NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Report 32-1369

A Path-Finding Algorithm for an Unmanned Roving Vehicle

Donald E. Kirk

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

May 15, 1969

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Report 32-1369

A Path-Finding Algorithm for an Unmanned Roving Vehicle

Donald E. Kirk

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

May 15, 1969

TECHNICAL REPORT 32-1369

Prepared Under Contract No. NAS 7-100
National Aeronautics and Space Administration

Preface

The work described in this report was performed by the Guidance and Control Division of the Jet Propulsion Laboratory.

The author is currently an Assistant Professor of Electrical Engineering at the Naval Postgraduate School in Monterey, California.

Contents

Table

Figures

Abstract

Plans for future space missions include an unmanned roving vehicle designed to explore the Martian surface. Because of the long transit times required for the transmission of TV information from Mars to earth, control of the vehicle by an earth-based operator viewing a TV picture of the terrain is not feasible. As a result, the vehicle is to be controlled by a computer which is on board. This report discusses a routing algorithm which has been proposed for controlling the roving vehicle. This algorithm utilizes gross terrain information to determine a nominal optimal path from some initial point to a specified destination.

A Path-Finding Algorithm for an Unmanned Roving Vehicle

I. Introduction

Plans for future space missions include an unmanned roving vehicle designed to explore the Martian Surface. Because of the long transit times required for the transmission of TV information from Mars to earth, control of the vehicle by an earth-based operator viewing a TV picture of the terrain is not feasible. As a result, the vehicle is to be controlled by a computer either in an orbiting satellite or in the vehicle itself.

A path-finding algorithm which uses only local sensor information has been developed by JPL (Ref. 1). Although this algorithm will find a path (if one exists) from a specified initial point to a designated terminal point, the vehicle may take a devious route with excursions which could be avoided by using more than on-board sensor information. To alleviate this difficulty, gross terrain information available from a previous Mars orbiter mission can be utilized to determine a nominal optimal path from some initial point to a specified destination. This can be accomplished by a computer program which uses Picard's method of successive approximations to solve a functional recurrence equation obtained by applying the principle of optimality to the routing problem (Ref. 2). The algorithm can be applied to a variety of performance criteria, such as elapsed time, distance travelled, or energy expended.

The two algorithms described above have been combined and are capable of determining a quasi-optimal path between any two points on a surface. The nominal optimal path is followed as long as no local obstacles are encountered. If obstacles are detected, control is switched to the algorithm which uses only local sensor information. After circumnavigating the detected obstacle, the vehicle continues along the nominal optimal path.

Further investigation is required to ascertain the feasibility of using a statistical cost measure and making the algorithm adapt to the observed features of the surface being explored.

This report describes the global section of the dual path-finding algorithm.

II. Problem Formulation and Solution

A. The Functional Recurrence Equation

Let (x_i, y_i), $i = 1, 2, \cdots, N$, be the coordinates of a set of N points in two-dimensional Euclidean space (E^2).

Let the cost of moving from the point k with coordinates (x_k, y_k) to the point j with coordinates (x_j, y_j) along the straight line joining these two points be denoted by t_{kj}. Assume that t_{kj} is known for all $j, k = 1, \cdots, N$, and that

$$t_{kj} \begin{cases} > 0 & k \neq j \\ = 0 & k = j \end{cases} \qquad k, j = 1, \cdots, N \tag{1}$$

The problem is to find the sequence of modes to be traversed in moving from some initial point k to any other specified grid point j so that the minimum cost is incurred. Such a path will be called the *optimal path* from k to j.

From the principle of optimality (Ref. 2), the functional recurrence equation

$$c_{kj} = \min_{i \neq k} \{t_{ki} + c_{ij}\} \; k, j = 1, 2, \cdots, N \tag{2}$$

is obtained. The term c_{kj} is, by definition, the *minimum* cost of going from point k to point j via any number of intermediate points. Clearly, the number of intermediate points cannot exceed $N - 2$ since this implies that at least one loop exists, and eliminating such a loop reduces the cost. The initial condition for Eq. (2) is

$$c_{kk} = 0 \qquad k = 1, 2, \cdots, N$$

To solve Eq. (2), we use Picard's method of successive approximations, that is,

$$c_{kj}^{(\ell+1)} = \min_{\substack{i \\ i \neq k}} \{t_{ki} + c_{ij}^{(\ell)}\} \qquad k, j = 1, 2, \cdots, N \tag{3}$$

with the initial value

$$c_{kj}^{(0)} = t_{kj} \qquad k, j = 1, 2, \cdots, N \tag{4}$$

The successive approximations have the following physical interpretation:

(1) The term $c_{kj}^{(0)}$ is the minimum cost to go from k to j directly, i.e., via no intermediate modes.

(2) The term $c_{kj}^{(1)} = \min_{i \neq k} \{t_{ki} + c_{ij}^{(0)}\} = \min_{i \neq k} \{t_{ki} + t_{ij}\}$ is the minimum cost to go from k to j via *at most* one intermediate mode.

(3) The term $c_{kj}^{(\ell+1)} = \min_{i \neq k} \{t_{ki} + c_{ij}^{(\ell)}\}$ is the minimum cost to go from k to j via at most $\ell + 1$ intermediate modes.

To solve for $c_{kj}^{(\ell+1)}$, t_{ki}, $i \neq k$, $i = 1, 2, \cdots, N$, (the k^{th} row of the T matrix), and $c_{ij}^{(\ell)}$, $i \neq j$, $i = 1, 2, \cdots, N$, (the j^{th} column of the $C^{(\ell)}$ matrix), are required. The solution is simply a matter of comparing the values of $t_{ki} + c_{ij}^{(\ell)}$ for $i = 1, 2, \cdots, N$, $i \neq k$, to find the minimizing value of i. A simple example is given in the Appendix.

The solution is carried out by first setting

$$c_{kj}^{(0)} = t_{kj} \qquad k, j = 1, 2, \cdots, N$$

Using $C^{(0)}$, $C^{(1)}$ can be generated from Eq. (2). The iterative process continues until $C^{(\ell+1)} = C^{(\ell)}$. The process must converge in at most $(N - 2)$ iterations because the optimal path can contain at most $(N - 2)$ intermediate modes. In addition to determining the minimum costs to move from point to point, the algorithm also generates information which is sufficient to determine the sequence of nodes on an optimal path.

