1968-06

On network analysis

Bohls, Robert Joseph

Monterey, California. Naval Postgraduate School

https://hdl.handle.net/10945/40041

UNITED STATES
NAVAL POSTGRADUATE SCHOOL

ON NETWORK ANALYSIS

by

Robert Joseph Bohls

June 1968

[redacted]

[redacted]

ON NETWORK ANALYSIS

by

Robert Joseph Bohls
Captain, United States Army
B.S., University of Toledo, 1962

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 1968

Signature of Author [illegible] [redacted]

[redacted] [redacted]

Chai [redacted]

[illegible]

ABSTRACT

This thesis reviews four of the existing matrix methods for finding the shortest path in a network, including the little known matrix method by Floyd. Floyd's method is then extended to determine all best paths. After a brief review of the nth best path problem, Floyd's method is again extended to determine the nth best path. Finally, the nth best path problem is investigated to determine its applicability to the traveling salesman problem.

TABLE OF CONTENTS

LIST OF FIGURES

1. Introduction

Four matrix methods to determine the shortest path in a network will be presented. After an explanation of each method there will follow a simple example to illustrate the procedure.

First, the method of matrix manipulation [3] will be set forth. Next, an improvement on this, the Cascade algorithm [7], will be presented. The two algorithms that appear to be the most efficient are next. Algorithm 97 was published by Floyd [8] in 1962 and went unnoticed for some time. In 1965 Murchland [15] published what he called "a new method," but which in reality was the same as Floyd's. A different approach but equally as efficient as Floyd's was put forth in 1966 by Dantzig [6] and is an inductive procedure.

In many practical problems a shortest route is not sufficient. All alternate shortest routes or the n best routes are needed. Thus a simple extension to Floyd's algorithm to determine all alternate routes will be presented. Following this will be a brief review of the nth best path problem and a second extension to Algorithm 97. This extension will determine the n best paths from all nodes i to all nodes j.

A logical extension of this method of determining all n best paths is to the traveling salesman problem. Although this problem is not solved here, a possible approach is set forth.

2. Notation

A graph or network, denoted $G = (X,U)$, is composed of two sets X and U,

$$X = [i;\ i=1, \cdots , n]$$

$$U = [(i,j);\ i \in X \text{ and } j \in X].$$

X is the set of nodes or vertices and the pair $(i,j) \epsilon U$ is called an arc, where i is the initial node and j is the terminal node. Two nodes i and j are adjacent if they are distinct and there exists an arc $u=(i,j)$ or $v=(j,i)$. A path is a sequence $(u_1,u_2, \cdots)$ of arcs of a network (X,U) such that the terminal node of each arc coincides with the initial node of the succeeding arc. A finite path in which the initial node coincides with the terminal node is called a circuit. The length of a path is the sum of the lengths of the arcs forming the path.

The matrix M associated with the network is a vertex-vertex matrix.

$$M = [m_{ij}] ,$$

where the element m_{ij} is the length of the arc from node i to adjacent node j, if no such arc exists then m_{ij} is equal to infinity. The distance m_{ij} can be less than, equal to, or greater than zero, but the length of any circuit must be non-negative. In general, m_{ij} does not necessarily equal m_{ji} and m_{ij} does not have to satisfy the triangular inequality

$$m_{ij} \leq m_{ik} + m_{kj}.$$

The shortest distance matrix M^* has elements m^*_{ij} equal to the distance of the shortest path from i to j. Thus m^*_{ij} is the distance of the path from i to j, whose sum of arcs is at a minimum.

Following Murchland [15], the symbol ":=" shall denote "is replaced by."

Following Bellman and Kalaba [1],

$$\min_n (x_1,x_2, \cdots, x_p) = \text{the nth smallest value of}$$

the quantities x_i.

Symbols frequently used are defined below.

ND = the number of nodes in the network.

INF = infinity, and is taken to be greater than the maximum distance of any possible path.

All matrices in the sequel represent vertex-vertex distance matrices.

3. The Shortest Path by Matrix Manipulation

The first method of determining the shortest path in a network, that of matrix manipulation [3], involves two operations. These operations play the roles of elementary addition and multiplication respectively.

Define the sum of two matrices

$$C = A + B.$$

as

$$c_{ij} = \min(a_{ij}, b_{ij}), \tag{1}$$

where a_{ij} and b_{ij} are corresponding elements.

Furthermore, define the product of two matrices

$$C = AB$$

as

$$c_{ij} = \min_k(a_{ik} + b_{kj}). \tag{2}$$

M contains elements that are the arc distances between adjacent nodes. Following equation (2), M^2 contains elements that are equal to the shortest distance, from i to j, in a path containing exactly two arcs. In general M^k contains elements equal to the shortest distance, from i to j, in a path containing exactly k arcs.

Thus, the shortest distance matrix,

$$M^* = M + M^2 + \cdot \cdot \cdot + M^{n-1}, \tag{3}$$

contains elements equal to the minimum distance, from i to j, for all paths containing from one to n-1 arcs.

As an example, consider the network in figure 1 and its associated initial distance matrix M, given below.

$$M = \left|\begin{array}{c|c|c} 7 & 5 & 3 \\ 1 & INF & 4 \\ INF & 1 & INF \end{array}\right|$$

Using equation (2) we obtain for elements $m^2(1,1)$ and $m^2(1,2)$,

$$m^2(1,1) = \min(7 + 7, 5 + 1, 3 + INF) = 6$$

$$m^2(1,2) = \min(7 + 5, 5 + INF, 3 + 1) = 4.$$

After computing the rest of the elements in similar fashion, we obtain the following matrix.

$$M^2 = \left|\begin{array}{c|c|c} 6 & 4 & 9 \\ 8 & 5 & 4 \\ 2 & INF & 5 \end{array}\right|$$

From equation (3) and using addition as defined in equation (1), we obtain the following shortest distance matrix.

$$M^* = \left|\begin{array}{c|c|c} 6 & 4 & 3 \\ 1 & 5 & 4 \\ 2 & 1 & 5 \end{array}\right|$$

In general M^{n-1} is a matrix using exactly n-1 arcs and visiting all n nodes. M^n would be needed if a shortest circuit from a node back to itself was desired. In general

$$M^o = M^* + M^n$$

where M^o shows the length of the shortest circuit from a node back to itself. For the above example

$$M^3 = \left|\begin{array}{c|c|c} 5 & 10 & 8 \\ 6 & 5 & 9 \\ 9 & 6 & 5 \end{array}\right|$$

and

$$M^o = \left|\begin{array}{c|c|c} 5 & 4 & 3 \\ 1 & 5 & 4 \\ 2 & 1 & 5 \end{array}\right|$$

is the shortest circuit matrix.

