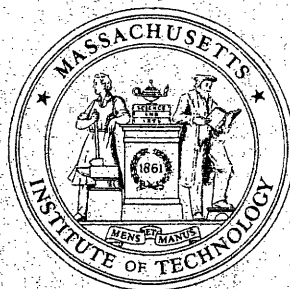# OPERATIONS RESEARCH CENTER

working paper

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

FINDING MINIMUM RECTILINEAR DISTANCE PATHS
IN THE PRESENCE OF BARRIERS

by

Richard C. Larson

and

Victor O.K. Li

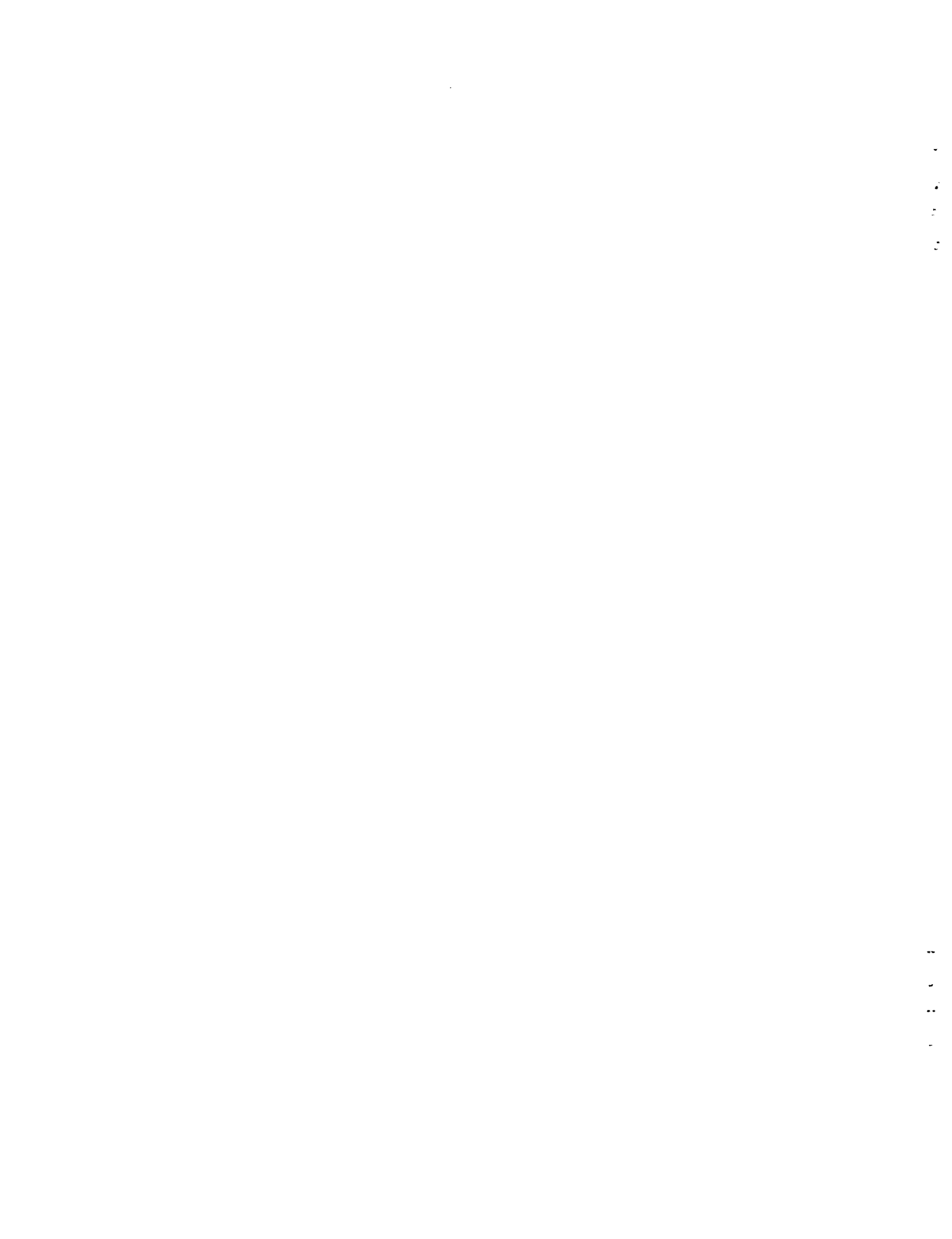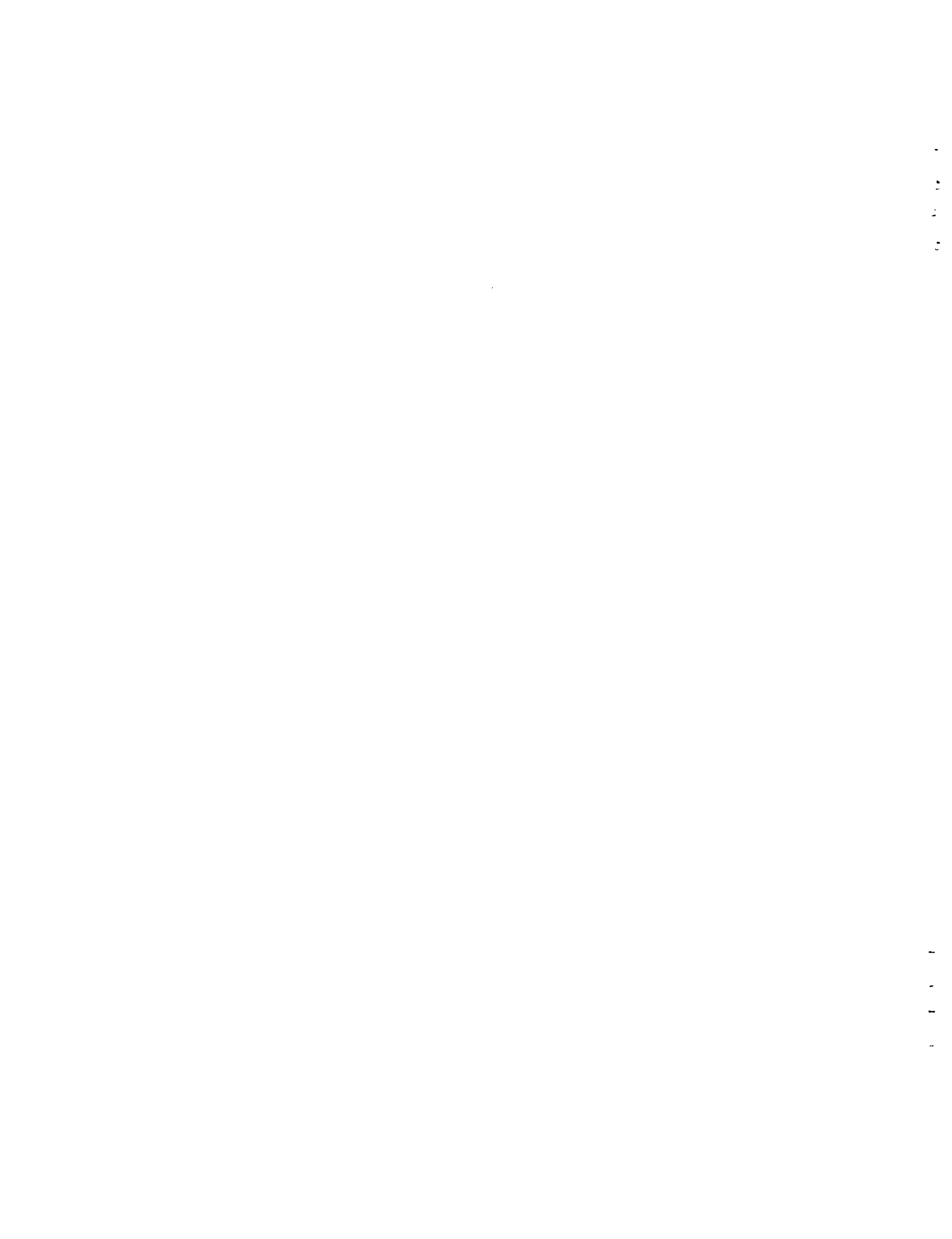OR 088-79   May 1979

Table of Contents

Abstract

List of Figures

List of Tables

ABSTRACT

Given a set of origin-destination points in the plane and a set of polygonal barriers to travel, this paper develops an efficient algorithm for finding minimal distance feasible paths between the points, assuming that all travel occurs according to the rectilinear distance metric. By geometrical arguments the problem is reduced to a finite network problem. The nodes are the origin-destination points and the barrier vertices. The links designate those node pairs that "communicate" in a simple way, where communication implies the existence of a node-to-node rectilinear path that is not made longer by the barriers. The weight of each link is the rectilinear distance between its two corresponding nodes.

Solution of the minimal distance path problem on the network procedes in two steps. First, for a given origin or root node, a tree is generated containing a minimal distance path to each node that communicates with the root node. Second, a modified Dijkstra-type iteration is utilized, starting with the nodes of the tree, sequentially adding nodes according to minimum "penalty distance", where the penalty is the extra travel distance caused by the barriers. The paper concludes with a discussion of the computational complexity of the procedure, followed by a numerical example.

Vaccaro [1] determined the shortest Euclidean (straight line) distance between points on a plane with barriers represented by line segments. He proved that all shortest paths must pass through end points of the barriers. The algorithm Vaccaro proposed is an exhaustive enumeration of all possible paths passing through the end points of the barriers. More recently, Lozano-Perez [2] described the VGRAPH algorithm for finding the shortest Euclidean distance between two points on a plane. This algorithm is similar to Vaccaro's algorithm, except that barriers are represented by convex polygons rather than line segments. The VGRAPH is defined thusly: the node set N consists of the vertices of the barriers and the origin and destination points; the link set consists of all branches between the nodes such that a straight line connecting the ith element of N to the jth does not intersect any barrier. The resulting graph is called the visibility graph (VGRAPH) of N since connected vertices in the graph can "see" each other. The VGRAPH algorithm was used for navigating SHAKEY, an early robot vehicle, described in [3]. However, this algorithm works only for Euclidean distances and convex barriers.

The paper procedes in three parts. In I we prove results necessary to transform the problem in the x-y plane, with a noncountably infinite number of paths, to a finite network problem. In II we develop the details of the algorithm, POLYPATH, and discuss its computational complexity. In III we present a numerical example. For convenience, Table I contains a glossary of nonstandard terms and Table II contains most of the symbol definitions used in the paper.

The rectilinear distance between two points (x_1, y_1), (x_2, y_2) is $|x_1 - x_2| + |y_1 - y_2|$. This distance is sometimes called right-angle, L_1, or Manhattan distance . This paper develops an algorithm for finding minimum rectilinear distance paths between given origin-destination points, in the presence of polygonal barriers to travel. The barriers need have no special properties, such as convexity.

The problem arises in a number of areas:

1. Urban transportation. Urban vehicles travelling on a street grid may be considered to be governed by the right angle metric. Barriers to travel could include cemeteries, parks, railroad yards, rivers, etc. Effective operation of these systems would require knowledge of "best paths" around such barriers. (Ironically, the motivation for this paper originated with transportation work by these authors in Manhattan, which revealed the inadequacy of the "Manhattan distance metric" as a measure of travel distance between two points due, primarily, to New York's Central Park.)

2. Plant and facility layout. Travel on a plant or factory floor can often be modelled as right-angle. Barriers to travel would be impassable areas on the floor, such as sub assembly areas, shops, sections of assembly lines, etc. The algorithm proposed here not only could reveal optimal paths between points, but also could be used to examine the plant floor travel-time consequences of alternative floor layouts.

