibility lines inside P that are on the boundary of a 0-region. We partition this set into two types of segments. Type-A segments are those that have at least one endpoint on the boundary of P. Type-B segments are those with both endpoints formed by intersections of visibility edges. Consider now the number of segments of each type that can occur in the set E_i, the visibility edges for threat T_i. Since there are at most two type-A segments per visibility line, it follows from Lemma 5.1 that the total number of type-A segments in E_i is $O(n)$.

Suppose now that the polygon P contains only one threat, say T_i, and suppose that we add the remaining threats one at a time. Let us consider what happens along some visibility edge e_x in E_i. There will be a first time (if ever) when a type-B segment is created along e_x. As additional threats are added, this segment may become smaller or it may be

split into two type-B segments separated by a segment that appears in the boundary of a 1-region. It is this latter event that increases the number of type-B segments. In order for it to occur, we must have a threat that projects a "thin beam" of visibility across e_x, i.e., the "middle" of e_x is visible to the threat while there must be two reflex vertices which block the threat from seeing the portions on either side of the visible portion. The key observation we need now is that this same threat can cause the same thing to occur in at most one more edge in E_i. Then, the total number of times that the number of type-B segments in E_i can be increased is $O(k)$. So, we have $O(k+n)$ segments of both types in VP_i and the total number of segments on the boundaries of 0-regions is $O(k(k+n))$. Clearly, the total number of 0-regions can be at most $O(k(k+n))$ since these segments can not be shared. ∎

We note that the upper bound on the number of 0-regions established by Lemma 5.4 can be achieved. For example, consider $O(k)$ threats arranged to produce $O(k)$ thin intersecting beams as shown in the top portion of the polygon in Figure 19 and another $O(k)$ threats arranged to produce $O(k)$ shadows within each of $O(n)$ "caves" as shown in the bottom portion of Figure 19.

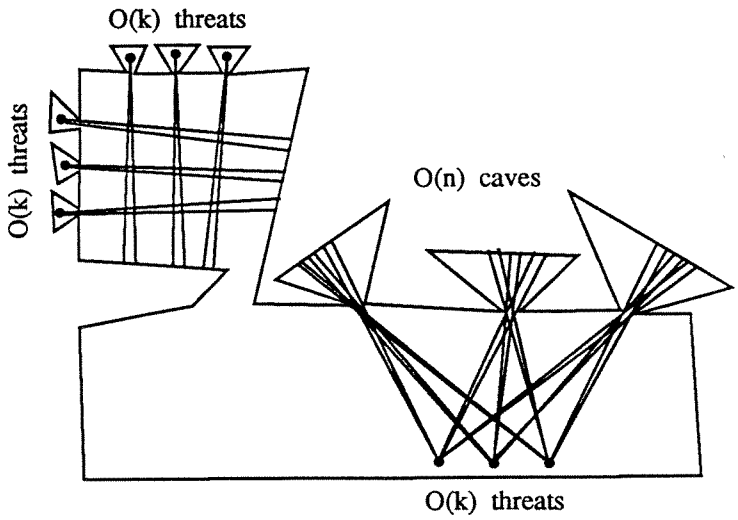


Figure 19. Achieving the upper bound in Lemma 5.4.

To construct a least-risk s-t path in a simple polygon with k threats, we use the $0/1/\infty$ weighted region algorithm of section 2. From Lemma 5.4 we have:

Theorem 5.5 *A least-risk s-t path inside a simple polygon P can be constructed in time $O(k^2(k+n)^2)$ $(= O(k^2n^2)$ under the reasonable assumption that $k \leq O(n))$.*

We note that 0-regions induced by threats (the risk-free regions) are either convex or their non-convex parts are shared with the boundary of the polygon, meaning that we can disregard them since a shortest path will never enter there. Then, if the number of 0-regions is small, a faster algorithm can be obtained using Theorem 2.11.

When the bounding polygon has holes, a similar approach can be used. The only change is in the number of 0-regions that are generated by k threats.

Lemma 5.6 *The number of 0-regions induced by k threats inside a polygon with holes is $O(k^2n^2)$. Furthermore, the number of edges on the boundaries of all of the risk-free regions is also $O(k^2n^2)$.*

Proof: The number of visibility edges generated by a threat is still $O(n)$, where now n is the total number of vertices in the boundary of P and the holes (obstacles) inside P. The number of type-A segments remains the same but the number of type-B segments in each visibility polygon can now be $O(kn)$ because each of the (up to) $O(n)$ possible obstacles can act as a "beam-splitter". Then, the total number of edges on the boundaries of the 0-regions is $O(k^2n^2)$, which is also a bound on the number of 0-regions. ∎

Again, the upper bound of Lemma 5.6 can be achieved as shown in Figure 20.