B. Computational Procedure

The algorithm is divided into two subsections. In the first, the minimum cost matrix, C, and a matrix named NODE are generated. The determination of C is obtained by solving Eq. (3) as described previously. The kj^{th} element of the matrix NODE contains the number of the node following k on the optimal path from k to j. This matrix is generated along with the matrix C. The kj^{th} element of the initial NODE matrix, n_{kj} is simply j. During the first iteration (the generation of $C^{(1)}$) in calculating $C_{kj}^{(1)}$ from Eq. (3), the value of i for which the minimum cost occurs is stored. Then a test is made to see if $C_{kj}^{(1)} < C_{kj}^{(0)}$; if so, n_{kj} is set equal to i; otherwise $(C_{kj}^{(1)} = C_{kj}^{(0)})$ n_{kj} is left unchanged. In this way the algorithm not only generates a minimum cost path, but selects the path having a minimum number of intermediate nodes.

The second subsection of the computational procedure calculates the optimal sequence of nodes to be traversed in moving from any point k to any other point j. This is done by the program in the following manner:

(1) q is set equal to k, $NSEQ(1) = k$, and $i = 2$. $NSEQ(i)$ is an array for storing the sequence of nodes on the optimal path from k to j.

(2) $NSEQ(i)$ is set equal to N_{qj}, the qj^{th} element of the matrix NODE.

(3) If $NSEQ(i) = j$, the procedure is completed and the array $NSEQ$ contains the sequence of nodes on the optimal path from k to j.

(4) If $NSEQ(i) = \ell \neq j$, $NSEQ(i)$ is set equal to ℓ, i is increased by one, q is set equal to ℓ, and the program returns to step 2. This procedure continues until the test described in step 2 is satisfied.

This algorithm was programmed in FORTRAN IV for operation on the IBM 7094 digital computer. A flow chart of the program is given in the Appendix.

A second program was written for the situation where the matrix T is symmetric (see the Appendix). In this case, it can be shown that the optimal path from j to k has the same nodes, traversed in reverse order, as the optimal path from k to j, and that every $C^{(i)}$ matrix is symmetric. As a result, only the elements above the diagonal of C need to be computed and sorted. The NODE matrix is not symmetric; however, any optimal path can be generated using only the above diagonal elements. Physically, a symmetric cost matrix occurs if the measure is the distance travelled.

C. Problem Formulation

To use the algorithm, a grid of points, or nodes, must be selected, and the costs of travelling between any two points directly must be determined. Factors which may influence the number and placement of grid points are:

(1) The resolution of TV reconnaissance data which is available.

(2) The observed topographical features of the terrain.

(3) The range of vehicle sensors.

(4) The amount of computer storage available.

Computer storage availability should not be critical since it is envisioned that the computations will be performed on earth, and relevant information concerning optimal routes will be relayed to the vehicle through a command link.

In the examples considered, uniform grids of points were used. Various grid sizes were employed to determine the sensitivity of path cost to the fineness of grid structure.

The cost matrix must also be selected. Several possible cost functions were considered:

(1) Two-dimensional Euclidean distance.

(2) Three-dimensional Euclidean distance.

(3) Elapsed time.

(4) Energy expended.

(5) Various statistical estimates of 1–4.

Eventually, it seems likely that a statistical measure of cost would be most useful, because of the uncertain nature of the terrain. A possible procedure for using a statistical terrain model is:

(1) From available reconnaissance data, divide the observed region into several terrain categories, e.g., very rough, rough, smooth, very smooth.

(2) Associate with each terrain category an estimate of the cost of a path and determine the optimal cost and node matrices using these estimates in the program described in Section II-B.

(3) When the vehicle performs missions on the Martian surface, measure the actual cost incurred for each path segment, and use this information to revise the original cost matrix, T. It is anticipated that the information obtained about a path in a certain category would be used to revise the estimated costs for *all* paths in that terrain category. In fact, it may even be advantageous to use observed data from one category to revise the estimated costs for paths in other categories.

(4) When sufficient information is available, the revised initial cost estimates would be used to re-compute the optimal cost and node matrices.

This procedure is suggested as an avenue for further investigation, because it makes the path-finding algorithm adaptive, and simultaneously incorporates terrain uncertainty into the algorithm.

III. Routing Examples

A. Two-Dimensional Euclidean Cost Measure

Figure 1 shows a terrain map synthesized using Gaussian hills (Ref. 1). In this example, the shaded area was defined to be *inadmissible*—no path is allowed to penetrate the shaded area. Straight-line paths between grid points which intersect the inadmissible regions are assigned very large costs in the T matrix so that such paths

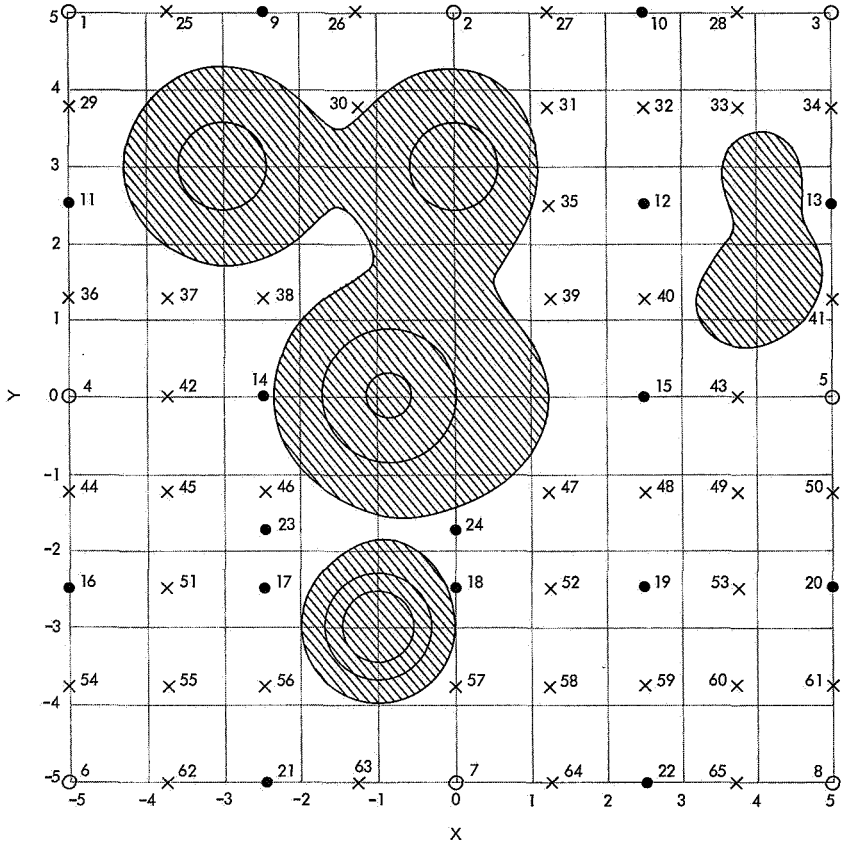

Fig. 1. A terrain map and its grid

will not be selected by the algorithm. Assigning large costs to inadmissible paths is simply a computational convenience.