3. Locating power lines. In sections of the midwest of the U.S. new power lines are only allowed to be located along the east-west or north-south boundaries between surveyed mile-square or quarter-mile square sections. The location of such a power line occurs, then, along a right-angle route. Barriers in this problem could include towns, highways, parks, or subsections otherwise not available for power line routing.

Other potential applications include design of printed circuit boards, certain cutting problems, and finding the path through a maze (perhaps for a robot).

We are aware of no other work on this problem. The closest work pertains to the analogous problem using the Euclidean metric. In 1974

Table I

Glossary of Nonstandard Terms

Barrier	A polygon impenetrable to travel
Communicate	Two points communicate if the minimal travel distance between them is not increased by the barriers.
Simply communicating	Two points are simply communicating if they communicate according to at least one of three specified criteria.
Root vertex	The vertex in a communicating tree with which all other points in the tree communicate.
Rectilinear path	A connected sequence of horizontal and vertical straight-line segments beginning at an origin point and terminating at a destination point; an allowable route of travel between origin and destination.
Subpath	A connected subset of a path.
Staircase path	A rectilinear path whose length between origin and destination is equal to the rectilinear distance between them.
Feasible path	A rectilinear path intersecting no barriers.
Path step	A horizontal or vertical straight line segment in a rectilinear path.
Turning step	A path step terminating one staircase subpath and starting another.
Path vertex	The point of termination of one path step and the beginning of another (except at origin or destination, where only one applies).
Predecessor link	The link immediately preceding a node in a specific path.

Table II

Table of Symbols
(in approximate order of usage)

n_Q Number of origin-destination points

n_B Number of barriers

$q(i)$ The ith origin-destination point

Q Set of origin-destination points
$(=\{q(i), i = 1,2,\ldots,n_Q\})$

$v(k,m)$ The kth clockwise-ordered vertex of barrier m

v_m Number of vertices of barrier m

V Set of barrier vertices $(=\{v(k,m),$
$k = 1,2,\ldots,v_n; \quad m = 1,2,\ldots,n_B\})$

$\{x(i), y(i)\}$ Coordinates of $q(i)$

$\{x(k,m), y(k,m)\}$ Coordinates of $v(k,m)$

P A rectilinear path in Euclidean space R^2.

$L(P)$ Length of path P

s $2(s + 1)$ is the number of steps in a path P.

$V(P)$ Sequence of path vertices in P, starting at the origin (x_o, y_o).

$[\{x_k\},\{y_k\}]$ Set of x and y coordinates of path vertices in a path P

W Set of origin-destination points and barriers vertices $(=Q \cup V)$

n_w Number of points in the set W
$$(= n_Q + \sum_{n=1}^{n_B} v_n)$$

$w(i)$ The ith point contained in W

$(x_w(i), y_w(i))$	Coordinates of $w(i)$
c_{ij}	$\|x_w(i) - x_w(j)\| + \|y_w(i) - y_w(j)\|$
P_{ij}	A (not necessarily optimal) path from $w(i)$ to $w(j)$
P^*_{ij}	An optimal (minimal distance) path from $w(i)$ to $w(j)$
$r_{x+}(w(i))$	The positive x probe from $w(i)$ [defined precisely in text] [Also $r_{x-}(w(i))$, $r_{y+}(w(i))$, $r_{y-}(w(i))$].
$VST(w(i))$	Vertex seeking tree of $w(i)$ [= a union of the four probes about $w(i)$].
$t((x,\dot{y}), (x,y))$	Minimal feasible rectilinear travel distance between $(x,\dot{y})$ and (x', y').
$t(i,j)$	Minimal feasible rectilinear travel distance between $w(i)$ and $w(j)$
$G(W,L)$	Network (graph) having node set W and link set L
$\ell(i,j)$	Link connecting $w(i)$ and $w(j)$
$D=(d_{ij})$	Matrix of link lengths, where d_{ij} is the rectilinear length of $\ell(i,j)$. ($d_{ij} = +\infty$ if link $\ell(i,j)$ does not exist.)
S	Set of simply communicating vertex pairs
$F(w(i))$	A tree containing all nodes communicating with $w(i)$.
$\Phi = (\phi_{ij})$	Matrix of node orientations ($\phi_{ij} = 1,2,..,8$).
$\Gamma(\phi_{ij})$	Set of orientations about node $w(j)$ that may include staircase paths through $w(j)$ to $w(i)$.
$\Delta_n(w(i))$	Set of node-link pairs in which each node is n-communicating with $w(i)$ and the corresponding link is that node's predecessor link

$\varepsilon(i,j)$ — The penalty distance caused by the barriers in a minimal distance path P^*_{ij}

$H_1(i|m)$ — The set of m least penalty nodes for travel to $w(i)$. [The "closed" nodes]

$H_2(i|m)$ — The set of n_w - m maximum penalty nodes for travel to $w(i)$ $(= N - H_1(i|m))$ [The "open" nodes]

$\hat{\varepsilon}(i,j|m)$ — Minimum simply communicating distance from some node $w(j) \in H_2(i|m)$ to any node contained in $H_1(i|m)$

$\gamma(i|m)$ — Set of pairs of nodes and predecessor links in the tree of m least penalty nodes for travel to $w(i)$.

I. <u>Transforming to a Network Problem</u>

    <u>Preliminaries</u>

We consider Euclidean 2-space R^2. Within R^2 there are n_Q origin-destination points

$$Q = \{q(i), \ i = 1, 2, \ldots, n_Q\}$$

and n_B polygonal barriers B_m ($m = 1, 2, \ldots, n_B$) with vertices

$$V = \{v(k,m), \ k = 1, 2, \ldots, v_m; \ m = 1, 2, \ldots, n_B\},$$

where $v(k,m)$ is the kth clockwise-ordered vertex of barrier m and $v_n < \infty$ is the number of its vertices. The coordinates of $q(i)$ are $\{x(i), y(i)\}$ and the coordinates of $v(k,m)$ are $\{x(k,m), y(k,m)\}$. B_m ($m = 1, \ldots, n_B$) is an open set, i.e., the intersection of v_n open half planes. Hence, the vertices and sides of B_m are not contained within B_m.

A rectilinear path P in R^2 having $2(s + 1)$ steps is specified by a sequence of path vertices $V(P) = \{(x_0, y_0), \ (x_1, y_0), \ (x_1, y_1), \ (x_2, y_1),$ $(x_2, y_2), \ldots (x_s, y_s), \ (x_{s+1}, y_s), \ (x_{s+1}, y_{s+1})\} = [\{x\}, \{y\}]$, $s = 0, 1, 2, \ldots,$ such that

$$P = \{(x, y_k) \ \varepsilon \ R^2; \ x_k \le x \le x_{k+1} \ \text{or} \ x_{k+1} \le x \le x_k, k = 0, 1, \ldots, s\}$$
$$\cup \{(x_k, y) \ \varepsilon \ R^2 : y_{k-1} \le y \le y_k \ \text{or} \ y_k \le y \le y_{k-1}, \ k = 1, 2, \ldots, s+1\}.$$

That is, a path is a connected set of points proceeding in straight line sequences horizontally from (x_0, y_0) to (x_1, y_0), then vertically to (x_1, y_1), then horizontally to (x_2, y_1), etc. To allow the first step to be vertical, simply set $x_1 = x_0$. In our development we consider the merged set of points

$W = Q \cup V$. A path P_{ij} from $w(i) \in W$ to $w(j) \in W$ is specified by a set of path vertices orginating at $(x_0, y_0) = (x_w(i), y_w(i))$ and terminating at $(x_{s+1}, y_{s+1}) = (x_w(j), y_w(j))$. A _feasible_ path from $w(i)$ to $w(j)$ penetrates no barriers, hence satisfies the condition $P_{ij} \cap [\bigcup_{m=1}^{n} B_m] = \phi =$ null set. The rectilinear length (or simply length) of path P is

$$L(P) = L_x(P) + L_y(P), \text{ where}$$

$$L_x(P) \equiv \sum_{k=0}^{s} |x_{k+1} - x_k| \text{ and } L_y(P) \equiv \sum_{k=0}^{s} |y_{k+1} - y_k|.$$

The minimum rectilinear travel distance between two points (x,y) and (x', y') is $t([x,y], [x', y'])$, obtained by following a shortest length feasible path between the points.

The problem is as follows: For any given origin $q(i)$ find a minimal length feasible path and the corresponding travel distance to each possible destination $q(j)$. Repeated application over i of any solution algorithm yields minimal length feasible paths between all origin-destination pairs.

Two points $w(i) \in W$, $w(j) \in W$, having coordinates $(x_w(i), y_w(i))$ and $(x_w(j), y_w(j))$, are said to _communicate_ if there exists at least one feasible path P_{ij} between them having length $L(P_{ij}) = |x_w(j) - x_w(i)| + |y_w(j) - y_w(i)| \equiv c_{ij}$. That is, two points communicate if the barriers cause no net increase in travel distance between them. The minimal feasible travel distance between $w(i)$ and $w(j)$ is abbreviated $t(i,j)$ $[= t([x_w(i), y_w(i)], [x_w(j), y_w(j)])]$.

A path P having path vertices $V(P) = \{(x_0, y_0), (x_1, y_0), \ldots, (x_{s+1}, y_{s+1})\}$ is a _staircase path_ having _staircase vertices_ if

$\{x_k, \{k=0,1,\ldots,s+1\}$ and $\{y_k, k=0,1,\ldots,s+1\}$ each form monotone sequences, i.e., if sgn $(x_{k+1} - x_k)$ = sgn $(x_k - x_{k-1})$ and sgn $(y_{k+1} - y_k)$ = sgn$(y_k - y_{k-1})$, k=1,2,..,s. [sgn(0) is arbitrary].

Lemma 1: Two points having coordinates (x_0, y_0) and (x_{s+1}, y_{s+1}) communicate if and only if there exists at least one feasible staircase path between them.

Proof: (1) Sufficency: If for some path P having path vertices $V(P) = [\{x_k\}, \{y_k\}]$ $\{x_k\}$ and $\{y_k\}$ form monotone sequences, then for all k=1,...,s, sgn$(x_{k+1} - x_k)$ = sgn$(x_k - x_{k-1})$ and sgn$(y_{k+1} - y_k)$ = sgn$(y_k - y_{k-1})$. Hence L(P) =

$|\overset{s}{\underset{k=0}{\Sigma}} x_{k+1} - x_k| + |\overset{s}{\underset{k=0}{\Sigma}} y_{k+1} - y_k| = |x_{s+1} - x_0| + |y_{s+1} - y_0|$.

(2) Necessity: Suppose $\{x_k\}$ is a nonmonotone sequence. Let $\mathcal{Q}_1 = \{x_k: x_{k+1} - x_k \geq 0\}$, $\mathcal{Q}_2 = \{x_k: x_{k+1} - x_k < 0\}$. Suppose further that $x_{s+1} > x_0$. Then $L_x(P) = \underset{k \in \mathcal{Q}_1}{\Sigma}(x_{k+1} - x_k) -$

$\underset{k \in \mathcal{Q}_2}{\Sigma}(x_{k+1} - x_k) = \overset{s}{\underset{k=0}{\Sigma}}(x_{k+1} - x_k) - 2\underset{k \in \mathcal{Q}_2}{\Sigma}(x_{k+1} - x_k) =$

$(x_{s+1} - x_0) - 2\underset{k \in \mathcal{Q}_2}{\Sigma}(x_{k+1} - x_k) > x_{s+1} - x_0$, a contradiction.