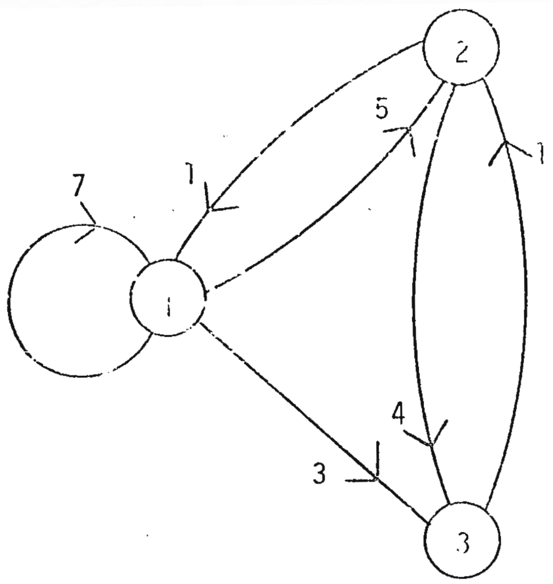


Figure 1

Network for Matrix Manipulation Example

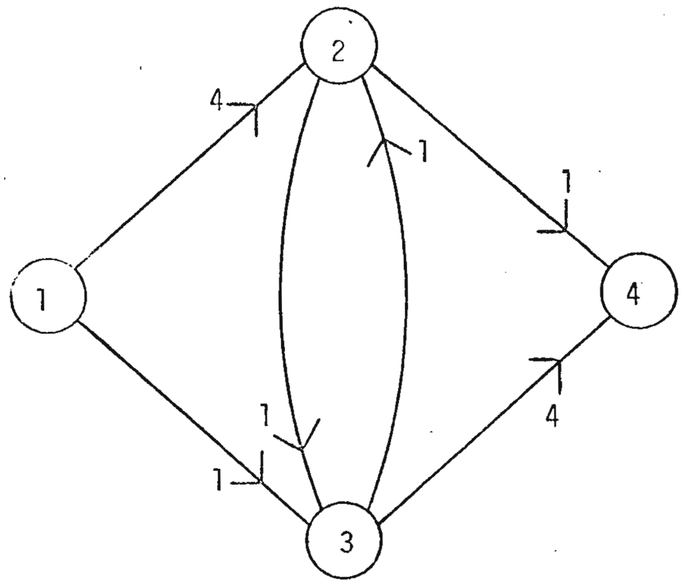


Figure 2

Network for Cascade Algorithm Example

4. The Cascade Algorithm

Whereas the previous method needed n-1 matrix operations, i.e. A^j for $j = 2, \cdots, n-1$ and one summation, the Cascade algorithm [7] needs only two matrix squarings. (This method is called the Cascade Algorithm by [7] and the Revised Matrix Algorithm by [12].)

In this method newly calculated elements immediately replace existing elements. Also, now the order of calculation becomes important. In the first squaring, the so-called forward Cascade process, the elements are calculated in the order $m_{11}, m_{12}, \cdots, m_{1n}; m_{21}, \cdots m_{2n}; \cdots; m_{n1}, \cdots m_{nn}$. Then in the second squaring, the so-called backward Cascade process, the elements are calculated in the order $m_{nn}, m_{n,n-1}, \cdots m_{n1}; m_{n-1,n}, \cdots m_{n-1,1}; \cdots; m_{1n}, \cdots m_{11}$.

Let $M^p = [m^p_{ij}]$, where

$$p = \begin{cases} 1 & \text{initially} \\ F & \text{at the end of the forward process} \\ * & \text{at the end of the backward process} \end{cases}$$

Thus $M^1 = M$, which is the initial arc matrix previously defined and M^* is still the shortest distance matrix.

M^F is obtained from M^1 by the following forward process.

$$m^F_{ij} := \min_k \left(m^q_{ik} + m^r_{kj}\right) \tag{4}$$

where

$$q = \begin{cases} 1 & k \geq j \\ F & k < j \end{cases}$$

and

$$r = \begin{cases} 1 & k \geq i \\ F & k < i. \end{cases} \tag{5}$$

M^* is then obtained from M^F by the following backward process.

$$m^*_{ij} := \min_k \left(m^q_{ik} + m^r_{kj}\right) \tag{6}$$

where

$$q = \begin{cases} F & k \leq j \\ * & k > j \end{cases}$$

and

$$r = \begin{cases} F & k \leq i \\ * & k > i. \end{cases} \qquad (7)$$

To illustrate the above procedure, consider the network in figure 2 and its associated initial distance matrix given below.

$$M = \left|\begin{array}{c|c|c|c} 0 & 4 & 1 & \text{INF} \\ \text{INF} & 0 & 1 & 1 \\ \text{INF} & 1 & 0 & 4 \\ \text{INF} & \text{INF} & \text{INF} & 0 \end{array}\right|$$

Beginning the forward process using equations (4) and (5),

$$m^F_{12} := \min \begin{bmatrix} m^F_{11} + m^1_{12} = 0 + 4 = 4 \\ m^1_{12} + m^1_{22} = 4 + 0 = 4 \\ m^1_{13} + m^1_{32} = 1 + 1 = 2 \\ m^1_{14} + m^1_{42} = \text{INF} + \text{INF} = \text{INF} \end{bmatrix} = 2 < m^1_{12}.$$

Since 2 is less than m^1_{12}, replacement would take place.

$$m^F_{13} := \min\ (0 + 1, 2 + 1, 1 + 0, \text{INF} + \text{INF}) = 1$$

Since $m^1_{13} = m^F_{13}$ no replacement need be done, although when using the computer, replacement would automatically take place.

$$m^F_{14} := \min\ (0 + \text{INF}, 2 + 1, 1 + 4, \text{INF} + 0) = 3 < m^1_{14}$$

Therefore $m^1_{14} := m^F_{14}$ would in fact take place. After computing the remaining elements and replacing when necessary, we obtain the following matrix.

$$M^F = \left|\begin{array}{c|c|c|c} 0 & 2 & 1 & 3 \\ \text{INF} & 0 & 1 & 1 \\ \text{INF} & 1 & 0 & 2 \\ \text{INF} & \text{INF} & \text{INF} & 0 \end{array}\right|$$

After performing the backward process using equations (6) and (7), we obtain the shortest distance matrix M* as shown.

$$M^* = \left|\begin{array}{c|c|c|c} 0 & 2 & 1 & 3 \\ \text{INF} & 0 & 1 & 1 \\ \text{INF} & 1 & 0 & 2 \\ \text{INF} & \text{INF} & \text{INF} & 0 \end{array}\right|$$

Thus we see that this is exactly the same as matrix squaring with the exception of replacing the elements once shorter paths are found. This replacing cuts down the number of operations performed tremendously, relative to the matrix manipulation method.