The cost measure used in this example was the planar distance between the points; thus, if no inadmissible regions were present, the optimal paths would all be simply the straight lines joining the initial and terminal points.

This problem was solved with 8, 24, and 65 grid points to illustrate the effect of grid size on the optimal paths and costs. In Run 1, only the points 1–8 were used (denoted by 0's on Fig. 1). In Run 2, points 1–24 were used, and Run 3 utilized all 65 grid points. Since only points 1–8 are common to the three runs, only the minimum costs associated with these points are shown below for Runs 1, 2, and 3.

Run 1

	1	2	3	4	5	6	7	8
1	0.	5.	10.	5.	15.	10.	15.	20.
2		0.	5.	10.	10.	15.	17.1	15.
3			0.	15.	5.	17.1	12.1	10.
4				0.	17.1	5.	10.	15.
5					0.	12.1	7.1	5.
6						0.	5.	10.
7							0.	5.
8								0.

Run 2

	1	2	3	4	5	6	7	8
1	0.	5.	10.	5.	13.5	10.	12.8	15.9
2		0.	5.	10.	8.5	15.	11.4	11.4
3			0.	14.6	5.	15.7	11.4	10.
4				0.	10.8	5.	8.1	11.5
5					0.	11.6	7.1	5.
6						0.	5.	10.
7							0.	5.
8								0.

Run 3

	1	2	3	4	5	6	7	8
1	0.	5.	10.	5.	12.3	10.	11.9	15.2
2		0.	5.	8.9	7.5	13.9	11.2	11.2
3			0.	13.9	5.	15.3	11.3	10.
4				0.	10.6	5.	7.3	11.2
5					0.	11.4	7.1	5.
6						0.	5.	10.
7							0.	5.
8								0.

To determine the sensitivity of path length to grid size, let us compare the mean path length for all of the points which are not "mutually visible" (the straight line joining these points passes through the inadmissible region). The points obstructed from each other are:

1–5, 1–7, 1–8, 2–4, 2–5, 2–6, 2–7, 2–8

3–4, 3–6, 3–7, 4–5, 4–7, 4–8, 5–6.

The mean path lengths between these points are:

Run 1	*Run 2*	*Run 3*
14.367	12.417	11.54

If the grid structure were infinitely fine, the optimal paths would be smooth curves. By using a finite grid as in this

example, the path determined is a constrained optimal path—constrained because all paths are compound of straightline segments which connect a finite number of nodes. As the grid structure becomes finer, the constrained optimal path approaches the true optimal path.

Some typical optimal paths are shown in Figs. 2, 3, and 4.

B. Three-Dimensional Euclidean Distance Cost Measure

The same terrain map as shown in Fig. 1 was used except that the shaded region was not declared to be inadmissible; rather, the cost measure used was the actual distance travelled by the vehicle. The initial T matrix was generated by the digital computer program. The example was solved using 8 and 24 points. The resulting minimum cost matrices are shown below.

Run 1 (used only 8 points)

	1	2	3	4	5	6	7	8
1	0.	5.05	10.01	5.01	11.20	10.01	12.	14.14
2		0.	5.12	9.43	7.16	13.02	14.25	11.24
3			0.	14.55	5.28	17.37	12.37	10.32
4				0.	10.03	5.00	7.37	11.18
5					0.	12.09	7.09	5.04
6						0.	5.00	10.00
7							0.	5.00
8								0.

Run 2 (used 24 points; points 1–8 shown below)

	1	2	3	4	5	6	7	8
1	0.	5.05	10.01	5.01	11.20	10.01	11.85	14.14
2		0.	5.12	8.11	7.16	12.30	11.73	11.24
3			0.	11.53	5.28	14.32	11.63	10.28
4				0.	10.03	5.00	7.37	11.18
5					0.	11.46	7.09	5.04
6						0.	5.00	10.00
7							0.	5.00
8								0.

The optimal routes for the two grid sizes are:

Initial	*Final*	*Run 1*	*Run 2*
1	5	1–5	1–5
3	6	3–5–7–6	3–12–6
4	5	4–5	4–5

The mean path costs for these grids are:

Run 1	*Run 2*
9.48	9.04

As one would anticipate, the cost matrices and paths are less sensitive to grid size when the vehicle is allowed to climb hills.

Some optimal paths are shown in Fig. 5. Notice that, compared with the preceding example, the paths tend to be simply the direct path between the points. Of course, this qualitative statement will not be valid for mountain ranges of arbitrary altitude.