A similar argument holds for the case $x_{s+1} < x_0$. ∎

Remark 1: Any path having minimal length joining two communicating points must be a staircase path.

Remark 2: Any two points, joined by a straight line segment that has no points in common with any barrier, communicate.

The second remark can be proved by assuming that the closest barrier to the line segment contains points no closer than $\varepsilon > 0$ to the segment and constructing a sequence of path vertices in a staircase manner such that the staircase is always less than $\varepsilon/2$ from the line segment. (See Figure 1). Note, as a consequence of this remark, all pairs of adjacent vertices of a barrier communicate.

Vertex-Seeking Tree

We now wish to construct a simple tree about each node which provides a basis for path formation. First consider a point $q(i) \in Q$ having coordinates $\{x(i), y(i)\}$. We wish to construct the "positive x probe" $r_{x+}(q(i))$ from $q(i)$. To do this, extend a ray from $(x(i), y(i))$ horizontally in the direction of increasing x. There are three possibilities (see Figure 2):

1) The ray intersects no barriers, in which case

$$r_{x+}(q(i)) = \{(x,y) \in R^2 : y = y(i), x \geq x(i))\} \text{ i.e., } r_{x+}(q(i)) \text{ is the ray.}$$

2) The ray intersects another point $w(j) \in W$ (either a barrier vertex or an origin-destination point), in which case $r_{x+}(q(i)) = \{(x,y) \in R^2 : y = y(i), x(i) \leq x < x_w(j)\}$; i.e., $r_{x+}(q(i))$ is the line segment connecting $q(i)$ and $w(j)$.

3) The ray intersects side k of a barrier m, the point of intersection being $\underline{a}$. The barrier side is thereby partitioned into two subsides having endpoints $(v(k,m), \underline{a})$ and $(\underline{a}, v(k+1,m))$. Here $r_{x+}(q(i))$ is the union of the line segment from $q(i)$ to $\underline{a}$ and that
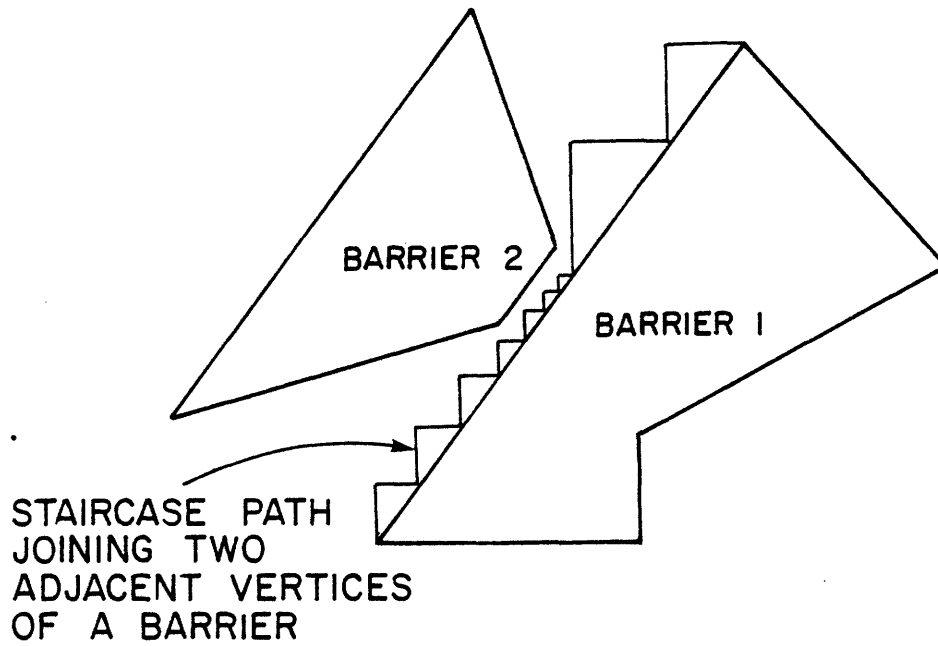
BARRIER 2

BARRIER 1

STAIRCASE PATH
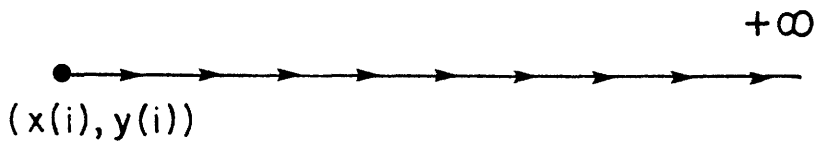JOINING TWO
ADJACENT VERTICES
OF A BARRIER

Illustration of Staircase Path Between Two Adjacent Vertices of
a Barrier

Figure 1

$+\infty$

(a) RAY INTERSECTS NO BARRIERS

BARRIER VERTEX

(x(i),y(i))

ANOTHER ORIGIN-DESTINATION POINT

(x(i),y(i))

(b) RAY INTERSECTS ANOTHER POINT

v(k,m)

$\underline{a}$

(x(i),y(i))

v(k+1,m)

RIGHT ANGLE v(k,m)

$\underline{a}$

(x(i),y(i))

v(k+1,m)
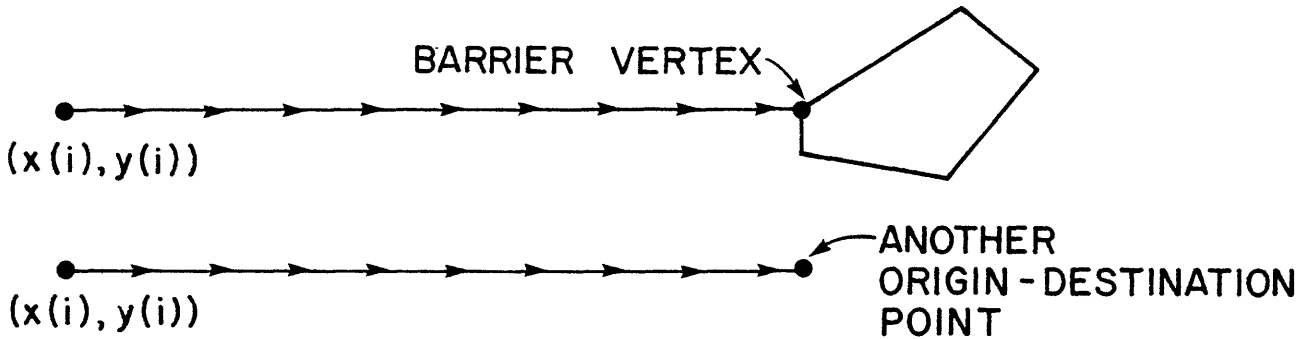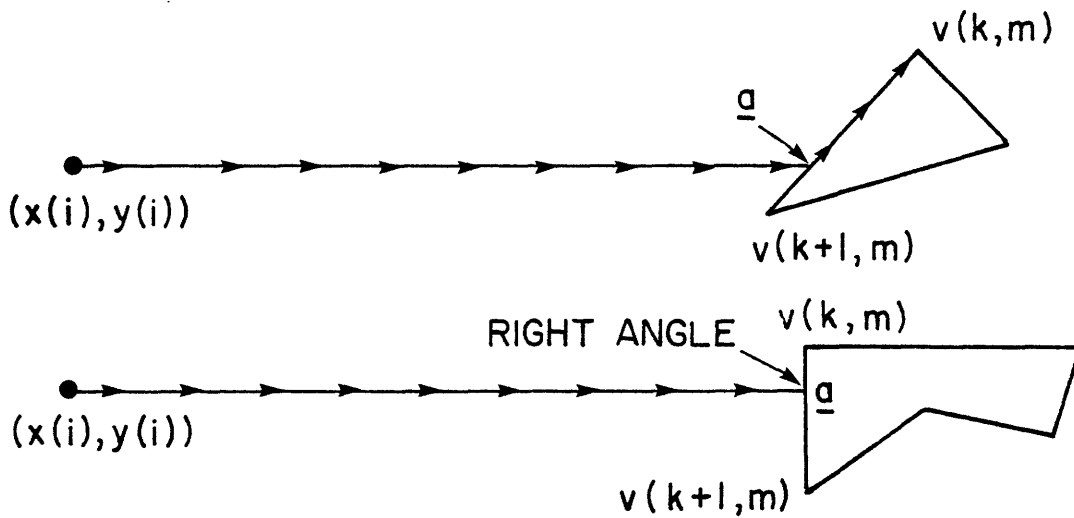
(c) RAY INTERSECTS A BARRIER SIDE

KEY: $= r_{x+}(q(i)) = $ POSITIVE x PROBE

(x(i),y(i))

The Positive X Probe: Examples

Figure 2

subside which lies at an obtuse angle to the segment from q(i)

to $\underline{a}$. (In the case of right angle intersection $r_{x+}(q(i))$ need not

have this second component.)

In a like manner we can construct probes $r_{y+}(q(i))$, $r_{y-}(q(i))$, and

$r_{x-}(q(i))$ in the positive y direction, negative y direction, and negative

x direction, respectively. The <u>vertex-seeking tree</u> about the <u>root node</u>

q(i) is

$$VST\ (q(i)) \equiv r_{x+}(q(i)) \cup r_{x-}(q(i)) \cup r_{y+}(q(i)) \cup r_{y-}(q(i)).$$

By construction of the tree, it is clear that for all $(x,y) \in VST\ (q(i))$

$t([x,y],\ [x(i),\ y(i)]) = |x - x(i)| + |y - y(i)|$, i.e., all

points on the tree communicate with q(i).

Now consider a barrier vertex $v(k,m) \in V$. A similar vertex-seeking

tree is constructed for v(k,m), with the obvious condition that no point

in the tree can penetrate barrier m. Thus, depending on the orientation

of the barrier about v(k,m), the number of feasible probes about v(k,m)

may be 4, 3, 2, 1, or even 0. (See Figure 3), [In the case of zero feasible

probes, any feasible path to the vertex must contain a countably infinite

number of steps, and the argument supporting Remark 2 must be modified

accordingly.]

<u>Lemma 2:</u> Suppose for two points $w(i) \in W$ and $w(j) \in W$, an x-probe from one

intersects a y-probe from the other at some point $\underline{b} \notin W$. Then

w(i) and w(j) communicate.