Narahari Pandit [16] first claimed that two forward processes were enough. However, [7] displayed a counterexample and proved one forward and one backward process was sufficient. Hu [12] later proved that three forward processes are sufficient.

5. An Inductive Approach

Dantzig [6] takes somewhat of a different approach to solve the shortest path problem. His inductive approach is described below.

Assume for nodes 1, 2, · · ·, k-1 that optimal distances m^o_{ij} are given. Optimal distances m^*_{ij} are desired for nodes 1, 2, · · ·, k.

For h = 1, · · ·, k - 1 and j = 1, · · ·, k - 1.

$$m^*_{kh} = \min_j (m_{kj} + m^o_{jh}) \qquad (8)$$

$$m^*_{hk} = \min_j (m_{jk} + m^o_{hj}) \qquad (9)$$

$$m^*_{kk} = \min_j (0, m^*_{kj} + m^*_{jk}) \qquad (10)$$

For i = 1,· · ·, k - 1 and j = 1, · · ·, k - 1

$$m^*_{ij} := \min (m^o_{ij}, m^*_{ik} + m^*_{kj}) \qquad (11)$$

Consider the network in figure 3 and its associated initial distance matrix given below.

$$M = \begin{vmatrix} 0 & 1 & 5 & \text{INF} \\ \text{INF} & 0 & 3 & 1 \\ \text{INF} & 5 & 0 & \text{INF} \\ \text{INF} & \text{INF} & 1 & 0 \end{vmatrix}$$

Assume we have found optimal distances m^o_{ij} for nodes 1, 2, 3 thus giving M^o.

$$M^o = \begin{vmatrix} 0 & 1 & 4 \\ \text{INF} & 0 & 3 \\ \text{INF} & 5 & 0 \end{vmatrix}$$

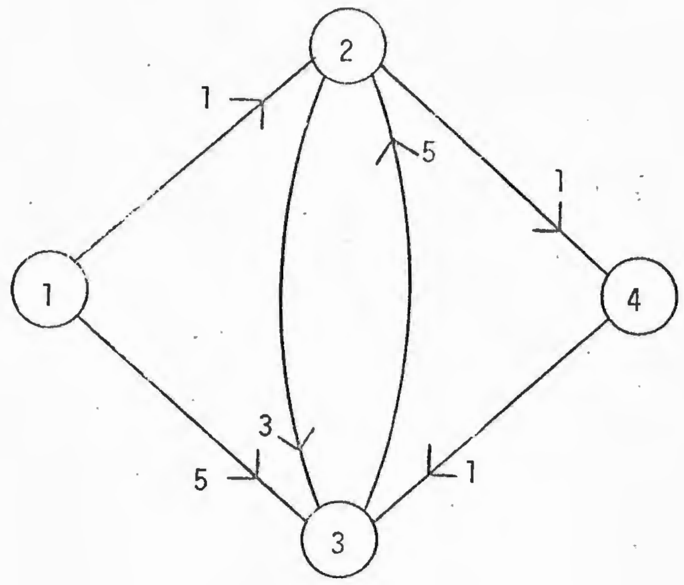


Figure 3

Network for Inductive Example

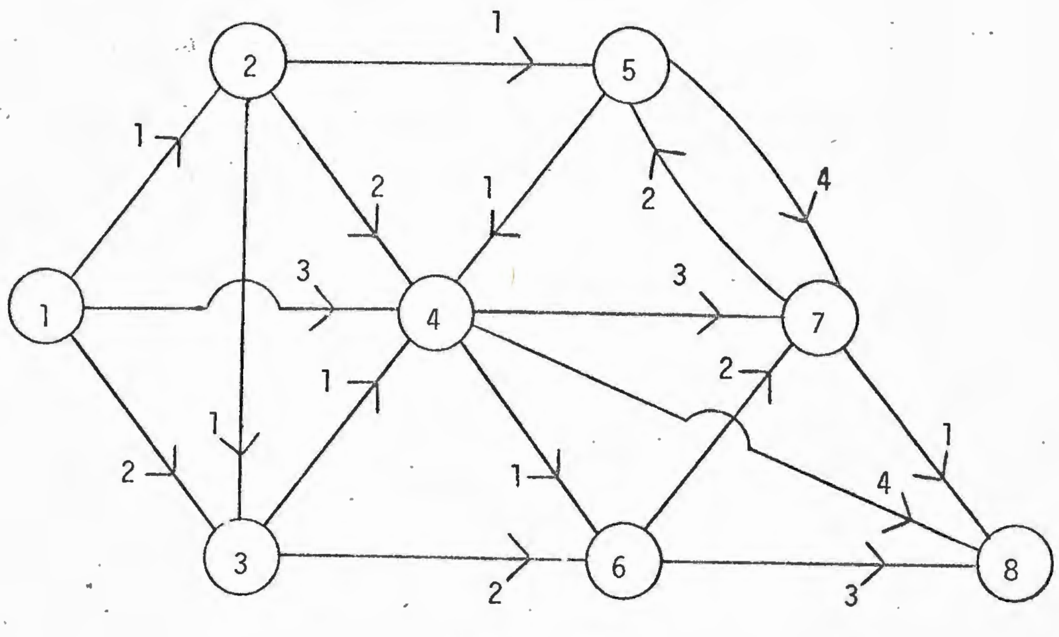


Figure 4

Network for Algorithm 97 Example

To obtain m^*_{ij} for nodes 1, 2, · · ·, k, we begin by using equation (8).

$$m^*_{4h} = \min \left[\begin{array}{l} m_{41} + m^o_{1h} \\ m_{42} + m^o_{2h} \\ m_{43} + m^o_{3h} \end{array} \right.$$

For h = 1,2,3

$$m^*_{41} = \min(\text{INF} + 0, \text{INF} + \text{INF}, 1 + \text{INF}) = \text{INF}$$

$$m^*_{42} = \min(\text{INF} + 1, \text{INF} + 0, 1 + 5) = 6$$

$$m^*_{43} = \min(\text{INF} + 4), \text{INF} + 3, 1 + 0) = 1$$

Concluding the calculations and replacements using equations (9), (10) and (11) we obtain M^*.

$$M^* = \left| \begin{array}{c|c|c|c} 0 & 1 & 3 & 2 \\ \text{INF} & 0 & 2 & 1 \\ \text{INF} & 5 & 0 & 6 \\ \text{INF} & 6 & 1 & 0 \end{array} \right|$$

6. Algorithm 97

Unnoticed for some time was the following algorithm by Floyd [8]. The two *'d statements were not in the original program but have been added to reduce operations. The program is written in FORTRAN IV.

```
      DO   1   I = 1, ND
      DO   1   J = 1, ND
      IF(M(J,I).EQ.INF) GO TO 1
     *IF(I.EQ.J)  GO TO 1
      DO   1   K = 1, ND
      IF(M(I,K).EQ.INF) GO TO 1
     *IF(I.EQ.K) GO TO 1
      MS = M(J,I) + M(I,K)
      IF(MS.GE.M(J,K)) GO TO 1
      M(J,K) = MS
1     CONTINUE
```

Flow diagram for the above program is contained in appendix I.