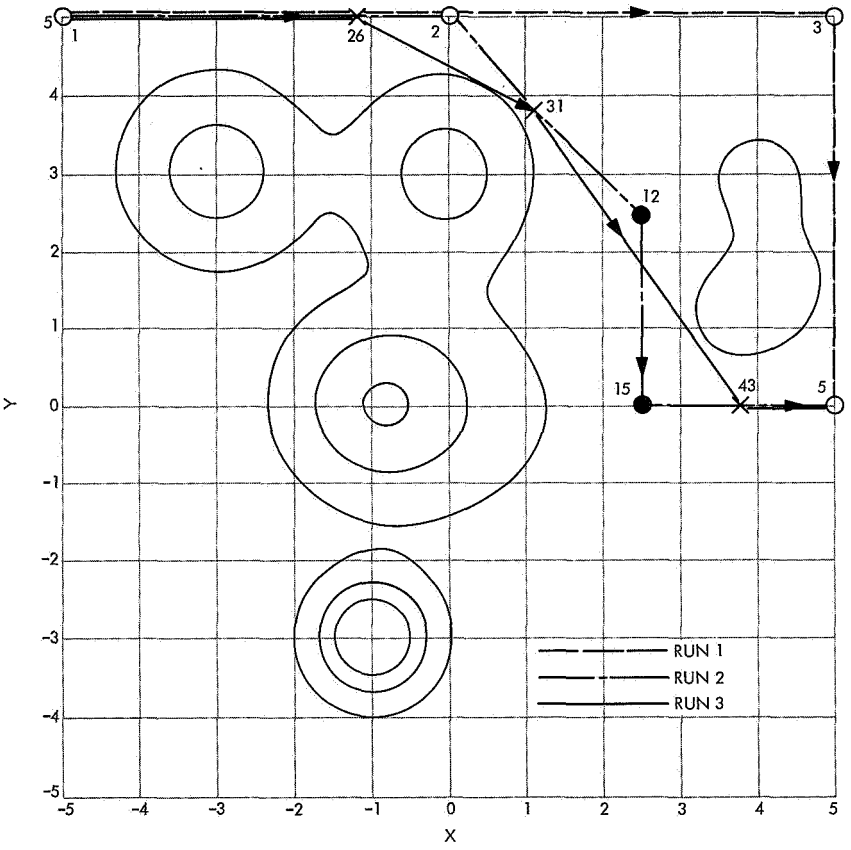

Fig. 2. Optimal paths from points 1–5; two-dimensional Euclidean distance measure