Proof: We prove the result for $x_w(i) \leq x_w(j)$, $y_w(i) \leq y_w(j)$ and $r_{x+}(w(i))$

$\cap r_{y-}(w(j)) \neq \phi$. All other cases are proved in the same manner.
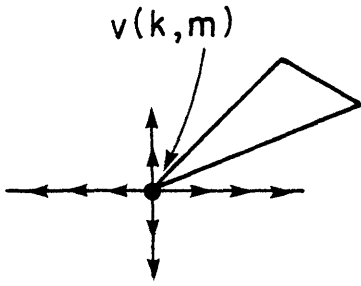
(a) FOUR PROBES POSSIBLE
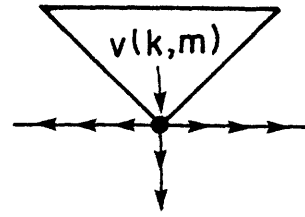
(b) THREE PROBES POSSIBLE

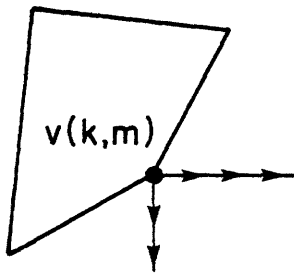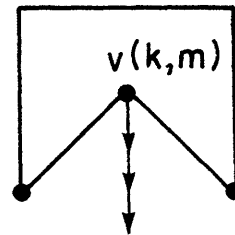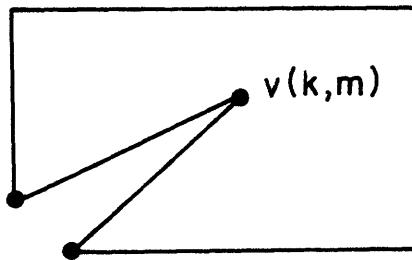(c) TWO PROBES POSSIBLE

(d) ONE PROBE POSSIBLE

(e) ZERO PROBES POSSIBLE

Possible Vertex - Seeking Trees from a Barrier Vertex

Figure 3

Suppose there exists $\underline{b} \notin W$ such that $\underline{b} = \{x_b, y_b\} \in r_{x+}(w(i))$ $\cap r_{y-}(w(j))$, i.e., $\underline{b}$ is an intersection point. Then $y_b \geq y_w(i)$, for otherwise y_b, by construction of $r_{x+}(w(i))$, would be on a barrier side not allowing intersection from above. Similarly, $x_b \leq x_w(j)$, for otherwise x_b would be on a barrier side not allowing intersection from the left. We also must have $y_b \leq y_w(j)$, for all points (x,y) in a negative y probe from $w(j)$ must fall on or below $w(j)$, i.e., $y \leq y_w(j)$. Similarly $x_b \geq x_w(i)$. By the communicating property of probes.

$$t(q(i),\underline{b}) = (x_b - x_w(i)) + (y_b - y_w(i))$$

$$t(\underline{b},q(j)) = (x_w(j) - x_b) + (y_w(i) - y_b).$$

Invoking the triangle inequality,

$$t(q(i),q(j)) \leq t(q(i),\underline{b}) + t(\underline{b},q(j)) = (x_w(j) - x_w(i))$$

$+ (y_w(j) - y_w(i))$, implying that $q(i)$ and $q(j)$ communicate. ■

We have now identified three conditions under which two vertices $w(i)$ and $w(j)$ communicate:

(1) If $w(i)$ and $w(j)$ are adjacent vertices of a barrier.

(2) If $w(i)$ is the root and $w(j)$ is a terminal point of a vertex-seeking tree.

(3) If $w(i)$ and $w(j)$ have x and y probes that intersect at a point other than a vertex $w(k)$.

Each pair of vertices $w(i)$ and $w(j)$ that satisfies at least one of the three conditions above is called _simply communicating_.

Vertex Following Paths

Theorem 1. Suppose two points $w(i) \in W$, $w(j) \in W$ communicate, but do not

communicate simply. Then there exists at least one feasible staircase

path P_{ij}^* between them containing a sequence of simply communicating

vertices $\{w(i), w(k_1^*), \ldots, w(k_n^*), w(j)\}$.

Proof: Consider the case $x_w(j) \geq x_w(i)$, $y_w(j) \geq y_w(i)$. We demonstrate

the existence of $w(k_1^*)$; the remaining $w(k_\ell^*)$ are obtained by

induction ($\ell = 2, 3, \ldots, n$). Since $w(i)$ and $w(j)$ communicate,

there exists at least one feasible staircase path P_{ij} between

them, with path vertices $V(P_{ij}) = \{(x_w(i) = x_0, y_w(i) = y_0),$

$(x_1, y_0), (x_1, y_1), \ldots, (x_s, y_s), (x_w(j) = x_{s+1}, y_s), (x_{s+1}, y_w(j)$

$= y_{s+1})\}$. By the staircase property, $x_{k+1} \geq x_k$, $y_{k+1} \geq y_k$,

$k = 0, 1, \ldots, s$. Invoking communication, $L(P_{ij}) = (x_w(j) - x_w(i))$

$+ (y_w(j) - y_w(i))$.

We now construct an alternative equal length feasible

staircase path from $w(i)$ to $w(j)$. If $x_1 > x_0$, start at x_1.

Otherwise start at y_1 and reverse the role of x and y in what

follows. Beginning at $x = x_1$, create a new path step $[(x, y_0),$

$(x, y_1)]$ and push it as far to the right as possible up to and

including x_2. If x_2 is reached without intersecting a barrier,

continue to push rightward with the amalgamated path step

$[(x, y_0), (x, y_2)]$. Eventually there must be an amalgamated

path step $[(x, y_0), (x, y_k)]$ that intersects a barrier vertex or

a barrier side for some $x = x_1'$, $x_k \leq x_1' < x_{k+1}$. If not, then x

could reach $x_w(j)$, implying that $w(i)$ and $w(j)$ communicate simply, a contradiction. Clearly this path push and amalgamation process yields a path having length equal to $L(P_{ij})$.

Case 1. If the intersection is with a barrier vertex $v(k,m) = \{x(k,m), y(k,m)\}$, we have $x(k,m) = x_1$, and $y(k,m) \geq y_w(i)$. Since $r_{x+}(w(i))$ obviously intersects $r_{y-}(v(k,m))$, $v(k,m) = w(k_1^*)$, i.e., $w(i)$ communicates simply with $v(k,m)$.

If the intersection is not with a barrier vertex, it must be with a barrier side interior point at (x_1',y_0). In this case, continue the step pushing and amalgamation process, with the intercepted barrier side being the step's lower end point.

Case 2. We may reach an endpoint or vertex $v'(k,m)$ of the barrier side, in which case our new path from $w(i)$ to $v'(k,m)$ has reproduced $r_{x+}(w(i))$ and thus $v'(k,m) = w(k_1^*)$; or .

Case 3. We reach a point x_1'' at which the amalgamated step intercepts a barrier vertex $v''(k,m)$; in this case it is clear that $r_{x+}(w(i)) \cap r_{y-}(v''(k,m)) \neq \phi$, and thus $v''(k,m) = w(k_1^*)$.

One continues by induction, generating each $w(k_{\ell+1}^*)$ by repeating the above process from $w(k_\ell^*)$. Eventually $w(j)$ is reached by treating $w(j)$ as a "barrier vertex" in the above procedure. ▨

An illustration of the key ideas in the proof is given in Figure 4.

Suppose two points $w(i) \in W$, $w(j) \in W$ do not communicate. Consider a minimal distance path P_{ij}' between $w(i)$ and $w(j)$, i.e., $L(P_{ij}') \leq L(P_{ij})$ for all feasible P_{ij} The path vertices are $V(P_{ij}') = \{(x_w(i) = x_0, y_w(i) = y_0),$
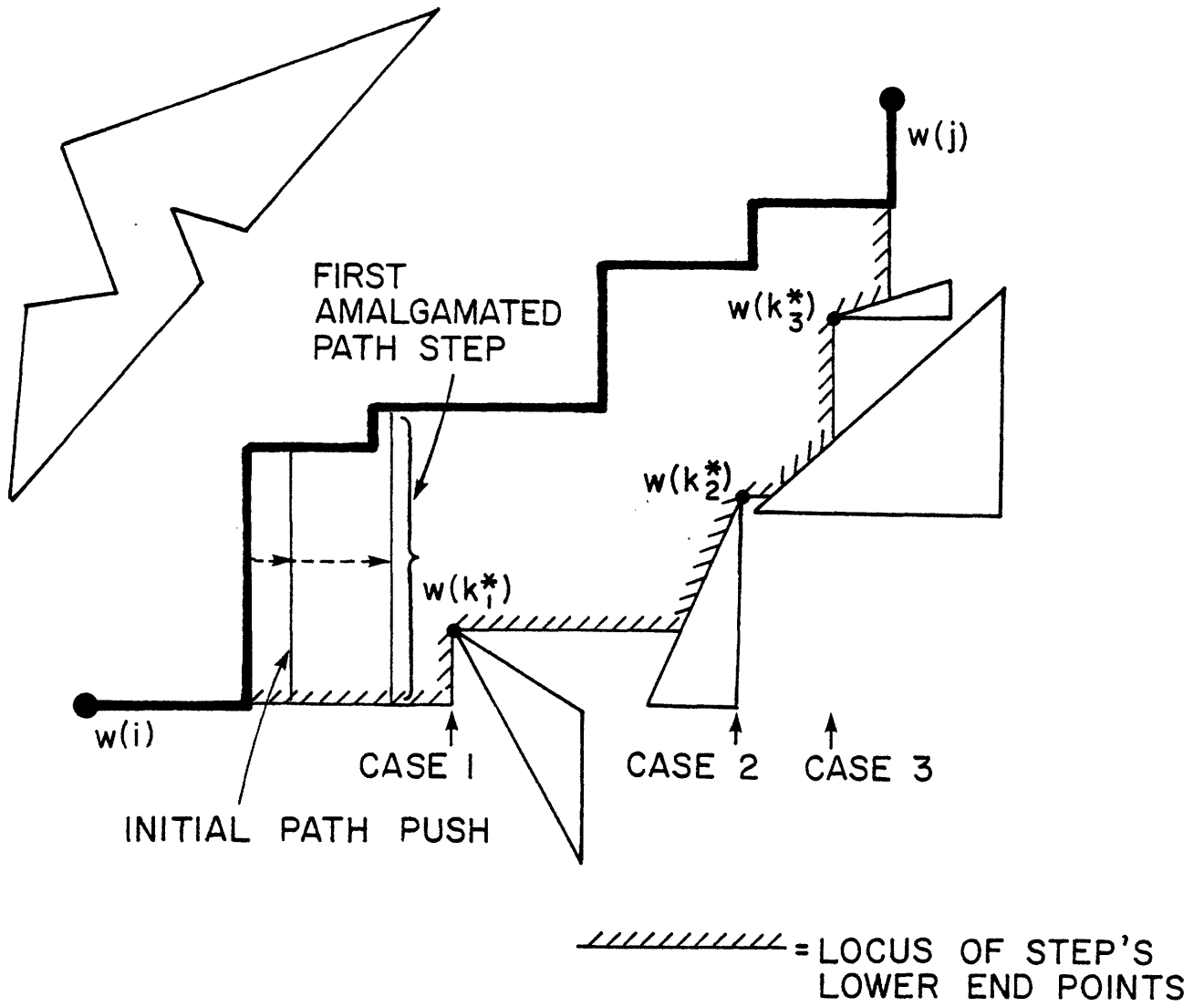
Illustration of the Path Push and Amalgamation Process

Figure 4

(x_1, y_0), (x_1, y_1),...,(x_s, y_s), $(x_w(j) = x_{s+1}, y_s)$, $(x_{s+1}, y_w(j) = y_{s+1})\}$.

Since $w(i)$ and $w(j)$ do not communicate $L(P'_{ij}) > |x_w(i) - x_w(j)| +$ $|y_w(i) - y_w(j)|$. Hence there exists at least one point $x_k \varepsilon \{x_\ell\}$ such that sgn $(x_k - x_{k-1}) \neq$ sgn $(x_{k+1} - x_k)$ or $y_k \varepsilon \{y_\ell\}$ such that sgn $(y_k - y_{k-1}) \neq$ sgn $(y_{k+1} - y_k)$. For any such x_k the path step $[(x_k, y_{k-1}),$ $(x_k, y_k)]$ is called an x <u>turning step</u>. And for any such y_k the path step $[(x_k, y_k), (x_{k+1}, y_k)]$ is called a y <u>turning step</u>. (Figure 5).

<u>Lemma 3</u>:   All turning steps in a minimal distance path intersect a barrier vertex, with the barrier situated in the direction of the turn.