This is similar to the previous matrix squaring methods. It is not actually a matrix squaring but resembles it. It performs the same operation as the Cascade algorithm,

$$m_{jk} := \min_i (m_{jk}, m_{ji} + m_{ik})$$

but calculates m_{jk} in a different order. Thus, by this order needs only a single "squaring." Upon "squaring" m_{jk} is now m^*_{jk}.

After a brief informal discussion of the algorithm an example will be worked to illustrate the procedure.

The matrix M is searched, in an order to be described shortly, for non-infinity, non-main-diagonal elements. Two of these elements are added and compared with a third element. Replacement is then made if the sum is less than the third element.

The search is as follows. The columns of M are searched in order, i = 1, · · ·, ND. Each column i is searched in order, j = 1, · · ·, ND. When a non-infinity element (j, i) is encountered in column i, row i is searched in order, k = 1, · · ·, ND. For all non-infinity elements in row i, set

$$ms = m_{ji} + m_{ik}.$$

If $ms < m_{jk}$, then $m_{jk}{:=}ms$. Thus we have formed a shorter path from j to k by combining two paths j to i and i to k.

As an example consider figure 4 and the following associated initial distance matrix.

$$M = \left|\begin{array}{c|c|c|c|c|c|c|c}
0 & 1 & 2 & 3 & INF & INF & INF & INF \\
INF & 0 & 1 & 2 & 1 & INF & INF & INF \\
INF & INF & 0 & 1 & INF & 2 & INF & INF \\
INF & INF & INF & 0 & INF & 1 & 3 & 4 \\
INF & INF & INF & 1 & 0 & INF & 4 & INF \\
INF & INF & INF & INF & INF & 0 & 2 & 3 \\
INF & INF & INF & INF & 2 & INF & 0 & 1 \\
INF & INF & INF & INF & INF & INF & INF & 0
\end{array}\right|$$

Applying the algorithm to M we see that except for i = 1 = j, column 1 has no non-infinity elements. Incrementing i to 2, the first non-infinity column element encountered is $m_{ji} = m_{12}$. Now row 2 is searched and the first non-infinity element is $m_{ik} = m_{23}$. Thus for the first comparison,

$$1.\quad ms = m_{12} + m_{23}$$
$$= 1 + 1 = 2$$

Since $2 = ms = m_{13} = 2$,

no replacement is needed. The two remaining non-infinity elements in row 2 are the next elements utilized.

$$2.\quad ms = m_{12} + m_{24}$$
$$= 1 + 2 = 3$$
$$3 = ms = m_{14} = 3$$
$$m_{14} :\neq ms$$
$$3.\quad ms = m_{12} + m_{25}$$
$$= 1 + 1 = 2$$
$$2 = ms < m_{15} = \text{INF}$$
$$m_{15} := 2$$

For i = 3 the following two replacements would take place in order given.

$$4.\quad m_{16} := 4$$
$$5.\quad m_{26} := 3$$

After the remaining columns, i = 4, · · ·, 8, have been searched and required replacements made, the shortest distance matrix is obtained and is given below.

$$M^* = \left|\begin{array}{c|c|c|c|c|c|c|c}
0 & 1 & 2 & 3 & 2 & 4 & 6 & 7 \\
\text{INF} & 0 & 1 & 2 & 1 & 3 & 5 & 6 \\
\text{INF} & \text{INF} & 0 & 1 & 6 & 2 & 4 & 5 \\
\text{INF} & \text{INF} & \text{INF} & 0 & 5 & 1 & 3 & 4 \\
\text{INF} & \text{INF} & \text{INF} & 1 & 0 & 2 & 4 & 5 \\
\text{INF} & \text{INF} & \text{INF} & 5 & 4 & 0 & 2 & 3 \\
\text{INF} & \text{INF} & \text{INF} & 3 & 2 & 4 & 0 & 1 \\
\text{INF} & \text{INF} & \text{INF} & \text{INF} & \text{INF} & \text{INF} & \text{INF} & 0
\end{array}\right|$$

Floyd [8] gave no proof of his algorithm but relied for proof on a theorem on boolean matrices by Warshall [19]. Hu [11] gives a somewhat simpler proof of Floyd's algorithm. Murchland [15] developed a similar algorithm, unaware of Floyd's accomplishment, and presents a somewhat lengthly proof in his paper.

If we are interested in not only the distance of the shortest path but the route as well, something more than just M^* is needed. One way to obtain the path is to define a new matrix, called the routing matrix,

$$MR = [mr_{jk}] ,$$

where mr_{jk} contains the number of the first node, after j, on the path from j to k. Initially,

$$mr_{jk} = k, \quad \text{for } j, k = 1, \cdot \cdot \cdot, n.$$

During the calculation, whenever

$$m_{jk} := m_{ji} + m_{ik} ,$$

the following replacement is also made,

$$mr_{jk} := mr_{ji} .$$

In the previous example the first replacement is done in comparison 3. Thus

$$m_{15} := 2 ,$$

would be followed by

$$mr_{15} := mr_{12} = 2 .$$

At the conclusion of the above example the routing matrix would be as shown as follows.

$$MR = \left|\begin{array}{c|c|c|c|c|c|c|c}
1 & 2 & 3 & 4 & 2 & 3 & 4 & 4 \\
1^* & 2 & 3 & 4 & 5 & 3 & 4 & 4 \\
1^* & 2^* & 3 & 4 & 4 & 6 & 4 & 4 \\
1^* & 2^* & 3^* & 4 & 7 & 6 & 7 & 8 \\
1^* & 2^* & 3^* & 4 & 5 & 4 & 7 & 4 \\
1^* & 2^* & 3^* & 7 & 7 & 6 & 7 & 8 \\
1^* & 2^* & 3^* & 5 & 5 & 5 & 7 & 8 \\
1^* & 2^* & 3^* & 4^* & 5^* & 6^* & 7^* & 8
\end{array}\right|$$

The elements with the stars are elements as they were initialized. No replacement took place since no finite path exists, as can be seen in M^*. In the computer program these starred elements would be set to infinity to facilitate reading MR.