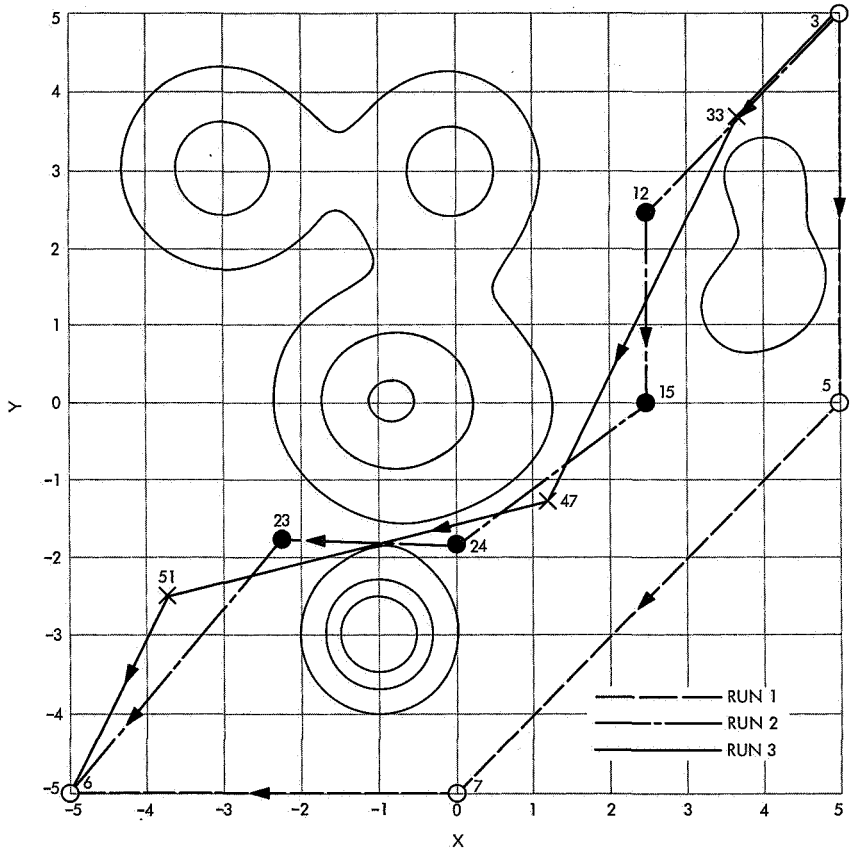

Fig. 3. Optimal paths from points 3–6

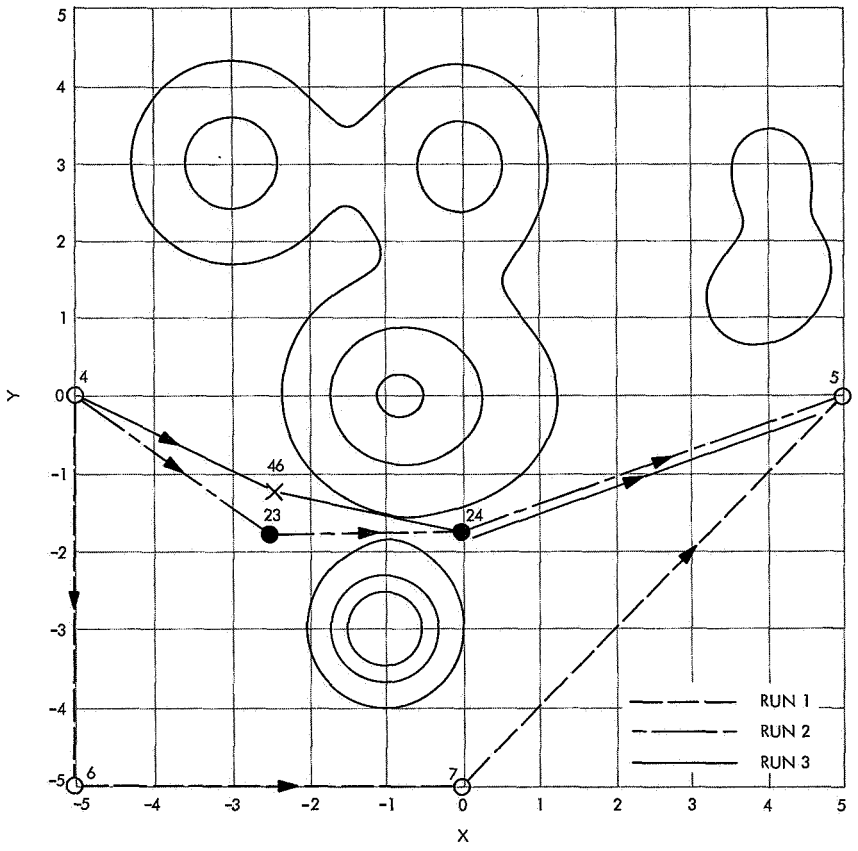

Fig. 4. Optimal paths from points 4–5

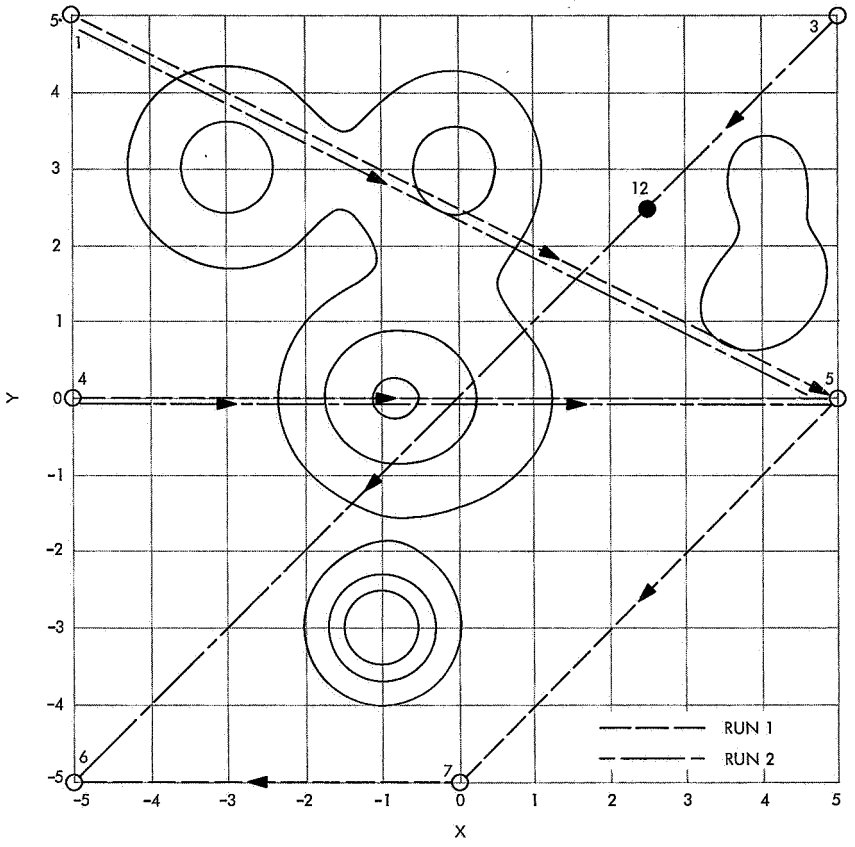

Fig. 5. Optimal routes for two grid sizes; three-dimensional Euclidean distance measure

Appendix
An Example

The matrix of costs of moving from any point on a five-mode grid to any other point directly is the following:

$$T = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0. & 1. & 5. & 10. & 2. \\ 1. & 0. & 6. & 3. & 9. \\ 5. & 6. & 0. & 2. & 15. \\ 10. & 3. & 2. & 0. & 4. \\ 2. & 9. & 15. & 4. & 0. \end{bmatrix} \end{array} \tag{A-1}$$

Setting $C^{(0)} = T$, from the successive approximation Eq. (3), we obtain values for the first row of the $C^{(1)}$ matrix as:

$$C_{11}^{(1)} = 0. \tag{A-2}$$

$$\begin{aligned} C_{12}^{(1)} &= \min\{t_{12} + C_{22}^{(0)}, t_{13} + C_{32}^{(0)}, t_{14} + C_{42}^{(0)}, t_{15} + C_{52}^{(0)}\} \\ &= \min\{1 + 0, 5 + 6, 10 + 3, 2 + 9\} = 1 \end{aligned} \tag{A-3}$$

and the 1–2 element of the node matrix is $n_{12} = 2$;

$$\begin{aligned} C_{13}^{(1)} &= \min\{t_{12} + C_{23}^{(0)}, t_{13} + C_{33}^{(0)}, t_{14} + C_{43}^{(0)}, t_{15} + C_{53}^{(0)}\} \\ &= \min\{1 + 6, 5 + 0, 10 + 2, 2 + 15\} = 5 \end{aligned} \tag{A-4}$$

$n_{13} = 3$;