<u>Proof</u>:   Suppose $[(x_k, y_{k-1}), (x_k, y_k)]$ is a turning step in a minimal distance path P_{ij}. Assume further that $x_k - x_{k-1} > 0$ and $x_{k+1} - x_k < 0$. (A similar argument holds for the other case). Suppose that the path step $[(x_k, y_{k-1}), (x_k, y_k)]$ does not intersect the vertex of a barrier situated to the left of the path step. Then the step could be pushed to the left by some distance $\varepsilon > 0$, reducing the total path length by 2ε. Since this is a contradiction, $[(x_k, y_{k-1}), (x_k, y_k)]$ must intersect a barrier vertex (with the barrier on the left). The identical argument holds for a y turning step. ∎

<u>Theorem 2</u>.   Suppose two points $w(i) \varepsilon W$, $w(j) \varepsilon W$ do not communicate. Then there exists at least one minimal distance feasible path P^*_{ij} between them containing a sequence of simply communicating vertices $\{w(i), w(k_1^*), w(k_2^*),...,w(k_n^*), w(j)\}$.

A y TURNING STEP

DIRECTION OF TURN

AN x TURNING STEP

DIRECTION OF TURN
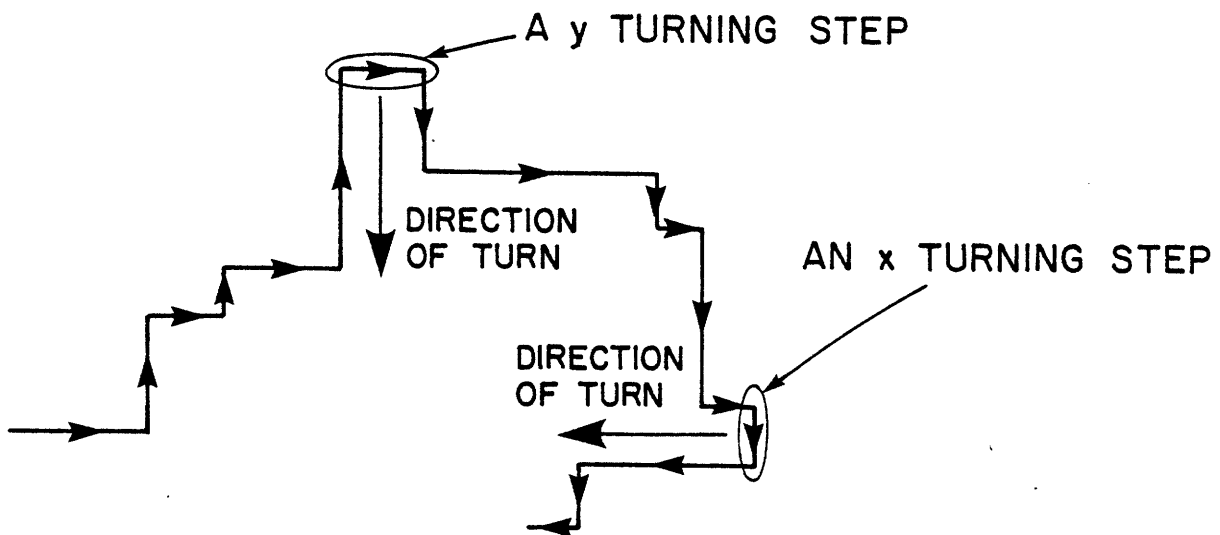
X and Y Turning Steps

Figure 5

Proof: From Lemma 3 all turning steps in a minimal distance path

P_{ij} contain a barrier vertex. But by the definition of

turning steps, the $\{x_k\}$ and $\{y_k\}$ between two successive

turning steps form monotone sequences, i.e. , staircase sub-

paths. By Theorem 1 for each staircase subpath we can con-

struct at least one feasible alternative staircase subpath

containing a sequence of simply communicating vertices,

beginning at one turning step barrier vertex and ending

at the next one. In this manner we can construct the

desired minimal distance path P_{ij}^*. ∎

II. The Minimal Distance Algorithm: POLYPATH

Our results demonstrate that any minimal distance path in R^2 can be replaced by an equi-distance path containing a sequence of simply communicating vertices. Thus the problem of finding a minimal distance rectilinear path between $q(i)$ and $q(j)$, in the presence of polygonal barriers, can be reduced to a finite network problem. The network nodes are the origin-destination points $q(i)$ and the polygon vertices $v(k,m)$. The arcs are bidirectional, having weights corresponding to the distances between simply communicating vertices.

The algorithm, called POLYPATH, contains three major steps:

1. Generation of the network.

2. Identification of all vertices communicating with a root vertex.

3. Finding the minimal distance path.

We consider each step next.

Generating the Network

The network for our problem is $G(W,L)$, where $W = Q \cup V$ is the set of nodes or vertices and L is the set of links. Let S = set of simply communicating vertex pairs. For any $\{w(i),w(j)\} \in S$, the length of the link $\ell(i,j) \in L$ is $d_{ij} = |x_w(i) - x_w(j)| + |y_w(i) - y_w(j)|$. For any $\{w(i),$ $w(j)\} \notin S$, $d_{ij} = +\infty$ [or, equivalently, link $\ell(i,j)$ does not exist]. The matrix of link lengths is $D = (d_{ij})$. As above, we let $t(i,j)$ be the length of a minimal distance path between nodes $w(i)$ and $w(j)$. It is assumed that the network is connected, i.e., that there exists at least one path between each pair of vertices.

Construction of the set S is straightforward. A vertex pair {w(i), w(j)} is included in S if and only if (1) w(i) and w(j) are adjacent vertices of a barrier; or (2) w(i) and w(j) are contained in the same vertex seeking tree, with either w(i) or w(j) being the root vertex of the tree; or (3) w(i) and w(j) have x and y probes that intersect at a point other than a vertex w(k). Generation of the vertex-seeking tree involves only tests for straight line intersection and, for obtuse angle checks, inequality tests. Similarly, checks for intersection of x and y probes involve only intersection tests.

A network minimal distance path between two points is called an optimal nodal path. It should be clear that traversal of any network link $\ell(i,j)$ corresponds to x-y travel along any one of a (usually) noncountably infinite number of staircase paths between the two simply communicating nodes w(i) and w(j). Thus, finding a minimal distance path on the network corresponds to finding a unique sequence of nodes to visit but a nonunique path in R^2.

Vertices Communicating with w(i)

In applications, typically many vertices (perhaps the great majority) communicate, though not simply. Thus it is beneficial to find a procedure for quickly identifying communicating vertices.

Consider w(i) to be the root vertex. We wish to construct a tree $F(w(i)) \subset G(W,L)$ containing all nodes communicating with w(i). F(w(i)) is usually not unique because of the possibility of multiple minimal distance paths; our construction will produce a tree F(w(i)) having a minimal number of nodes in the tree path from the root w(i) to any $w(j) \in F(w(i))$.

To generate F(w(i)), we define eight "orientations" or "directions of travel" from w(i):

1. <u>East:</u> $\{(x,y) \in R^2 : x \geq x_w(i), y = y_w(i)\}$

2. <u>Northeast:</u> $\{(x,y) \in R^2 : x > x_w(i), y > y_w(i)\}$

3. <u>North:</u> $\{(x,y) \in R^2 : x = x_w(i), y \geq y_w(i)\}$

4. <u>Northwest:</u> $\{(x,y) \in R^2 : x < x_w(i), y > y_w(i)\}$

5. <u>West:</u> $\{(x,y) \in R^2 : x \leq x_w(i), y = y_w(i)\}$

6. <u>Southwest:</u> $\{(x,y) \in R^2 : x < x_w(i), y < y_w(i)\}$

7. <u>South:</u> $\{(x,y) \in R^2 : x = x_w(i), y \leq y_w(i)\}$

8. <u>Southeast:</u> $\{(x,y) \in R^2 : x > x_w(i), y < y_w(i)\}$

We define the node orientation matrix $\Phi = (\phi_{ij})$, where

$$\phi_{ij} \equiv \text{orientation of node } w(j) \text{ with respect to node } w(i) \ (\phi_{ij} = 1,2,\ldots,8).$$

For instance, if $\phi_{ij} = 4$, then node $w(j)$ is northwest of node $w(i)$; however, $w(j)$ and $w(i)$ need not communicate. For computer storage purposes, it is useful to note that $\phi_{ji} = (\phi_{ij} + 3)_{\text{mod } 8} + 1$.

For each pair of communicating nodes $w(i)$ and $w(j)$ having orientation ϕ_{ij} we define a <u>permissable direction set</u> $\Gamma(\phi_{ij})$ such that

$$\Gamma(\phi_{ij}) \equiv \text{set of orientations about node } w(j) \text{ that may include staircase paths through } w(j) \text{ to } w(i).$$

For instance, $\Gamma(2) = \{1,2,3\}$, meaning that any vertex $w(j)$ northeast of a root vertex $w(i)$ may have staircase paths from directions 1,2, or 3 passing through it to the root vertex. The sets $\Gamma(\phi_{ij})$ are given in Table 3. These ideas are illustrated in Figure 6.

Permissable Direction Sets

Table III

$$\Gamma(1) = \{1, 2, 3, 7, 8\}$$

$$\Gamma(2) = \{1, 2, 3\}$$

$$\Gamma(3) = \{1, 2, 3, 4, 5,\}$$

$$\Gamma(4) = \{3, 4, 5\}$$

$$\Gamma(5) = \{3, 4, 5, 6, 7\}$$

$$\Gamma(6) = \{5, 6, 7\}$$

$$\Gamma(7) = \{5, 6, 7, 8, 1\}$$

$$\Gamma(8) = \{7, 8, 1\}$$

We say that $w(j)$ n-communicates with $w(i)$ if the <u>minimal number of</u>
nodes in <u>all optimal nodal paths</u> from $w(i)$ to $w(j)$ is $n+1$, $n=0,1,\ldots,n_w-1$.
$w(i)$ 0-communicates with itself. For some node $w(j)$ that n-communicates
with $w(i)$, link $\ell(k,j)$ is a <u>predecessor link</u> for $w(j)$ if $\ell(k,j)$ is on
the algorithm-selected $(n+1)$-node optimal path from $w(i)$ to $w(j)$. For any such
$\ell(k,j)$, $w(k)$ must $(n-1)$ - communicate with $w(i)$ and $\phi_{kj} \ \varepsilon \ \Gamma(\phi_{ik})$.