To illustrate the procedure of extracting a path from MR, the path from node 1 to node 8 would be

$$mr_{18} = 4$$

$$mr_{48} = 8$$

or

1 to 4 to 8.

It may be observed from the example that more than one shortest path exists from 1 to 8. Algorithm 97 will only give the first best path. The modification required to extract all alternate paths shall be treated in the following section.

The above procedure of using the routing matrix to determine the paths is applicable, not only to Algorithm 97 but to all four matrix methods.

7. Comparison of Methods

Already some idea exists of the relative efficiency of the four methods discussed. Simply in terms of number of matrix squarings, the matrix manipulation method requires n-1, the Cascade algorithm two, and Algorithm 97 one. To more critically compare all four methods let's consider the basic operations involved. A matrix multiplication between two matrices of order n requires three basic operations. These operations are indexing, addition, and comparison. Below is a listing of the four methods and next to each, the number of each of the basic operations required.

Matrix Manipulation	- $n^3 \log (n-1)$
Cascade Algorithm	- $2n^3$
Induction Method	- $n(n-1)(n-2)$
Algorithm 97	- $n(n-1)(n-2)$

8. All Alternate Best Paths

In practical problems it may arise that all alternate best paths are wanted. A decision maker may want to evaluate all best paths and select from them by means of some other criterion. To extract all best paths Algorithm 97 must be modified slightly.

For the shortest path problem we needed two ND x ND matrices, M and MR. For this problem M will remain the same. Recall that mr_{jk} contains the number of the first node, after j, on the path from j to k. Now since we want to keep all best paths we must go to a three-dimensional matrix. The third dimension on MR is not governed by the number of best paths from j to k but by the number of first nodes on the path from j to k. A number of paths may share the same first node and later branch off. Consider the paths u = (1-2-4-7-8) and v = (1-2-4-6-8). Node 2 need only be stored once, since paths u and v coincide from node 1 to node 4. Indication of a branch would occur at node 4. Here two first nodes would appear, node 6 and node 7.

A priori it is not known how many best paths there may be from j to k. Thus it also is not known how many first nodes are needed. This is of no real concern if calculations are going to be done with pencil and paper. For the computer, however, a specific number is needed. Inspection of the given network can give an indication of the maximum number of first nodes expected. Let NA denote this maximum number. Surely

$$NA \leq ND.$$

Thus MR would be dimensioned ND x ND x NA and mr_{jki} would contain the number of the ith first node, after j, on a best path from j to k. In the computer program, to facilitate printing out the paths and reading the matrices, mr_{jkh} is set to infinity for h = 2, · · ·, NA. Also, upon completion of the algorithm, for all elements m_{jk} equal to infinity, the corresponding elements mr_{jkl} are set to infinity. Thus the routing matrix will also indicate all non-existing paths.

For the computer program in appendix III, a value was read in for NA. If during execution of the program the number of first nodes becomes equal to NA, an error message will be printed and execution will cease. This will prevent the computer from attempting to address either outside the matrix or wrong elements in the matrix. Since computer time may be costly the program could be altered so as to ignore all additional first nodes once the total number exceeded NA. An error message would still be desired to indicate that alternate paths may have been thrown away.

Now the modification of Algorithm 97 will be explained to enable us to extract all best paths. Departure from the main algorithm occurs at the point where path (j,k) is compared with the sum of the two paths (j,i) and (i,k). If the sum is greater, that path, as before, is ignored. Now let us consider separately the cases where the sum is less than and where it is equal to m_{jk}.

For the case "less than," a path has been found that is shorter than the existing path. This is the same as for the shortest path. Now, however,

$$mr_{jkp} := mr_{jip} \quad p = 1, \cdot \cdot \cdot, NA$$

In the previous section for the shortest path, p was equal to one. Now there may already exist more than one first node at mr_{jip}. Therefore they all must be transferred to mr_{jkp}. (It is true that there may not be NA first nodes at mr_{jip}. The transfer could be handled a number of ways on the computer. One way would be to determine which of the NA elements are first nodes and transfer only them. The other way is as above, transfer all NA elements. Whichever is more efficient would depend on the particular matrix.)

For the case "equal to," a path has been found that is the same distance as the existing path. This we want to keep. Now a comparison must be made between the first nodes of mr_{jkp} and mr_{jiq}. Only first nodes of mr_{jiq} that are distinct from the first nodes of mr_{jkp} will be transferred to mr_{jkp}. Redundancy of first nodes would only take up storage space and no paths will be thrown away by the discarding.

As an example consider the network in figure 4. M and M* remain unchanged. The routing matrix would now be as given below.

$$MR = \left\|\begin{array}{c|c|c||c|c|c||c|c|c||c|c|c||c|c|c||c|c|c||c|c|c||c|c|c}
1 & & & 2 & & & 3 & 2 & & 4 & 2 & 3 & 2 & & & 3 & 2 & 4 & 4 & 2 & 3 & 4 & 2 & 3 \\
INF & & & 2 & & & 3 & & & 4 & 3 & 5 & 5 & & & 3 & 4 & 5 & 4 & 3 & 5 & 4 & 3 & 5 \\
INF & & & INF & & & 3 & & & 4 & & & 4 & 6 & & 6 & 4 & & 4 & 6 & & 4 & 6 & \\
INF & & & INF & & & INF & & & 4 & & & 7 & 6 & & 6 & & & 7 & 6 & & 8 & 6 & 7 \\
INF & & & INF & & & INF & & & 4 & & & 5 & & & 4 & & & 7 & 4 & & 4 & 7 & \\
INF & & & INF & & & INF & & & 7 & & & 7 & & & 6 & & & 7 & & & 8 & 7 & \\
INF & & & INF & & & INF & & & 5 & & & 5 & & & 5 & & & 7 & & & 8 & & \\
INF & & & INF & & & INF & & & INF & & & INF & & & INF & & & INF & & & 8 & &
\end{array}\right\|$$

The columns of MR are divided by double lines. Between the double lines are the elements of the third dimension, the first, second and third first nodes of the alternate paths from j to k. It can be shown that there are 25 alternate paths from node 1 to node 8.

Appendix II shows the flow diagram for the algorithm. The full computer program is shown in appendix III.

9. Nth Best Path

A generalization of the shortest path problem is the nth best path problem. It may arise that the shortest path(s) may not be the "best" path when evaluated by some other criterion. A decision maker may be willing to deviate from the shortest path(s) by a certain amount to avoid undesirable aspects of the shortest path(s). Thus the nth best path becomes important.