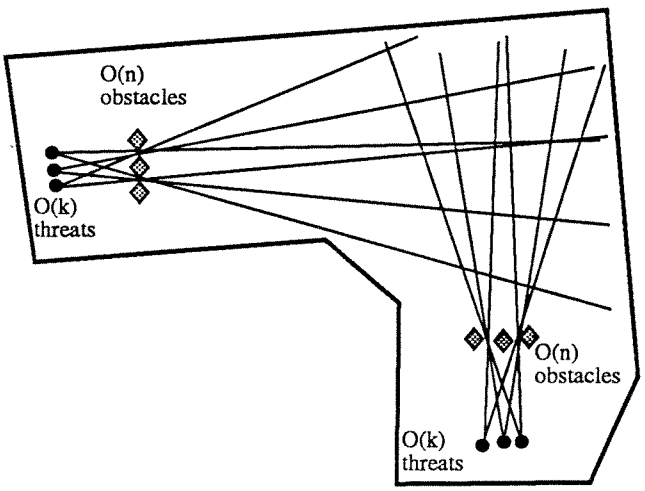


Figure 20. Achieving the upper bound in Lemma 5.5.

Combining Lemma 5.6 with the $0/1/\infty$-WRP algorithm of section 2, we have:

Theorem 5.7 *A least-risk s-t path in a polygon with holes and k threats can be constructed in time $O(k^4n^4)$, where n is the total number of vertices in the polygon and the holes.*

6. Least-Risk Watchman Routes

In this section we consider the problem of finding a least-risk watchman route in a rectilinear polygon with n sides and k threats. From Lemma 5.4, we have that k threats can induce $O(k(k+n))$ risk-free regions inside a polygon. This bound holds for rectilinear polygons as well. Since it is known that finding a shortest watchman route in a rectilinear polygon with rectilinear holes is NP-hard [CN1], it follows directly that the least-risk watchman route problem in such polygons is also intractable (consider a polygon with holes whose interior is completely covered by 1-regions). Thus, we consider simple rectilinear polygons.

In [CN1], a linear-time algorithm (after triangulation) for the standard watchman route problem in simple rectilinear polygons is presented. The approach used in [CN1] is to identify a set of essential edges (cuts) inside the polygon such that a shortest watchman route (a) must visit these essential edges in the order in which they occur in a scan of the boundary of the polygon and (b) will never enter the regions of the polygon past these essential edges. We assume that a starting point for the watchman route is specified by a point x on the boundary of the polygon.

The construction of a shortest watchman route in a rectilinear polygon is illustrated in Figure 21. Figure 21a shows the essential edges for a rectilinear polygon and the (shaded) regions in which no shortest watchman route need enter. The algorithm proceeds by triangulating the polygon and reflecting appropriate portions of it with respect to each of the essential edges so that the problem of finding a shortest watchman route is reduced to that of finding a shortest path from a point to an image of itself in the rolled-out polygon (Figure 21b). A shortest watchman route is obtained by folding back the polygon along each essential edge (Figure 21c).

When we consider least-risk watchman routes, the presence of 0-regions alters the situation. It is still the case that an order in which the least-risk watchman route should visit the essential edges can be established. However, it is now possible that the route will visit each essential edge many times and it may enter the regions behind the essential edges in order to take advantage of travel in 0-regions. For example, in Figure 22, the