We construct the tree $F(w(i))$ by iteratively obtaining the sets of
vertices that n-communicate with $w(i)$, together with the corresponding
predecessor node for each vertex, for n first equaling 1, then 2,3, etc.
The necessary node-link sets are

$$\Delta_n(w(i)) \equiv \{(w(j), \ell(k,j)) \ \varepsilon \ G(W,L): \ w(k) \ (n-1)\text{-communicates with}$$
$$w(i), \ d_{kj} < \infty \ , \ \phi_{kj} \ \varepsilon \ \Gamma(\phi_{ik})\} \qquad n=1,2,3,\ldots, \ n_w.$$

EIGHT POSSIBLE
ORIENTATIONS ABOUT
THE ORIGIN, GIVEN IN
COUNTERCLOCKWISE ORDER

(a) EIGHT POSSIBLE ORIENTATIONS

ORIENTATION OF w(j)
WITH RESPECT TO w(i)

$\Gamma(2) = \{1, 2, 3\}$

w(j)

w(i)

ORIENTATIONS
ABOUT w(j)
THAT MAY INCLUDE
STAIRCASE PATHS
THROUGH w(j) TO w(i)
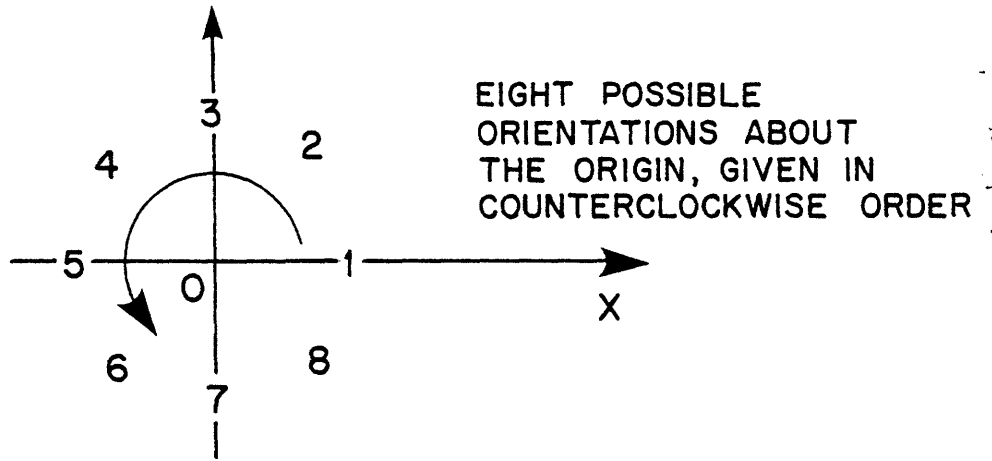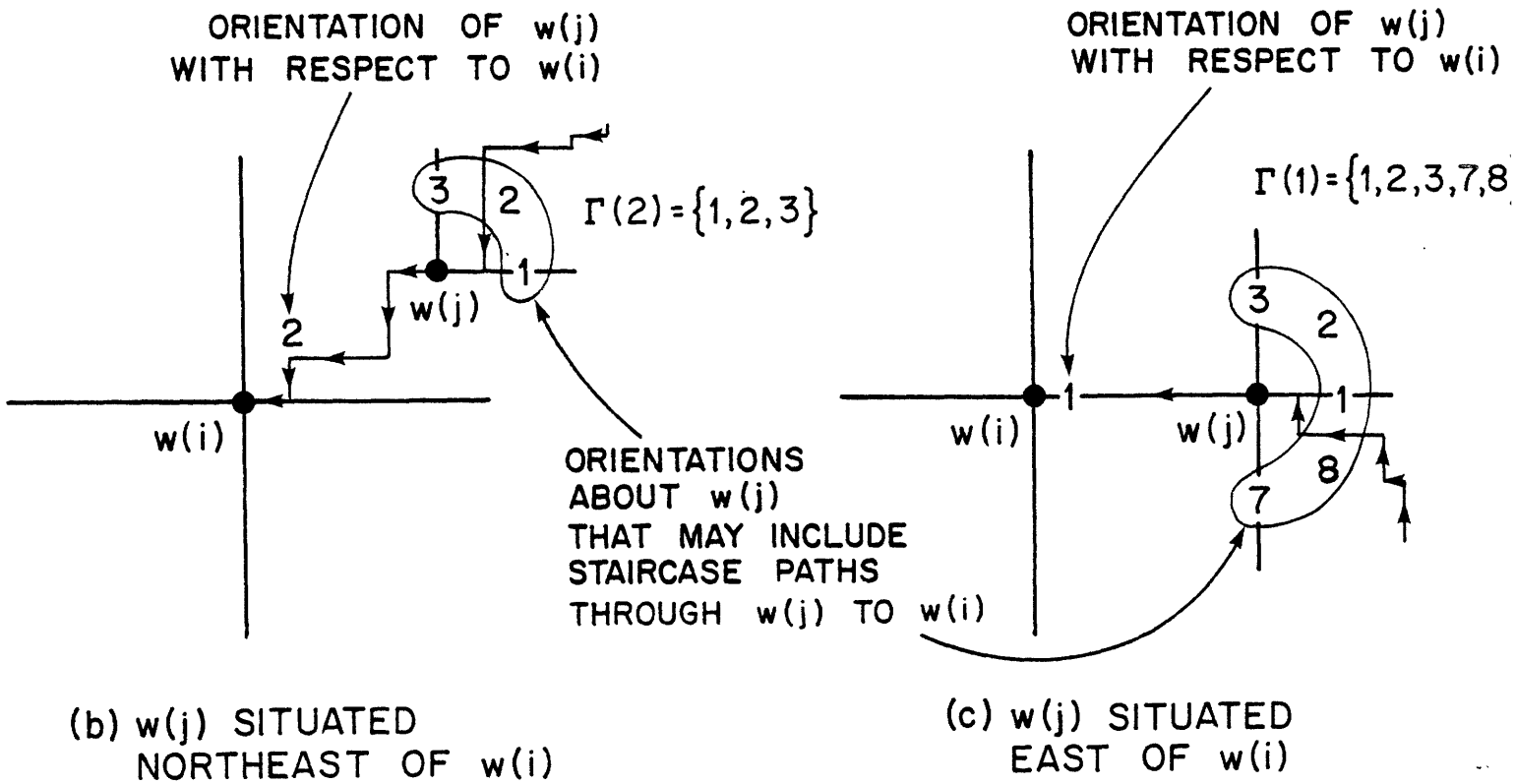
(b) w(j) SITUATED
NORTHEAST OF w(i)

ORIENTATION OF w(j)
WITH RESPECT TO w(i)

$\Gamma(1) = \{1, 2, 3, 7, 8\}$

w(i)

w(j)

(c) w(j) SITUATED
EAST OF w(i)

Orientations and Permissable Direction Sets

Figure 6

As the boundary condition, we say that $\Delta_0(w(i))$ contains only $w(i)$ with

no predecessor link, i.e., $\Delta_0(w(i)) = \{(w(i),\phi)]$. Clearly,

$$F(w(i)) = \bigcup_{n=0}^{n_w} \Delta_n(w(i)).$$

It is noted that some of the $\Delta_n(w(i))$ may be empty, and that $\Delta_n(w(i)) = \phi$

implies that $\Delta_{n+1}(w(i)) = \phi$. The iterative tree generation algorithm

is as follows:

Algorithm A1: Tree Generation

1. $n = 0$, $\Delta_0(w(i)) = \{(w(i)),\phi)\}$; $\Delta_n(w(i)) = \{(\phi,\phi)\}$, $n=1,2,..,n_w$.

2. For each $w(k) \in \Delta_n(w(i))$, scan the kth rows of Φ and D to find

   all links $\ell(k,j)$ such that $d_{kj} < \infty$ and $\phi_{kj} \in \Gamma(\phi_{ik})$.

   For each such link $\ell(k,j)$, if its terminal node $w(j) \notin \bigcup_{m=0}^{n} \Delta_m(w(i))$,

   then

   $$\Delta_{n+1}(w(i)) \leftarrow \Delta_{n+1}(w(i)) \cup \{w(j),\ell(k,j)\}.$$

3. $n \leftarrow n+1$

4. If $n = n_w-1$ or if $\Delta_n(w(i)) = \{(\phi,\phi)\}$, STOP.

   Else go to step 2.

The fact that the minimal distance path from $w(i)$ to any node on the tree

$F(w(i))$ contains a minimal number of links may be useful in applications.

Clearly for any $w(j) \, \varepsilon \, F(w(i))$, $t(i,j) = \left| x_w(i) - x_w(j) \right| + \left| y_w(i) - y_w(j) \right|$. This computation could also be carried out in the above algorithm by setting $t(i,j) = 0$ in step 1 and adding the computation $t(i,j) = t(i,k) + d_{kj}$ at the end of step 2.

Finding All Minimal Distance Paths

We now develop an iterative procedure for finding the minimal distances and corresponding minimal distance paths from a root node $w(i)$ to all other nodes $w(j)$. The procedure is analogous to Dijkstra-type [4] iterations for finding minimal distance paths on a network, except that we begin with the tree $F(w(i))$ and focus on penalty travel distances, not absolute travel distances. Here, the penalty distance associated with an optimal path from $w(j)$ to $w(i)$ is

$$\varepsilon(i,j) = t(i,j) - \left\{ \left| x_w(i) - x_w(j) \right| + \left| y_w(i) - y_w(j) \right| \right\}.$$

The penalty $\varepsilon(i,j)$ represents the net increase in travel distance due to the presence of the barriers. Note that a minimal distance path corresponds to a minimal penalty path. Clearly for two nodes $w(i)$ and $w(j)$ that communicate, $\varepsilon(i,j) = 0$.

Suppose we arrange the nodes in order of their penalty distances $\varepsilon(i,j)$, using the index α_n = identification number of the node that has the nth smallest $\varepsilon(i,j)$. If $F(w(i))$ contains e nodes, then the ordering of the first e nodes is arbitrary since their penalty terms are all 0. For any $m \geq e$, partition the nodes $\{w(i) \varepsilon W\}$ into two sets:

The "closed" nodes: $H_1(i|m) = \{w(j) \in W: j \in [\alpha_1, \alpha_2, \ldots, \alpha_m]\}$

The "open" nodes: $H_2(i|m) = W - H_1(i|m)$

$H_1(i|m)$ is the set of m least penalty nodes for $w(i)$.

Consider all the open nodes that communicate simply with at least one closed node, i.e., $H_2'(i|m) = \{w(j) \in H_2(i|m): d_{\alpha_n j} < +\infty, n=1,2,\ldots,m\}$.

For any open node $w(j) \in H_2'(i|m)$ define its m-stage <u>estimated</u> <u>penalty</u> <u>distance</u> for a trip to $w(i)$ to be

$$\hat{\varepsilon}(i,j|m) \equiv \text{MIN}\ [d_{\alpha_n j} + t(i,\alpha_n)] - (|x_w(i) - x_w(j)| + |y_w(i) - y_w(j)|).$$

$$n \in [1,\ldots,m]:\ d_{\alpha_n j} < +\infty.$$

Here $\hat{\varepsilon}(i,j|m)$ is our current "best guess" of the true penalty $\varepsilon(i,j|m)$, given the identity of the m least penalty nodes. The best guess is generated by attaching each node in $H_2'(i|m)$ directly through one link to a node in $H_1(i|m)$ in a way which minimizes the total penalty.

<u>Theorem 3</u>: For any given $m=1,2,\ldots,n_w$, the $m + 1^{st}$ least penalty node is one with minimal estimated penalty distance, and its penalty distance is this estimate. (i.e., $\varepsilon(i,\alpha_{m+1}) = \underset{j:\ w(j) \in H_2'(i|m)}{\text{MIN}}[\hat{\varepsilon}(i,j|m)] \equiv \varepsilon_0$

and α_{m+1} is a node $w(j) \in H_2'(i|m)$ that achieves the minimum). [Ties can be broken arbitrarily].

<u>Proof</u>: 1. Suppose $\varepsilon(i,\alpha_{m+1}) > \varepsilon_0$. Then closed node (α_{m+1}) could be replaced by an open node $w(j)$ that achieves the

minimum, resulting in a penalty smaller than $\varepsilon(i,\alpha_{m+1})$, a contradiction.