First a brief review of the known methods is in order. The simplest method is given by Pollack [17]. The network must first be solved for the shortest path, say u. Actually all shortest paths, u_p, must be found. Once given u_p, from a given node j to a given node k, composed of r_p arcs, the length of each arc in u_p is set, in turn, to infinity. The shortest path problem is solved (at most) r_p times for each u_p, once for each arc set to infinity. (An arc appearing in more than one u_p need be set to infinity only once.) Thus the maximum number of shortest path problems to be solved is equal to the sum of the r_p's. The shortest of these new best paths is the second best path, say v. There may be more than one second best path. Thus v_q will denote the qth second best path and will have s_q arcs.

For the third best path, the length of each arc of u_p and v_q is set, in turn, to infinity. Extending this to the nth best path, the length of each arc of all first best, second best, $\cdot\cdot\cdot$, n-1 best paths must in turn be set to infinity. As before, for each arc set to infinity a shortest path problem must be solved. By now it must be apparent that the number of shortest path problems to be solved may quickly become astronomically large. Thus this method appears to be good only when r and s are small and when p and q are very small, hopefully one.

The method of Bock, Kanter, and Haynes [5] differs from Pollack's in that the shortest path need not be known a priori. Using stems and trees, this method simply is a systematic listing of all paths from a given j to a given k. Again, however, this method is limited to small networks.

As in the two previous, the method of Hoffman and Pavley [10] determines the nth best path from a given j to a given k. However, now not only the knowledge of the shortest path from j to k is required, but the knowledge of all shortest paths from j to all other nodes in the network. Then by deviations from all of the shortest paths the nth best path is determined. Thus we have a problem similar to the one in Pollack's method. Pollack's method, however, being a shortest path problem generates paths with no loops. Such is not the case with this method. Paths with loops may well emerge as nth best paths. How much of a problem this is depends on the particular results desired.

The limitation to small networks is escaped by Bellman and Kalaba's [1] method. Here the nth best path is determined from all nodes j of the network to a given node k. This can be considered an advantage or disadvantage depending on what paths are wanted. This method uses the functional equation technique of dynamic programming. Restrictions are imposed in the method to insure loopless paths.

Floyd's algorithm provides a basis for the development of the following matrix method to determine the nth best path. The method of searching the matrix remains the same but now additional calculations are performed. Upon completion of this method the first, second, $\cdots$, nth best paths are given from all nodes j, to all nodes k. The only restrictions placed on the size of the network, or how large n can be, is the amount of computer storage available. The description of the method will be given below with the flow diagram given in appendix IV and the full computer program in appendix V.

For the method to determine all alternate best paths the matrix M was unchanged, but MR was increased to a three-dimensional matrix. To accommodate the nth best path both matrices must be three-dimensional. M and MR will be dimensioned ND x ND x NK. NK is equal to the predetermined value of n. Now, m_{jkp} will contain the distance of the pth best path from j to k and mr_{jkp} will contain the first node of the pth best path from j to k.

To determine the nth best path from j to k, the following operation is performed.

$$m^*_{jkn} := \min[m_{jkn}, \min_n (m_{jip} + m_{ikq})]$$
$$p = 1, \cdot \cdot \cdot, NK$$
$$q = 1, \cdot \cdot \cdot, NK$$

Essentially this operation forms all combinations of paths from j to k through i. Since not all of the NK p's or q's may be non-infinity, there will be a maximum of $(NK)^2$ of these paths. The path through i with the nth best distance is compared with the existing path, m_{jkn}. The minimum of these two paths is m^*_{jkn}. The m_{jkn} replaced by m^*_{jkn} is not simply discarded but becomes the $m_{jk,n+1}$ best distance. (It can be observed that when n, p and q are all one, the above operation reduces to the same operation originally described in the Cascade Algorithm and Algorithm 97.)

To illustrate the procedure consider the network in figure 5 and its associated initial distance matrix given below.

$$M = \left|\begin{array}{c|c|c|c|c} 0 & 3 & 4 & 9 & INF \\ INF & 0 & 6 & INF & 7 \\ INF & INF & 0 & 4 & INF \\ INF & INF & INF & 0 & 8 \\ INF & INF & 9 & 13 & 0 \end{array}\right|$$

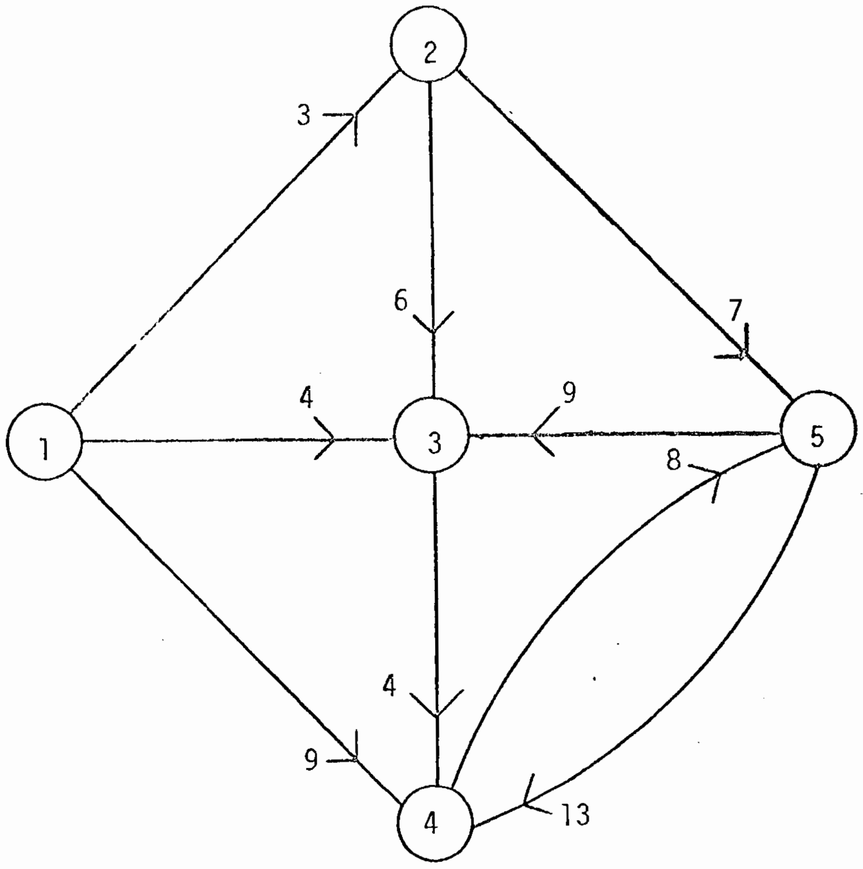
The first non-infinity column element encountered is $m_{jin} = m_{121}$. Searching row 2, the first non-infinity element is $m_{ikn} = m_{231}$. Following is the first set of comparisons.