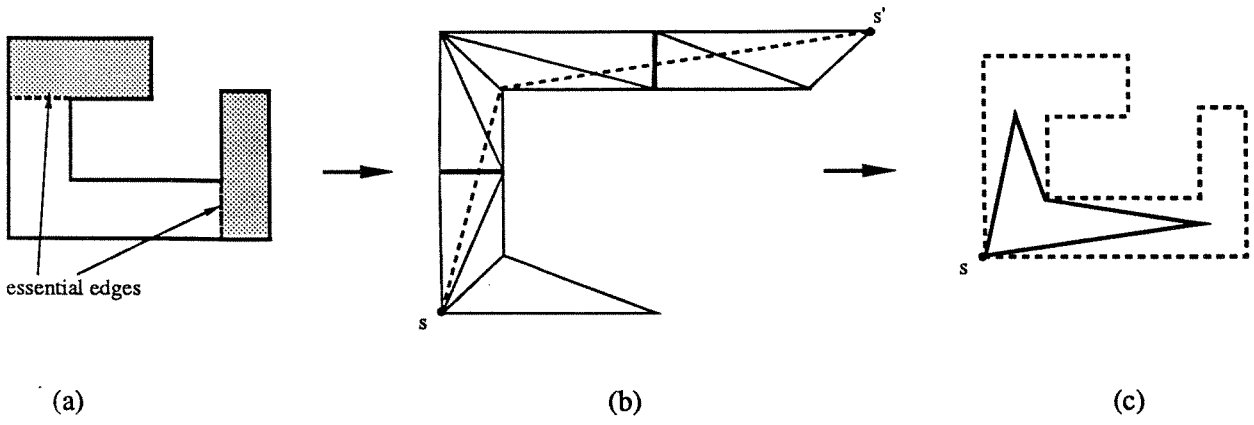


Figure 21. Finding shortest watchman routes in simple rectilinear polygons.

watchman route enters the region below the long horizontal essential edge, in order to take advantage of the 0-region where travel is hidden from both threats t_1 and t_2.

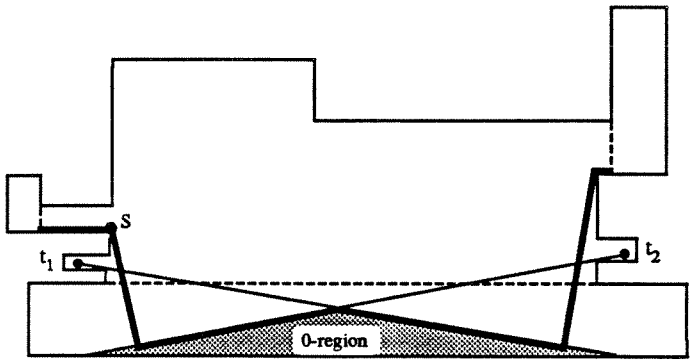


Figure 22. Crossing an essential edge to take advantage of 0-regions.

An important property retained by shortest watchman routes in the presence of 0-regions is that there must exist a least-risk watchman route that does not "cross" itself: if a route crosses itself, it can be replaced by a route that is at least is good but does not cross itself (see Figure 23).

Lemma 6.1 *There is a shortest watchman route that does not cross itself.*

This allows us to combine some of the ideas in [CN1] with our $0/1/\infty$-WRP algorithms to obtain a least-risk watchman route.

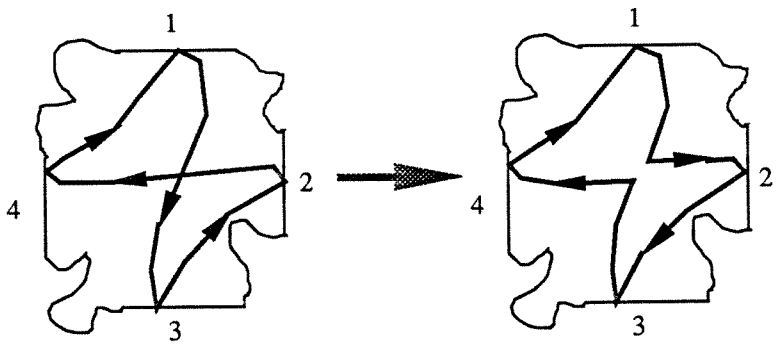


Figure 23. Improving a route that crosses itself.

The approach we take is to create up to $O(n)$ copies (one for each essential edge) of the critical graph $\mathcal{G}$ used in finding least-risk paths inside P. Each copy is indexed by the corresponding essential edge. Connections within each copy are the standard ones specified by our critical graph construction. Most of the connections betwen copies occur for copies with successive indices. These connections represent: (1) shortest paths that connect a 0-region or a reflex vertex (in the i^{th} copy) to a 0-region or a reflex vertex (in the $(i+1)^{st}$) copy by making a (perfect) reflection on the essential edge; and (2) shortest paths connecting a 0-region behind the essential edge to a 0-region or reflex vertex on the other side of the essential edge (there will be two such connections, one for each copy). These two types of connections are illustrated in Figure 24.