2. Suppose $\varepsilon(i,\alpha_{m+1}) < \varepsilon_0$. Then there exists some open node $w(j'') \varepsilon H_2'(i|m)$ having penalty smaller than the indicated minimum. We need consider only $w(j'') \varepsilon H_2'(i|m)$ since a minimal distance path from any $w(j'') \varepsilon H_2(i|m) - H_2'(i|m)$ must pass through an open node simply communicating with a closed node in $H_1(i|m)$, incurring a penalty not less than that of the simply communicating node. But for any $w(j'') \varepsilon H_2'(i|m)$, a decrease in penalty from that node associated with ε_0 requires a new multi-link path from $w(j'')$ eventually to some node in $H_1(i|m)$. The penalty of any such path cannot be less than the penalty of the penultimate node in the path, which itself is contained in $H_2'(i|m)$. And this penalty cannot be less than ε_0, the indicated minimum. We have reached a contradiction, and the theorem is proved. ▓

We now have all the results necessary to find the minimal distance paths from any root vertex $w(i)$ to all other vertices $w(j)$. Basically, after identifying all nodes that communicate with $w(i)$, we add one node at a time to the tree of our least-penalty nodes. This is a Dijkstra type iteration performed on first differences rather than magnitudes of travel distances. The tree containing the m least penalty nodes for $w(i)$ is fully specified by

$\gamma(i|m) \equiv$ set of pairs of nodes and predecessor links in the tree of m least penalty nodes for $w(i)$

We now detail

Algorithm A2: Finding Minimum Penalty Nodes

1. **Initialization**

   For all e $w(j)$ ε $F(w(i))$, $\varepsilon(i,j) = 0$, $t(i,j) = |x_w(i) - x_w(j)|$ $+ |y_w(i) - y_w(j)|$. Assign each α_i, $i=1,2,\ldots,e$, to a unique vertex having $\varepsilon(i,j) = 0$.

$$m = e$$

$$H_1(i|m) = \{w(i) \varepsilon W: \; i \varepsilon \{\alpha_1, \alpha_2, \ldots, \alpha_m\}\}$$

$$H_2(i|m) = W - H_1(i|m)$$

From A1, $\Upsilon(i|m) = \bigcup_{n=0}^{m} \Delta_n(w(i))$

2. **First Iteration**

   Compute $\hat{\varepsilon}(i,j|m) = MIN [d_{\alpha_n j} + t(i,\alpha_n)] - (|x_w(i) - x_w(j)| + |y_w(i) - y_w(j)|$

   $n \varepsilon [1,\ldots,m]: \; d_{\alpha_n j} < +\infty$

3. **Subsequent Iterations**

   Set $\varepsilon_0 = MIN [\hat{\varepsilon}(i,j|m)]$

   $\quad\quad\quad j: \; w(j) \varepsilon H_2(i|m)$

   α_{m+1} = any vertex j $(w(j) \varepsilon H_2(i|m))$ having $\hat{\varepsilon}(i,j|m) = \varepsilon_0$.

   $t(i,\alpha_{m+1}) = \varepsilon_0 + |x_w(i) - x_w(\alpha_{m+1})| + |y_w(i) - y_w(\alpha_{m+1})|$

   The predecessor link $\ell(\alpha_n, \alpha_{m+1})$ in $\Upsilon(i|m)$ is determined by that α_n, say α_n^o, which yields the minimum ε_0.

$$\gamma(i\,|\,m+1) = \gamma(i\,|\,m) \cup \{w(\alpha_{m+1}), \ \ell(\alpha_n^o, \alpha_{m+1})\}$$

$$H_1(i\,|\,m) = H_1(i\,|\,m) \cup w(\alpha_{m+1})$$

$$H_2(i\,|\,m) = H_2(i\,|\,m) - w(\alpha_{m+1})$$

$$m \leftarrow m+1$$

4. <u>New Estimated Penalties</u>

Compute

$$\hat{\varepsilon}(i,j\,|\,m) = \text{MIN}\,\{\hat{\varepsilon}(i,j\,|\,m-1), \text{MIN}[d_{\alpha_n j} + t(i,\alpha_n) - (|x_w(i) - x_w(j)| + |y_w(i) - y_w(j)|)\}$$
$$m : d_{\alpha_m j} < +\infty$$

If $H_2(i\,|\,m) = \phi$ then exit, else go to Step 3.

Algorithm A2 completes the POLYPATH algorithm, a summary description of which is given in Table IV.

Table IV

POLYPATH: Summary Description

1. Network Formulation

   Network node set $W = Q \cup V$.

   Construct set S of simply communicating vertex pairs:

   $\{w(i), w(j)\}$ are included in S if

   (a) they are adjacent vertices of a barrier

   (b) they are contained in the same vertex seeking tree, with one
        as a root vertex

   (c) they have x and y probes that intersect at a point other
        than another vertex.

   For each $\{w(i), w(j)\} \; \varepsilon \; S$, the length of link $\ell(i,j)$ is

   $$d_{ij} = |x_w(i) - x_w(j)| + |y_w(i) - y_w(j)|;$$

   else $d_{ij} = +\infty$.

   Compute the node orientation matrix $\Phi = (\phi_{ij})$.

2. Find Vertices Communicating with w(i)

   Execute Algorithm A1: Tree Generation

3. Find Minimal Distance Paths to Remaining Vertices

   Execute Algorithm A2: Finding Minimum Penalty Nodes

   Repeat for all i, $i = 1, 2, \ldots, n_Q$, if entire matrix $T = (t(i,j))$

   is desired.

Computational Complexity

The computational complexity of the POLYPATH algorithm, once the network is constructed, is usually less than that of the Dijkstra algorithm. Recall that in our problem there are n_Q origin-destination points, $\sum_{m=1}^{n_B} v_m$ barrier vertices, and $n_w = n_Q + \sum_{m=1}^{n_B} v_m$ nodes in the constructed network.

Suppose our problem is to generate the entire $(n_Q \times n_Q)$ minimal travel distance matrix between all origin-destination pairs $(q(i),q(j))$. We do this by executing POLYPATH n_Q times, once for each possible origin point. [This ignores possible exploitation of the symmetry of the matrix.] Each such execution is done on an n_w - node network, yielding n_w minimal distances and straightforward application of the Dijkstra "node labeling" procedure would require approximately $(5/2) \, n_w^2$ additions and comparisons [5].

But in POLYPATH the tree generation algorithm Al can be executed with no additions or comparisons, just simple storage manipulations (file creation), if the D and Φ matrices are appropriately rearranged to allow easy tracing of trees through permissable directions. This is accomplished by grouping together all nodes that are direction η from $q(i)$, for each $\eta = 1,2,\ldots,8$, and using indirect addressing to retrieve these nodes. Using this technique, the computer can sequentially create the communicating tree of nodes (the zero penalty nodes) with "no computational effort".

Suppose a fraction f of nodes in the complete n_w - node network communicate. Then, Dijkstra - type iterations must be performed only on the $(1-f)n_w$ positive penalty nodes. [These iterations are performed only on the distances, not absolute distances, but the computational work is the same.] The number of additions and comparisons required for each possible origin

point is thus reduced to approximately $(5/2) \; [(1-f)n_w]^2$. The computational complexity associated with generating the entire $(n_Q \times n_Q)$ matrix (ignoring its symmetry] is thus approximately $n_Q \frac{5}{2} \; [(1-f)n_w]^2$. If $f=0.5$, say, POLYPATH reduces the amount of computational work, compared to Dijkstra, by 75 percent.

III. Example

We now illustrate our algorithm by a simple example. To minimize the amount of manual computations, we have chosen a two-barrier, six origin-destination point problem as shown in Figure 7. This example, however, does include all of the different situations that can occur with a vertex-seeking tree; thus we have nodes with 0, 1, 2, 3, or 4 probes and probes terminating on another origin-destination point, or intersecting a barrier side or terminating at a barrier vertex. We shall find the minimum rectilinear distance to all origin-destination points from point 1.

The first step is the generation of the link length (D) and orientation (ϕ) matrixes. (Table V). Each entry (i,j) of the matrix includes the pair d_{ij} and ϕ_{ij} where d_{ij} is the link length from node i to node j and ϕ_{ij} is the orientation of j with respect to i. Any (i,j) for which $d_{ij} = + \infty$ is left blank. Although this matrix generation involves tedious but simple inspections, it can be done efficiently by a computer.

Using the D and ϕ matrixes, we invoke Algorithm A1 to generate the tree of vertices that communicate with node 1, i.e., $F(w(1))$ in our notation. It is noted that nodes 7, 8, 14, 15, 16, 1-communicate with 1, while nodes 3, 5, 9, 11, 2-communicate with 1. There are no n-communicating nodes for n $\geq$ 3. It is noted that 10 of the original 16 nodes are included in the tree. (See Figure 8.)

The final step is to apply Algorithm A2 to find the minimum rectilinear distances to other nodes not on the tree. We use the notation of Algorithm A2:

Figure 7 – Example: Two Barriers, Six Origin-Destination Points

NODE COORDINATES

KEY: (x,y) ——— NODE NUMBER
        7
        ● ——— NODE LOCATION
        NODE LOCATION
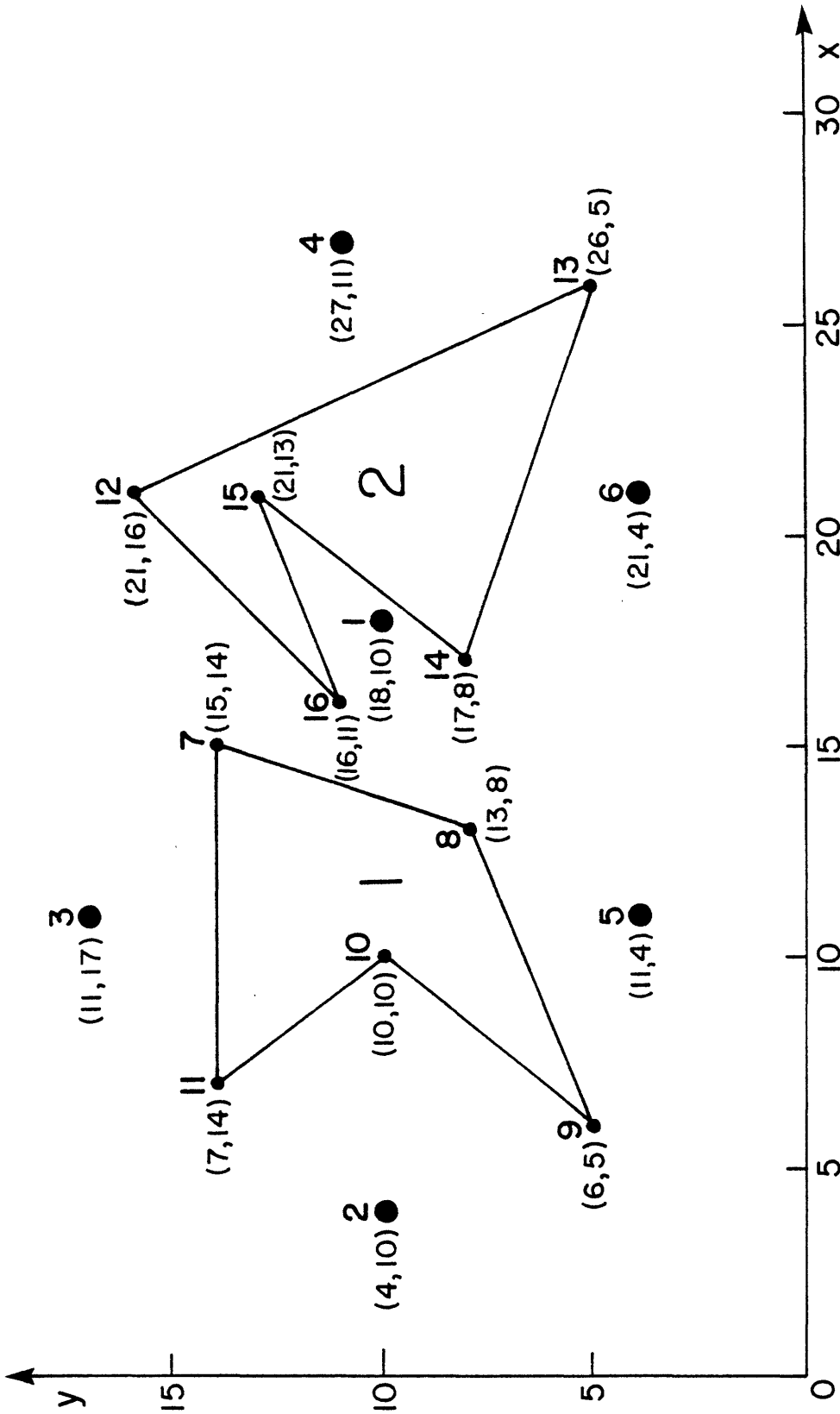
Examples of differing numbers of probes:

node 15 – 0 probe
node 10 – 1 probe
node 8 – 2 probes
node 12 – 3 probes
node 13 – 4 probes