1.1 $ms = m_{121} + m_{231} = 3 + 6 = 9 > m_{131} = 4$

$m_{131} :\neq ms$

1.2 $ms = m_{121} + m_{231} = 9 < m_{132} = INF$

$m_{132} := 9$



Fiaure 5

Network for Nth Best Path Example

Continuing across row 2,

$$2.1 \quad ms = m_{121} + m_{251} = 3 + 7 = 10 < m_{151} = \text{INF}$$

$$m_{151} := 10.$$

Following are the steps for column 3.

$$3.1 \quad ms = m_{131} + m_{341} = 4 + 4 = 8 < m_{141} = 9$$

$$m_{142} := m_{141} = 9$$

$$m_{141} := 8$$

$$3.2 \quad ms = m_{132} + m_{341} = 9 + 4 = 13 > m_{142} = 9$$

$$m_{142} :\neq ms$$

$$3.3 \quad ms = m_{132} + m_{341} = 13 < m_{143} = \text{INF}$$

$$m_{143} := 13$$

$$4.1 \quad ms = m_{231} + m_{341} = 6 + 4 = 10 < m_{241} = \text{INF}$$

$$m_{241} := 10$$

$$5.1 \quad ms = m_{531} + m_{341} = 9 + 4 = 13 = m_{541}$$

$$m_{541} :\neq ms$$

At this point the matrix would be as given below.

$$M = \left\| \begin{array}{c|c|c||c|c|c||c|c|c||c|c|c||c|c|c} 0 & & & 3 & & & 4 & 9 & & 8 & 9 & 13 & 10 & & \\ \text{INF} & & & 0 & & & 6 & & & 10 & & & 7 & & \\ \text{INF} & & & \text{INF} & & & 0 & & & 4 & & & \text{INF} & & \\ \text{INF} & & & \text{INF} & & & \text{INF} & & & 0 & & & 8 & & \\ \text{INF} & & & \text{INF} & & & 9 & & & 13 & & & 0 & & \end{array} \right\|$$

Upon completion of the algorithm, M^* would appear as below.

$$M^* = \left\| \begin{array}{c|c|c||c|c|c||c|c|c||c|c|c||c|c|c} 0 & & & 3 & & & 4 & 9 & 19 & 8 & 9 & 13 & 10 & 16 & 17 \\ \text{INF} & & & 0 & & & 6 & 16 & 27 & 10 & 20 & 31 & 7 & 18 & \\ \text{INF} & & & & & & 0 & 21 & & 4 & 25 & & 12 & & \\ \text{INF} & & & & & & 17 & & & 0 & 21 & & 8 & & \\ \text{INF} & & & & & & 9 & & & 13 & & & 0 & 21 & \end{array} \right\|$$

As before, the columns of the matrices are divided by double lines. Between the double lines are the three elements of the third dimension, the first, second and third best distances from j to k.

Proof of the nth best path algorithm relies on the proof of Algorithm 97. For if Algorithm 97 is valid, it must be true that all possible combinations of paths are enumerated and only the shortest retained. If Algorithm 97 fails to consider even one path from j to k, then no claim can be made of its validity. For there can be no guarantee that this one neglected path is not the shortest. The proof of Algorithm 97 has been sufficiently established by Warshall [19], Murchland [15] and Hu [11]. Thus, all possible combinations of paths from j to k are considered and only the shortest kept. For the nth best path algorithm, the remaining paths are not discarded. All paths are essentially placed in order and stored.

The procedure employed in constructing MR to enable the path to be enumerated is the same as before. First node replacement occurs immediately after the distance replacement. Considering the preceding example, replacements in MR would occur as follows.

1.2	$mr_{132} := mr_{121} = 2$
2.1	$mr_{151} := mr_{121} = 2$
3.1	$mr_{142} := mr_{141} = 4$
	$mr_{141} := mr_{131} = 3$
3.3	$mr_{143} := mr_{132} = 2$

At this point the first row of MR would appear as the following.

$$\left\| \begin{array}{ccc|ccc|ccc|ccc|ccc} 1 & & & 2 & & & 3 & 2 & & 3 & 4 & 2 & 2 & & \end{array} \right\|$$

Following is MR as it exists upon completion of the algorithm.

$$MR = \left\| \begin{array}{c|c|c||c|c|c||c|c|c||c|c|c||c|c|c} 1 & & & 2 & & & 3 & 2 & 2 & 3 & 4 & 2 & 2 & 3 & 4 \\ & & & 2 & & & 3 & 5 & 3 & 3 & 5 & 3 & 5 & 3 & \\ & & & & & & 3 & 4 & & 4 & 4 & & 4 & & \\ & & & & & & 5 & & & 4 & 5 & & 5 & & \\ & & & & & & 3 & & & 4 & & & 5 & 4 & \end{array} \right\|$$

Extraction of the path from MR is somewhat different than in the previous examples. The first node of the nth best path from j to k will not necessarily be found in the nth, third dimensional element. Consider the nth best path u = (j, · ·, p, · ·, q, · ·, k). The first node of the path u would be mr_{jkn} and found in the nth element. Succeeding first nodes may be found in the nth element until say the pth node is reached. Now the first node of the path from p to k may be in the n-1st element, indicating that this is now the n-1st best path from p to k. Continuing, the path from q to k may be the first best path. Thus, the distance matrices (including original) must be used in conjunction with the routing matrices as the path is traced from j to k. Upon reaching the first node of u, mr_{jkn}, the distance of the arc from j to mr_{jkn} must be subtracted from the total distance of u. The elements of $m^*_{mr_{jkn},k}$ are now searched for this lesser distance. The third-dimensional element will indicate what best path we shall continue on.

Using matrices M, M^*, and MR let us determine the second best path from node 1 to node 5. From mr_{152} we obtain the first node as 3. Subtracting the original m_{131} from the total distance of the path $m_{152} = 16$, we obtain $m_{35i} = 12$. Searching m_{35i} we find $i = 1$, meaning from now on we are on the first best path so no more subtracting is necessary. Continuing, $mr_{351} = 4$ and lastly $mr_{451} = 5$. Thus, the second best path from 1 to 5 is $u = (1, 3, 4, 5)$ and has a distance of 16 units.

10. The Traveling Salesman Problem

Algorithm 97 determines the shortest path from j to k. The first extension discussed allowed us to expose all possible minimum paths. In both cases m_{jj} was initialized to zero. If, instead, m_{jj} was set to infinity, we could have extracted the minimum circuits. Extending this, if m_{jj} was set to infinity in the nth best path algorithm, we would have obtained the nth best circuit from j to j, for all j. If the nth best circuit contains all the nodes of the network and the only node appearing more than once is j, which appears exactly twice, first and last, that circuit is defined as a tour and is a contender for the optimal solution to the traveling salesman problem. Thus, the traveling salesman problem modeled as a nth best path problem is next to be investigated.