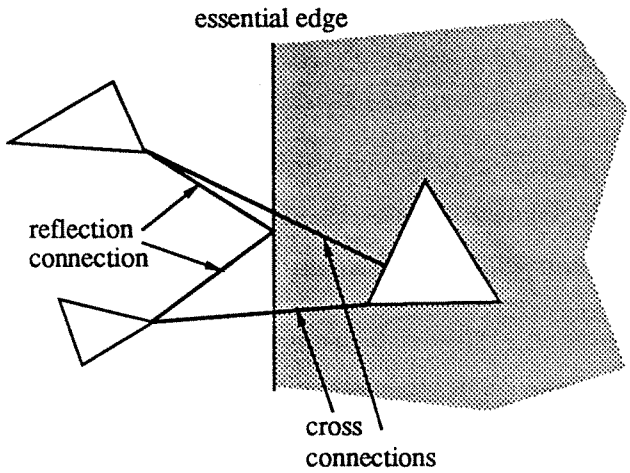


Figure 24. Reflection and cross connections.

Finally, we add connections between copies with non-successive indices to account for shortest paths that connect reflex vertices and/or 0-regions by making multiple reflections on essential edges. Figure 25 illustrates the case in which there can be several reflections on essential edges along an optimal watchman subpath between two 0-regions (shown shaded) and shows how the corresponding shortest paths can be constructed. These multiple reflection paths will be represented in our graph as connections between the endpoints of each path in the appropriate copies of the critical graph. For example, a path that makes three reflections starting with the i^{th} essential edge will be represented by a connection between the i^{th} and the $(i+3)^{rd}$ copies of the critical graph. Note that the number of reflections that a shortest path can make without visiting a reflex vertex or a 0-region is severely limited in most cases. Since the polygon is rectilinear, successive essential edges are orthogonal to each other and after four reflections the path will usually return to the essential edge where the first reflection occurred (i.e., it cannot be a subpath of a shortest s-t path). The only case in which a long sequence of successive reflections can occur is when the essential edges form a spiral structure (so that each set of four reflections brings the path to a parallel essential edge that is in the next level of the spiral).

The starting point x is included only in the first copy of the critical graph and we include a copy of it, x', in the last copy of the critical graph only. We index the essential edges and the corresponding copies of the critical graph in the order that they are first visited in a clockwise scan of the boundary of the polygon starting from x. Let $\mathcal{Q}$ be the resulting graph. We assign weights (lengths) to the edges of $\mathcal{Q}$ equal to the length of the corresponding paths in the polygon. Then, a shortest path connecting x to x' in $\mathcal{Q}$ corresponds to a least-risk watchman route in the original polygon.

Theorem 6.2 *A least-risk watchman route through a specified point x on the boundary of a simple rectilinear polygon with k threats can be constructed in time $O(k^2n^3)$.*

Proof: The graph $\mathcal{Q}$ consists of $O(n)$ copies of the critical graph $\mathcal{G}$. The connections between copies i and $i+1$ involve the i^{th} essential edge. Connections between copies i and $i+j$ $(j>1)$ involve all the essential edges with indices in the range $[i, i+j-1]$. This forces all paths from x to x' to come in contact with each essential edge at least once. Thus, the route R, obtained from the shortest path in $\mathcal{Q}$, will contact all of the essential edges; i.e., it is a valid watchman route. Note that some essential edges may be visited many times. In such cases, we consider the "official" visit to be the one corresponding to a connection between the appropriate copies of the critical graph.