$$\begin{aligned} C_{14}^{(1)} &= \min\{t_{12} + C_{24}^{(0)}, t_{13} + C_{34}^{(0)}, t_{14} + C_{44}^{(0)}, t_{15} + C_{54}^{(0)}\} \\ &= \min\{1 + 3, 5 + 2, 10 + 0, 2 + 4\} = 4 \end{aligned} \tag{A-5}$$

$n_{14} = 2$;

$$\begin{aligned} C_{15}^{(1)} &= \min\{t_{12} + C_{25}^{(0)}, t_{13} + C_{35}^{(0)}, t_{14} + C_{45}^{(0)}, t_{15} + C_{55}^{(0)}\} \\ &= \min\{1 + 9, 5 + 15, 10 + 4, 2 + 0\} = 2 \end{aligned} \tag{A-6}$$

$n_{15} = 5$.

Table A-1 shows the cost and node matrices after one iteration. In this example, the procedure converged after one iteration, so $C^{(1)}$ and NODE$^{(1)}$ are the optimal cost and NODE Matrices.

Table A-1. The cost and node matrices after the first iteration

A. Cost matrix

	1	2	3	4	5
1	0.0000	1.0000	5.0000	4.0000	2.0000
2	1.0000	0.0000	5.0000	3.0000	3.0000
3	5.0000	5.0000	0.0000	2.0000	6.0000
4	4.0000	3.0000	2.0000	0.0000	4.0000
5	2.0000	3.0000	6.0000	4.0000	0.0000

$= C^{(1)}$

B. Node matrix

	1	2	3	4	5
1	1	2	3	2	5
2	1	2	4	4	1
3	1	4	3	4	4
4	2	2	3	4	5
5	1	1	4	4	5

$= \text{NODE}^{(1)}$

To see how an optimal path is obtained from NODE, suppose that it is desired to move from point 1 to point 4. Then, n_{14} is 2; so the first segment of the path is from 1 to 2. Thus, point 2 is on the optimal path from 1 to 4. By the principle of optimality then, the optimal path from 1 to 4 must include the optimal path from 2 to 4. From the NODE matrix, we see that $n_{24} = 4$; therefore, the optimal path from 1 to 4 is 1–2–4.