Table V Link Length and Orientation Matrixes (D,Φ)

Nodes	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0 −															
2	14 6	0 −														
3		14 2	0 −	22 4												
4			22 8	0 −												
5	13 4			13 2	0 −	13 6	7 4	7 6								
6					13 8	0 −	7 8									
7					7 4	18 2	0 −	7 6	6 1							
8	7 2					16 8	8 6	0 −	7 8							
9	7 4		7 4		6 8	18 6	14 2	6 2	0 −	9 2	10 2	26 2	20 1	14 2		16 2
10		6 5			14 6	16 8	14 2	8 6	0 −	0 −	7 4			9 6	0 −	7 6
11		7 6	7 2		10 5	12 8	8 2	8 6	16 6	7 8	0 −	16 2	16 8	0 −	9 2	4 4
12	23 6	11 4	11 4	11 8	16 6	6 6	20 4	16 8	26 6		16 6	0 −	0 −	12 4	0 −	16 4
13		27 4	27 4	7 2	16 6	6 6	20 4	16 8	20 5		16 4	16 4	0 −	0 −		16 2
14	3 2				10 6	8 8	8 4	4 5	14 6		0 −	12 8	0 −		9 2	4 4
15	6 6										9 2		9 6	0 −	0 −	7 6
16	3 0	11 4	11 4	12 6	12 8	4 4	6 6	16 6	10 2		10 2	16 8	4 8	7 2	0 −	0 −

Key: The matrix pair (a(i,j), b(i,j)) in row i, column j
      is (d_ij, φ_ij). [Any (i,j) for which d_ij = +∞ is left blank.]

Note: d_ji = d_ij, φ_ji = (φ_ij + 3) mod 8 + 1

Figure 8 - Tree of Vertices Communicating with Node 1 [F(w(1))]
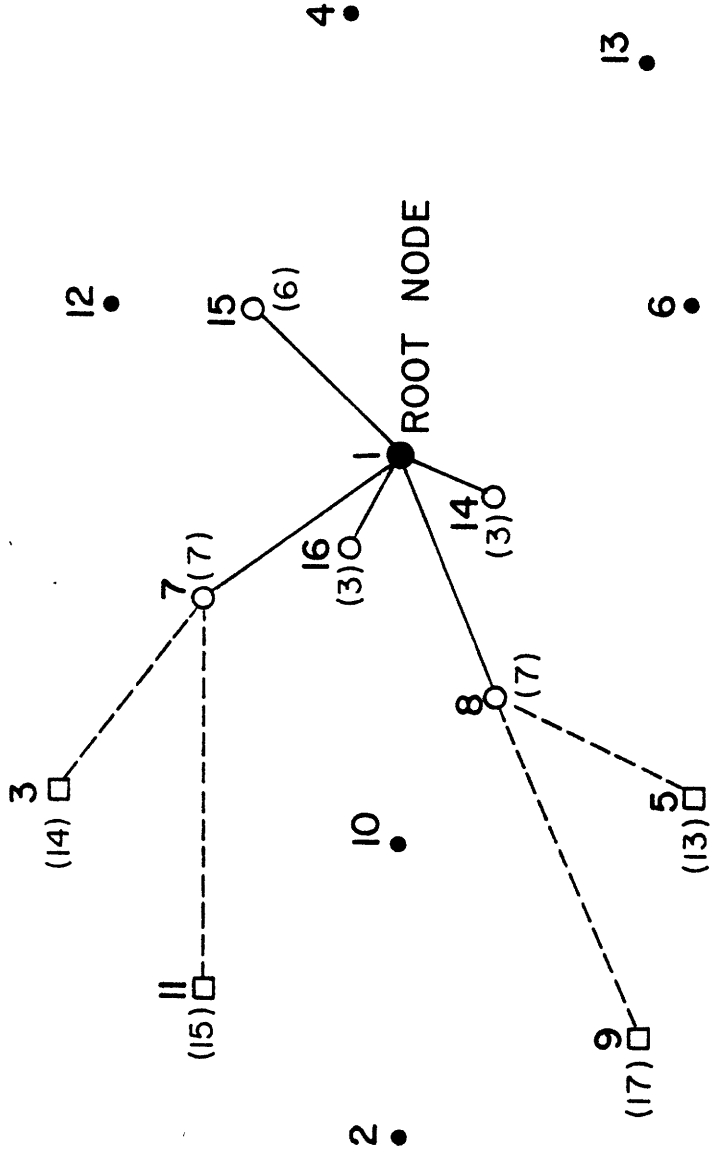
1. Initialization

10 closed nodes: $\{1, 3, 5, 7, 8, 9, 11, 14, 15, 16\}$

$t(1,1) = 0$, $t(1,3) = 14$, $t(1,5) = 13$, $t(1,7) = 7$, $t(1,8) = 7$,
$t(1,9) = 17$, $t(1,11) = 15$, $t(1,14) = 3$, $t(1,15) = 6$, $t(1, 16) = 3$

$\alpha_1 = 1$, $\alpha_2 = 3$, $\alpha_3 = 5$, $\alpha_4 = 7$, $\alpha_5 = 8$, $\alpha_6 = 9$, $\alpha_7 = 11$, $\alpha_8 = 14$,

$\alpha_9 = 15$, $\alpha_{10} = 16$

$m = 10$

$H_1(1|10) = \{w(i)\varepsilon W: i\varepsilon \{\alpha_1,\ldots,\alpha_{10}\}\}$

$H_2(1|10) = \{2, 4, 6, 10, 12, 13\}$

$\gamma(1|10) = \{\{1,\phi\}, \{7,[7,1]\}, \{8,[8,1]\}, \{14,[14,1]\},$
$\{15,[15,1]\}, \{16,[16,1]\}, \{3,[3,7]\}, \{11,[11,7]\},$
$\{5,[5,8]\}, \{9,[9,8]\}\}$

2. First Iteration

Using Closed Node \ Open Node	2 $\hat{\varepsilon}(1,2|10)$	4 $\hat{\varepsilon}(1,4|10)$	6 $\hat{\varepsilon}(1,6|10)$	10 $\hat{\varepsilon}(1,10|10)$	12 $\hat{\varepsilon}(1,12|10)$	13 $\hat{\varepsilon}(1,13|10)$
3	14	(16)			16	28
5	12		15			
7			14		6	14
8			10			10
9	9			16		
11	(8)			(14)		
14			(2)			(2)
15						
16					(4)	

$\hat{\varepsilon}(1,2|10) = \text{MIN} \{14, 12, 9, 8\} = 8$ (hence circled above)

Similarly, $\hat{\varepsilon}(1,4|10) = 16$. $\hat{\varepsilon}(1,6|10) = 2$, $\hat{\varepsilon}(1,10|10) = 14$,

$\hat{\varepsilon}(1,12|10) = 4$, $\hat{\varepsilon}(1,13|10) = 2$.

$$\varepsilon_0 = \underset{j:\ w(j)\varepsilon H_2(1|10)}{\text{MIN}} \{\hat{\varepsilon}(1,j|10)\} = \text{MIN}\ \{8,\ 16,\ 2,\ 14,\ 4,\ 2\} = 2$$

$\alpha_{10+1} = \alpha_{11} = 6$ or 13 (Ties can be broken arbitrarily;

choose $\alpha_{11} = 6$)

$t(1,6) = 2 + c_{16} = 2 + 9 = 11$

3. Using closed node 6, $\hat{\varepsilon}(1,j|11) = \hat{\varepsilon}(1,j|10)$ for $j = 2, 10, 12, 13$

$$\hat{\varepsilon}(1,4|11) = 14$$

$$\varepsilon_0 = \underset{j:\ w(j)\varepsilon H_2(1|11)}{\text{MIN}} \{\hat{\varepsilon}(1,j|11) = \text{MIN}\ \{8,\ 14,\ 14,\ 4,\ 2\} = 2$$

$\alpha_{11+1} = \alpha_{12} = 13$

$t(1,13) = 2 + c_{1(13)} = 2 + 13 = 15$

4. Using closed node 13, $\hat{\varepsilon}(1,j|12) = \hat{\varepsilon}(1,j|11)$ for $j = 2, 10, 12$

$$\hat{\varepsilon}(1,4|12) = 12$$

$$\varepsilon_0 = \underset{j:\ w(j)\varepsilon H_2(1|12)}{\text{MIN}} \{\hat{\varepsilon}(1,j|12)\} = \text{MIN}\ \{8,\ 12,\ 14,\ 4\} = 4$$

$\alpha_{12+1} = \alpha_{13} = 12$

$t(1,12) = 9 + 4 = 13$

5. Using closed node 12, $\hat{\varepsilon}(1,j|13) = \hat{\varepsilon}(1,j|12)$ for $j = 2, 4, 10$

$$\varepsilon_0 = \underset{j:\ w(j)\varepsilon H_2(1|13)}{\text{MIN}} \{\hat{\varepsilon}(1,j|13)\} = \text{MIN}\ \{8,\ 12,\ 14\} = 8$$

$\alpha_{13+1} = \alpha_{14} = 2$

$t(1,2) = 14 + 8 = 22$

6. Using closed node 2, $\hat{\varepsilon}(1,j|14) = \hat{\varepsilon}(1,j|13)$ for $j = 4, 10$

$$\varepsilon_0 = \underset{j : w(j) \varepsilon H_2(1|14)}{\text{MIN}} \hat{\varepsilon}(1,j|14) = \text{MIN} \{12, 14\} = 12$$

$$\alpha_{14+1} = \alpha_{15+1} = 4$$

$$t(1,4) = 10 + 12 = 22$$

7. Using closed node 4, $\hat{\varepsilon}(1,10|15) = \hat{\varepsilon}(1,10|14) = 14$

$$\varepsilon_0 = 14, \quad \alpha_{15+1} = \alpha_{16} = 10$$

$$t(1,10) = 8 + 14 = 22$$

$$H_2(1|16) = \phi$$

At this point, all nodes are closed. We have found the minimum rectilinear distance to all origin-destination points from point 1 and the example is concluded.

References

1. Vaccaro, Henry, "Alternative Techniques for Modeling Travel Distance," Masters Thesis in Civil Engineering, (unpublished), Massachusetts Institute of Technology, June 1974.

2. Lozano-Perez, T., and M. Wesley, "An Algorithm for Planning Collision-Free Paths Amongst Polyhedral Obstacles," IBM Thomas J. Watson Research Center, RC 7171, June 1978.

3. Nilsson, N.J., "A Mobile Automation: An Application of Artificial Intelligence Techniques," Proc. IJCAI, 1969, pp. 509-520.

4. Dijkstra, E.W., "A Note on Two Problems in Connexion with Graphs," Numerische Mathematik 1, pp. 269-271 (1959).

5. Dreyfus, Stuart E., "An Appraisal of Some Shortest-Path Algorithms, Operations Research, May-June 1969, pp. 395-412.