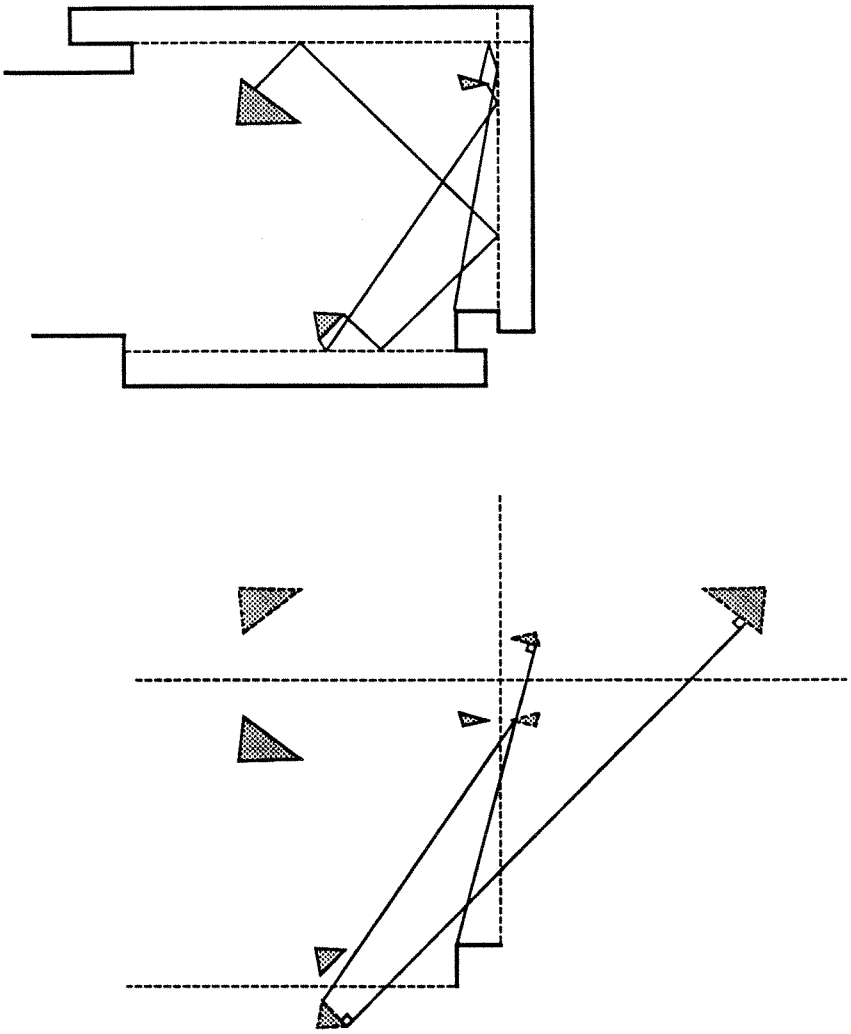


Figure 25. Multiple reflection connections along paths between 0-regions.

To show that R is a shortest watchman route, suppose that there exists a shorter watchman route through x, say R'. If there is a path in $\mathcal{Q}$ that corresponds to R', we have a contradiction since R corresponds to a shortest path in $\mathcal{Q}$. Otherwise, we must have that R' does not (officially) visit the essential edges in the specified order. (The corresponding path "skips" one or more copies of the critical graph. Note that R' can still be a watchman route because it can visit the skipped essential edges unofficially, i.e., on the way to an official visit). But this would imply that R' crosses itself, contradicting Lemma 6.1.

Each copy of the critical graph has $O(kn)$ vertices (from Lemma 5.4 and making the reasonable assumption that k is at worse $O(n)$). The number of connections between

two successive copies is at most $O(k^2n^2)$, and these connections can be constructed by reflecting the whole polygon with respect to each essential edge and using the methods described in section 2. The number of connections representing multiple reflection paths is also at most $O(k^2n^2)$. (If there is no spiral structure, we can have at worst all pairs of reflex vertices and 0-regions connected with paths that make up to three reflections. If there is a spiral structure, the reflex vertices and 0-regions are distributed along the spiral, so the possible first ant last reflections for each pair are again limited to a small constant.) Then, the graph $\mathcal{Q}$ will have $O(kn^2)$ vertices and $O(k^2n^3)$ edges. Using the result of [FT] to run Dijkstra's algorithm, we can construct the shortest path in time $O(k^2n^3)$. ∎

7. Conclusion

Several directions of research are suggested by our results:

1) An important open problem is to examine the special case of the weighted region problem with weights in the set $\{1, 2\}$ (or any two non-zero finite weights). It seems that this special case may be just as hard as the general weighted region problem.

2) Can the complexity of the shortest path problem among 0-regions be improved to something subquadratic?

3) Our results here have applications to the maximum flow problem in the continuum [Mi5]. In particular, our shortest path algorithm can be used to find minimum-capacity cuts in polygons with holes.

4) The approach taken in this paper was to find shortest paths by means of discretizing the problem and performing a search on a shortest paths preserving graph. An alternative approach would be to apply the continuous Dijkstra paradigm to build the single-source shortest path map (or the multi-source Voronoi diagram). This approach is applied to the special case of 0/1 weights in the paper [Mi5], where the connection with maximum flows is discussed.