In a similar manner, the optimal path from 3 to 5 is found by looking up $n_{35} = 4$, and then $n_{45} = 5$; thus, the optimal path from 3 to 5 is 3–4–5.

For an optimal route (non-symmetric) and an optimal route (symmetric) of the programmed algorithm, refer to flow-chart Figs. A-1 and A-2, respectively, which follow.

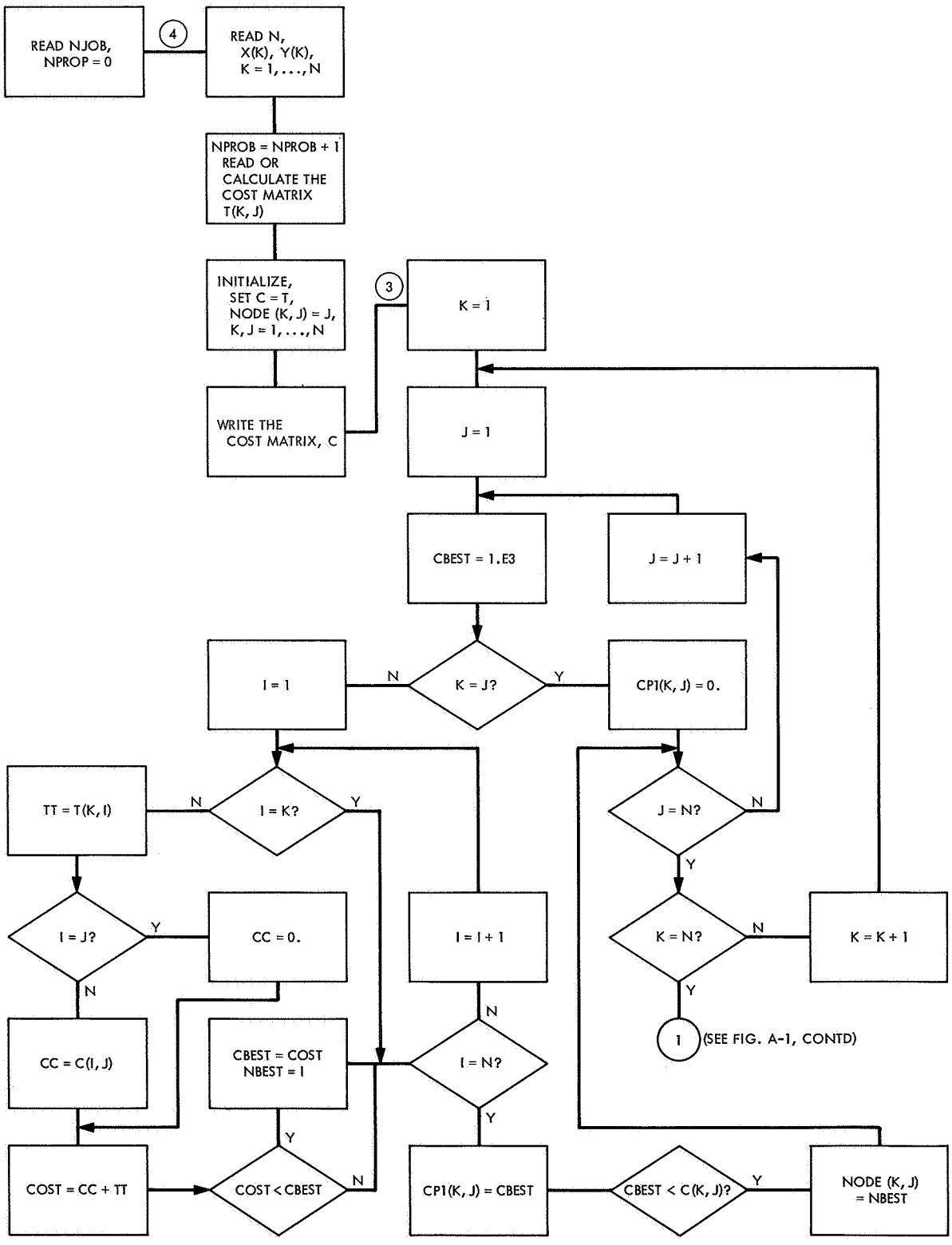

Fig. A-1. Optimal route—nonsymmetric

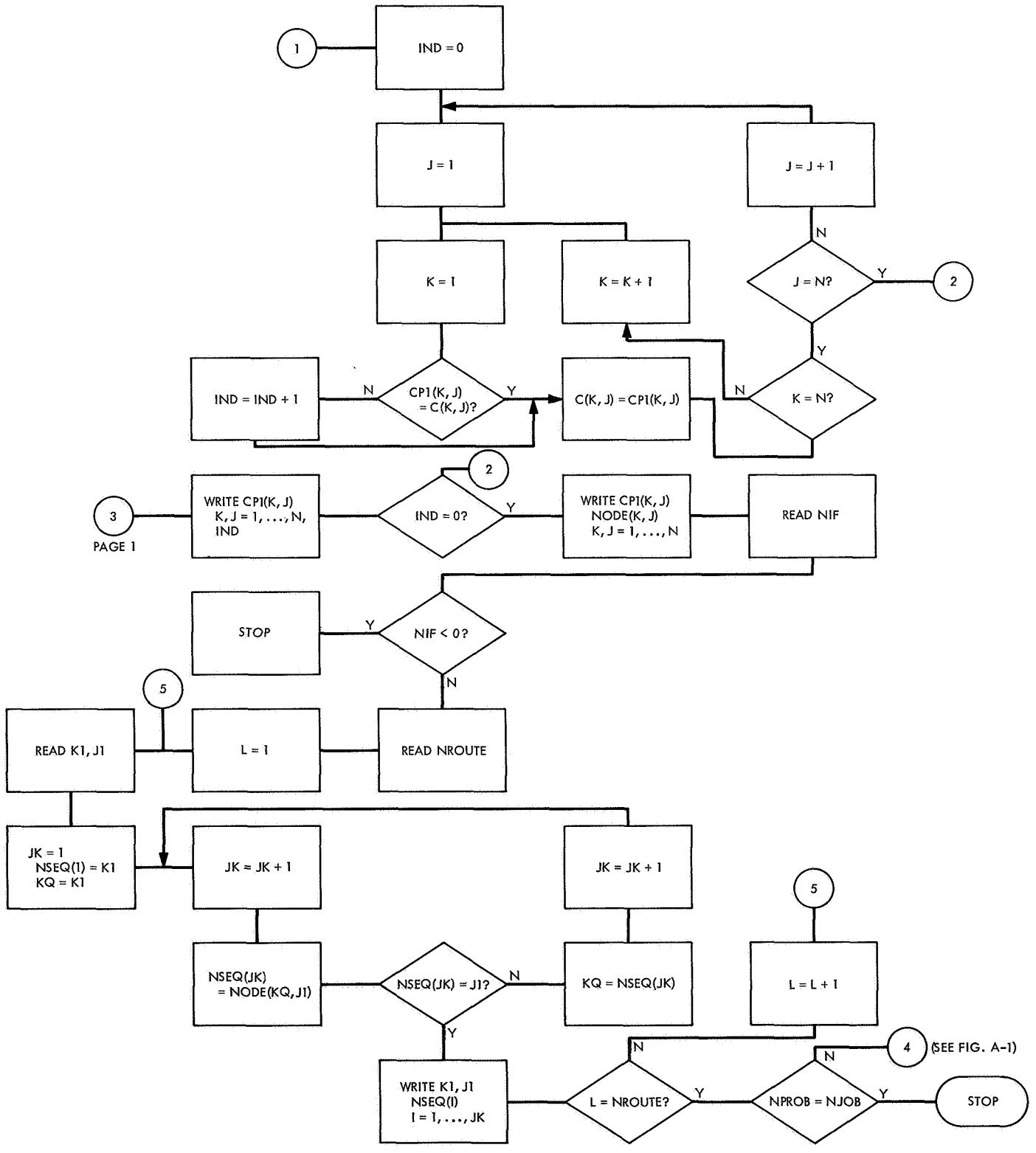
1
IND = 0
J = 1
J = J + 1
N
K = 1
K = K + 1
J = N?
Y
2
Y
IND = IND + 1
N
CP1(K, J) = C(K, J)?
Y
C(K, J) = CP1(K, J)
N
K = N?
2
3
PAGE 1
WRITE CP1(K, J) K, J = 1, . . . , N, IND
IND = 0?
Y
WRITE CP1(K, J) NODE(K, J) K, J = 1, . . . , N
READ NIF
STOP
Y
NIF < 0?
N
5
READ K1, J1
L = 1
READ NROUTE
JK = 1 NSEQ(1) = K1 KQ = K1
JK = JK + 1
JK = JK + 1
5
NSEQ(JK) = NODE(KQ, J1)
NSEQ(JK) = J1?
N
KQ = NSEQ(JK)
L = L + 1
Y
N
N
4
(SEE FIG. A-1)
WRITE K1, J1 NSEQ(I) I = 1, . . . , JK
L = NROUTE?
Y
NPROB = NJOB
Y
STOP

Fig. A-1 (contd)