5) Can we find a way to search critical graphs in the presence of roads in quadratic time? The bottleneck at the moment is in the fact that there are a quadratic number of critical points along road edges.

6) Curved roads could be allowed. Then, the local optimality condition at an interior road point y must be applied as if the boundary at y is the straight line tangent.

7) Can we exploit further the special structure present in the 0-regions that are generated by threats? See [Mi4] for preliminary work on some special cases in which least-risk paths can be found efficiently.

8) We could allow the roads, obstacles, and 0-regions to reside on a nonplanar surface (e.g., the surface of a polyhedron). A generalization to the algorithm of [MMP] to handle the case of roads and 0-regions seems possible.

9) In [CN1], it is shown that, if a starting point for the watchman route problem in simple rectilinear polygons is not specified, a point that lies on a shortest watchman route can be found without affecting the complexity of the algorithm. It would be interesting to see if this can be done for least-risk watchman routes in the presence of 0-regions. Can we find a point that lies on a least-risk watchman route in time $O(k^2n^3)$?

10) Another important open problem is the least-risk watchman route problem in simple polygons. A polynomial-time ($O(n^4 \log n)$) algorithm for the standard case is given in [CN2]. This algorithm constructs an initial watchman route and then adjusts it to obtain a shortest watchman route. The adjustments are needed because the edges that any watchman route must visit may intersect and it is not clear which segment in each edge should be used by a shortest route. The polynomial-time algorithm in [CN2] is based on establishing a uniform way in which we can adjust the route and get closer to the optimum solution. It is not clear that this uniformity in the adjustment sequence carries over in the presence of 0-regions.

Remark: The special case of the $0/1/\infty$-WRP in which there are precisely two simple polygonal regions and no roads has recently (and independently) been solved in $O(n^2)$ time in [AAI]. Also, recent work on the problem of shortest paths in the presence of linear features appears in [Row], where several examples, algorithms, and heuristics are discussed.

Acknowledgement

J. Mitchell is partially supported by by National Science Foundation grants IRI-8710858 and ECSE-8857642 and by a grant from the Hughes Research Laboratories, Malibu, CA. S. Ntafos is partially supported by a grant from Texas Instruments, Inc.

8. References

[AAI] T. Asano, T. Asano, and H. Imai, "Shortest Path Between Two Simple Polygons", *Information Processing Letters*, **24**, 1987, pp. 285-288.

[AAGHI] T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai, "Visibility of Disjoint Polygons", *Algorithmica*, Vol. 1, (1986), pp. 49-63.

[CR] J. Canny and J. Reif, "New Lower Bound Techniques for Robot Motion Planning Problems", *Proc. 28th FOCS*, pp. 49-60, Oct. 1987.

[CN1] W.P. Chin and S. Ntafos, "Optimum Watchman Routes", *Information Processing Letters*, **28**, pp. 39-44, May 1988.

[CN2] W.P. Chin and S. Ntafos, "Watchman Routes in Simple Polygons", Technical Report, Computer Science, University of Texas at Dallas, 1987. To appear: *Discrete and Computational Geometry.*

[Ed] H. Edelsbrunner, Private communication, 1988.

[EG] H. Edelsbrunner and L. Guibas, "Topologically Sweeping an Arrangement", In *Proc. 18th Ann. ACM Sympos. Theory Comput.* (1986), pp. 389-403.

[EA] H.A. El Gindy and D. Avis, "A Linear Algorithm for Computing the Visibility Polygon From a Point", *Journal of Algorithms*, Vol. 2 (1981), pp. 186-197.

[FT] M. Fredman and R. Tarjan, "Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms", *Proc. 25th Annual IEEE Symposium on Foundations of Computer Science*, pp. 338-346, 1984.

[GMN] L. Gewali, A. Meng, and S. Ntafos, "Low Risk Path Planning", Manuscript, Computer Science Program, The University of Texas at Dallas, 1987.

[GMMN] L. Gewali, A. Meng, J.S.B. Mitchell, and S. Ntafos, 'Path Planning in $0/1/\infty$ Weighted Regions With Applications (Extended Abstract)", *Proceedings of the Fourth Annual ACM Symposium on Computational Geometry*, Champaign-Urbana, IL, June 6-8, pp. 266-278.

[GM] S.K. Ghosh and D.M. Mount, "An Output Sensitive Algorithm for Computing Visibility Graphs", Technical Report CS-TR-1874, Department of Computer Science, University of Maryland, July 1987. (Also appears in FOCS, 1987.)

[KM] S. Kapoor and S.N. Maheshwari, "Efficient Algorithms for Euclidean Shortest Path and Visibility Problems with Polygonal Obstacles", *Proc. Fourth Annual ACM Symposium on Computational Geometry*, Urbana-Champaign, IL, June 6-8, 1988, pp. 172-182.

[Le1] D.T. Lee, "Proximity and Reachability in the Plane", Ph.D. Thesis, Technical Report ACT-12, Coordinated Science Laboratory, University of Illinois, Nov. 1978.

[Le2] D.T. Lee, "Visibility of a Simple Polygon", *Computer Vision, Graphics, and Image Processing*, Vol. 22 (1983), pp. 207-221.

[LC] D.T. Lee and Y.T. Ching, "The Power of Geometric Duality Revisited", *Information Processing Letters*, **21** (1985), pp. 117-122.

[Mi1] J.S.B. Mitchell, "Planning Shortest Paths", PhD Thesis, Department of Operations Research, Stanford University, August, 1986. (Available as Research Report 561, Artificial Intelligence Series, No. 1, Hughes Research Laboratories, Malibu, CA.)

[Mi2] J.S.B. Mitchell, "An Algorithmic Approach to Some Problems in Terrain Navigation", *Arificial Intelligence*, **37** (1988) pp. 171-201.

[Mi3] J.S.B. Mitchell, "Shortest Paths Among Obstacles, Zero-Cost Regions, and Roads", Technical Report No. 764, Department of Operations Research, Cornell University, 1987.

[Mi4] J.S.B. Mitchell, "On the Maximum Concealment Problem", Manuscript in preparation, School of Operations Research and Industrial Engineering, Cornell University.

[Mi5] J.S.B. Mitchell, "On Maximum Flows in Polyhedral Domains", Extended abstract appears in *Proc. Fourth Annual ACM Symposium on Computational Geometry*,

Urbana-Champaign, June 6-8, 1988, pp. 341-351. Full paper to appear in *Journal of Computer and System Sciences*, 1989.

[Mi6] J.S.B. Mitchell, "A New Algorithm for Shortest Paths Among Obstacles in the Plane", Technical Report No. 832, School of Operations Research and Industrial Engineering, Cornell University, October, 1988.

[MMP] J.S.B. Mitchell, D.M. Mount, and C.H. Papadimitriou, "The Discrete Geodesic Problem", *SIAM Journal on Computing*, **16**, No. 4, pp. 647-668, August, 1987.

[MP] J.S.B. Mitchell and C.H. Papadimitriou, "The Weighted Region Problem", Technical Report, Department of Operations Research, Stanford University, 1985. Extended abstract appears in *Proc. Third Annual ACM Conference on Computational Geometry*, Waterloo, June 1987. Full appear to appear in *Journal of the ACM.*

[Nt] S. Ntafos, "The Robber Route Problem", Technical Report, Computer Science, University of Texas at Dallas, 1988.

[O'R] J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, 1987.

[PS] F.P. Preparata and M.I. Shamos, *Computational Geometry*, Springer-Verlag, New York, Berlin, Heidelberg, Tokyo, 1985.

[RS] J.H. Reif and J.A. Storer, "Shortest Paths in Euclidean Space with Polyhedral Obstacles", Technical Report CS-85-121, Computer Science Department, Brandeis University, April, 1985.

[Roh] H. Rohnert, "Shortest Paths in the Plane with Convex Polygonal Obstacles", Technical Report A 85/06, University of Saarbrücken, West Germany, 1985.

[Row] N.C. Rowe, "Roads, Rivers, and Rocks: Optimal Two-Dimensional Route Planning Around Linear Features for a Mobile Robot", to appear in *International Journal of Robotics Research.* Also available as Technical Report NPS52-87-027, Department of Computer Science, U.S. Naval Postgraduate School, June 1987.

[SS] M. Sharir and A. Schorr, "On Shortest Paths in Polyhedral Spaces", *SIAM Journal of Computing*, Vol. 15, No. 1, pp. 193-215, February 1986.

[We] E. Welzl, "Constructing the Visibility Graph for n Line Segments in $O(n^2)$ Time", *Information Processing Letters*, Vol. 20 (1985), pp. 167-171.