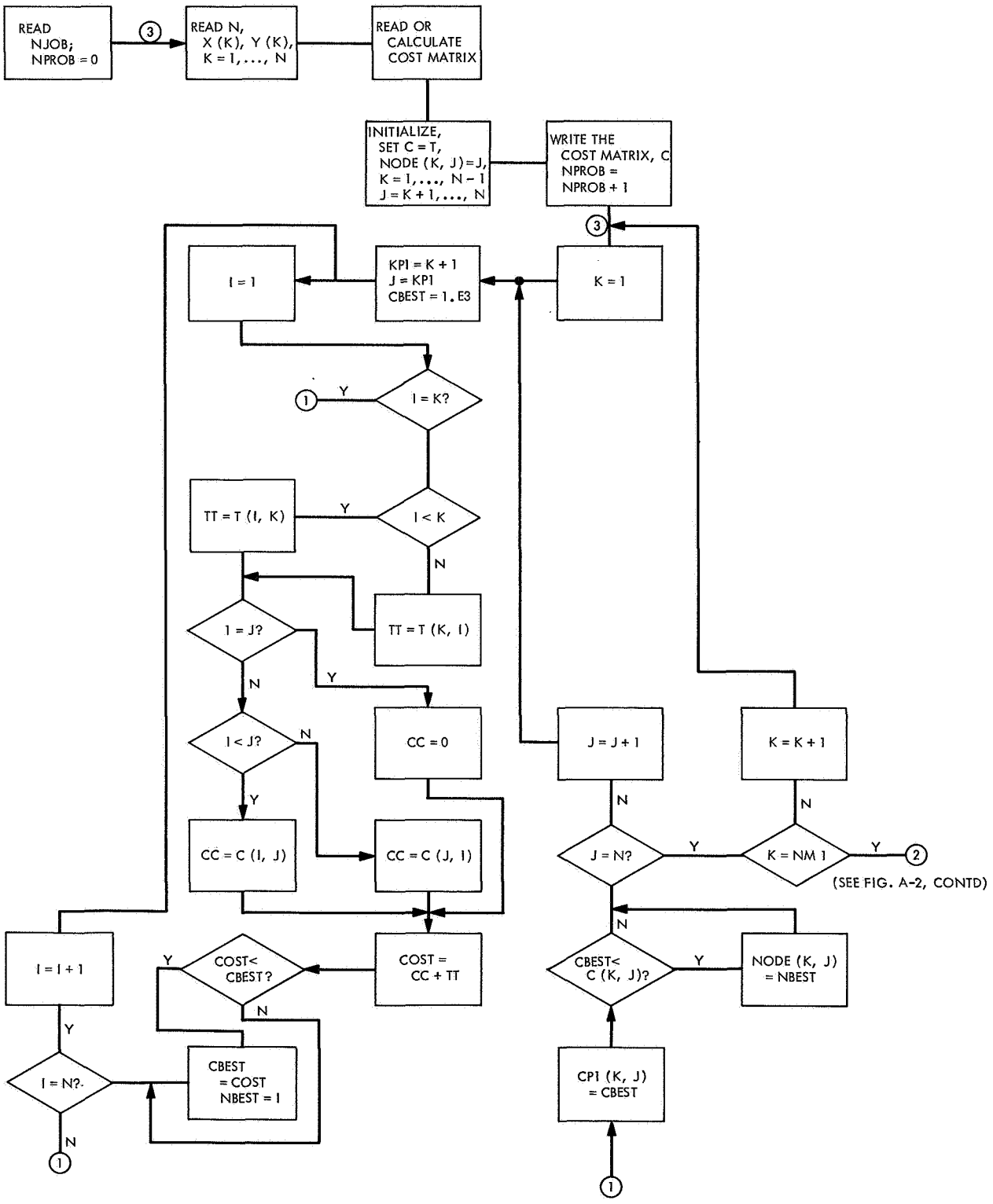

Fig. A-2. Optimal route—symmetric

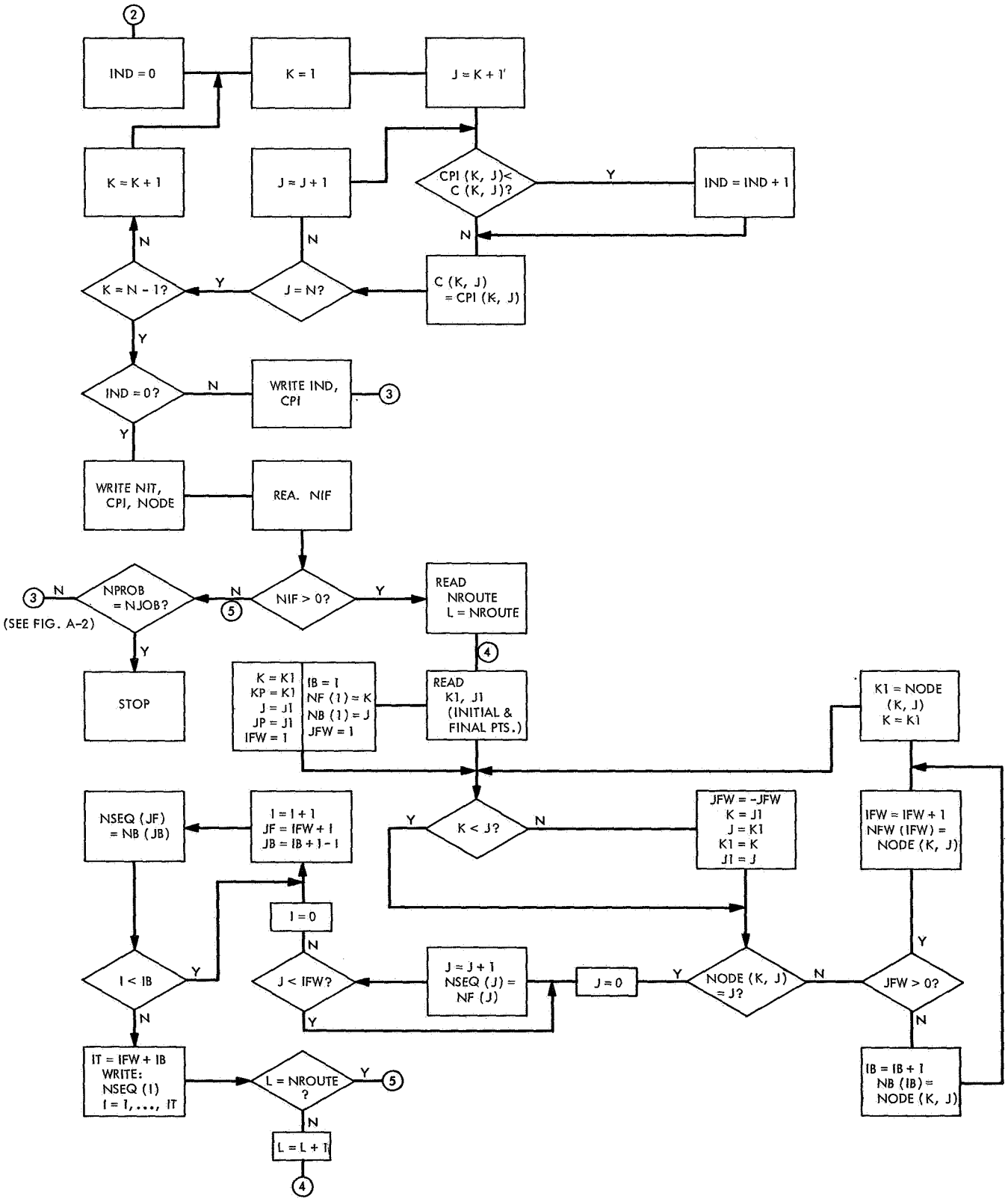
2
IND = 0
K = 1
J = K + 1
K = K + 1
J = J + 1
CPI (K, J)< C (K, J)?
Y
IND = IND + 1
N
N
N
K = N - 1?
Y
J = N?
C (K, J) = CPI (K, J)
Y
IND = 0?
N
WRITE IND, CPI
3
Y
WRITE NIT, CPI, NODE
REA. NIF
3
N
NPROB = NJOB?
N
5
NIF > 0?
Y
READ NROUTE L = NROUTE
(SEE FIG. A-2)
Y
4
STOP
K = K1 KP = K1 J = J1 JP = J1 IFW = 1
IB = 1 NF (1) = K NB (1) = J JFW = 1
READ K1, J1 (INITIAL & FINAL PTS.)
K1 = NODE (K, J) K = K1
NSEQ (JF) = NB (JB)
I = I + 1 JF = IFW + 1 JB = IB + 1 - 1
Y
K < J?
N
JFW = -JFW K = J1 J = K1 K1 = K J1 = J
IFW = IFW + 1 NFW (IFW) = NODE (K, J)
I = 0
Y
I < IB
Y
N
J < IFW?
J = J + 1 NSEQ (J) = NF (J)
J = 0
Y
NODE (K, J) = J?
N
JFW > 0?
N
IT = IFW + IB WRITE: NSEQ (I) I = 1, ..., IT
L = NROUTE ?
Y
5
N
L = L + 1
4
IB = IB + 1 NB (IB) = NODE (K, J)

Fig. A-2 (contd)

References

1. Lim, L. Y., *A Pathfinding Algorithm for a Myopic Robot*, Technical Report 32-1288. Jet Propulsion Laboratory, Pasadena, Calif., Aug. 1, 1968.

2. Bellman, R., and Kalaba, R., *Dynamic Programming and Modern Control Theory*, p. 50. Academic Press, New York, 1965.