The nth best path formulation in the previous section is not a minimum path formulation, as is Pollack's. Thus, paths with loops can be expected. This could be remedied by the insertion into the computer program of a subroutine called, say, NOLOPS. Each time a new path was formed by adding two paths, NOLOPS would construct the path and discard it if it obtained a loop.

Thus we are assured of loopless paths. Recall that alternate nth best paths were discarded. Only the first nth best path was retained. There is no way of knowing if that discarded alternate nth best path was optimal. A way of fixing this would be to compare alternate paths as they arise and retain the one with the most nodes. But this is not really correct. For that alternate nth best path thrown away may have combined with some other path and formed an optimal circuit. Thus, all alternate nth best paths must be retained. Now, all alternate nth best combinations of paths must be considered. This could be accomplished by adding a forth-dimension to the matrices M and MR. Each would be dimensioned ND x ND x NK x NA and m_{jknp} would be the pth alternate nth best path from j to k. Upon completion of the algorithm the n best circuits from j to j would be searched and the first tour encountered would be an optimal solution.

These modifications would seem to overcome the difficulties of the nth best formulation. The optimal solution would be an exact solution to the traveling salesman problem. It is anticipated, however, that to handle large networks in this way a computer would have to satisfy the following conditions.

1. Handle four dimensions
2. Have a storage capacity larger than even the present day advanced computers
3. Be faster than the present day computers

BIBLIOGRAPHY

1. Bellman, R. and Kalaba, R., "On kth Best Policies," Journal of the Society for Industrial and Applied Mathematics, Vol. 8, No. 4 (1960), pp. 582-588.

2. Bellmore, M. and Nemhauser, G. L., "The Traveling Salesman Problem: A Survey," The Johns Hopkins University.

3. Berge, C., The Theory of Graphs and its Applications, translated by A. Doig, Methuen, London and John Wiley, New York (1962), pp. 138-140.

4. Berge, C. and Ghouila-Houri, A., Programming, Games and Transportation Networks, translated by M. Merrington and C. Ramanujacharyulu, Methuen (1965), pp. 175-182.

5. Bock, F., Kantner, H. and Haynes, J., "An Algorithm (The Nth Best Path Algorithm) for Finding Ranking Paths through a Network," Research Report, Armour Research Foundation, Chicago (November 1957).

6. Dantzig, G. B., "All Shortest Routes in a Graph," Technical Report No. 66-3, Stanford University, Operations Research House (November 1966).

7. Farbey, B. A., Land, A. H. and Murchland, J. D., "The Cascade Algorithm for Finding all Shortest Distances in a Directed Graph," Management Science, Vol. 14, No. 1 (1967), pp. 19-28.

8. Floyd, R. W., "Algorithm 97: Shortest Path," Communications of the Association for Computing Machinery, Vol. 5 (1962), pp. 345.

9. Ford, L. R., Jr. and Fulkerson, D. R., Flows in Networks, Princeton University Press (1962), pp. 130.

10. Hoffman, W. and Pavley R., "A Method for the Solution of the Nth Best Path Problem," Journal of the Association for Computing Machinery, Vol. 6, No. 4 (1959), pp. 506.

11. Hu, T. C., "A Decomposition Algorithm for Shortest Paths in a Network," Operations Research, Vol. 16, No. 1 (1968), pp. 92-102.

12. Hu, T. C., "Revised Matrix Algorithms for Shortest Paths," SIAM Journal of Applied Mathematics, Vol. 15, No. 1 (1967), pp. 207-218.

13. Land, A. H. and Stairs, S. W., "The Extension of the Cascade Algorithm to Large Graphs," Management Science, Vol. 14, No. 1 (1967), pp. 29-33.

14. Murchland, J. D., "Bibliography of the Shortest Route Problem," LBS-TNT-6, London Business School, August 1967, (Revision).

15. Murchland, J. D., "A New Method for Finding all Elementary Paths in a Complete Directed Graph," LBS-TNT-22, London Business School (1965).

16. Narahari Pandit, S. N., "The Shortest Route Problem - An Addendum," Operations Research, Vol. 9, (1961), pp. 129-132.

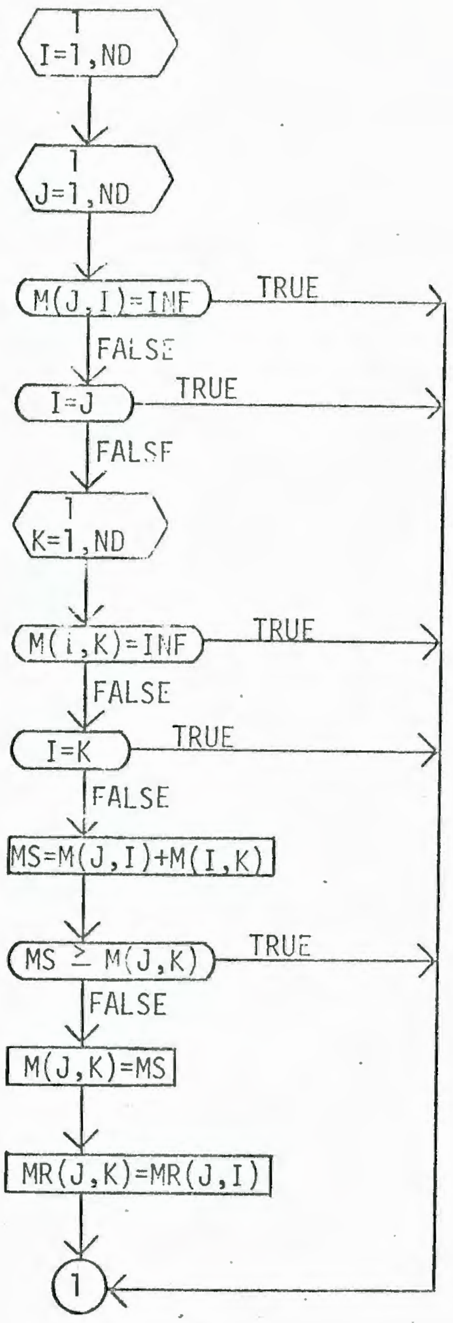
17. Pollack, M., "The kth Best Route through a Network," Operations Research, Vol. 9, No. 4 (1961), pp. 578-580.

18. Pollack, M., "Solutions of the kth Best Route through a Network - a Review," Journal of Mathematical Analysis and Applications, Vol. 3 (1961), pp. 547-559.

19. Warshall, S., "A Theorem on Boolean Matrices," Journal of the Association for Computing Machinery, Vol. 9 (1962), pp. 11.

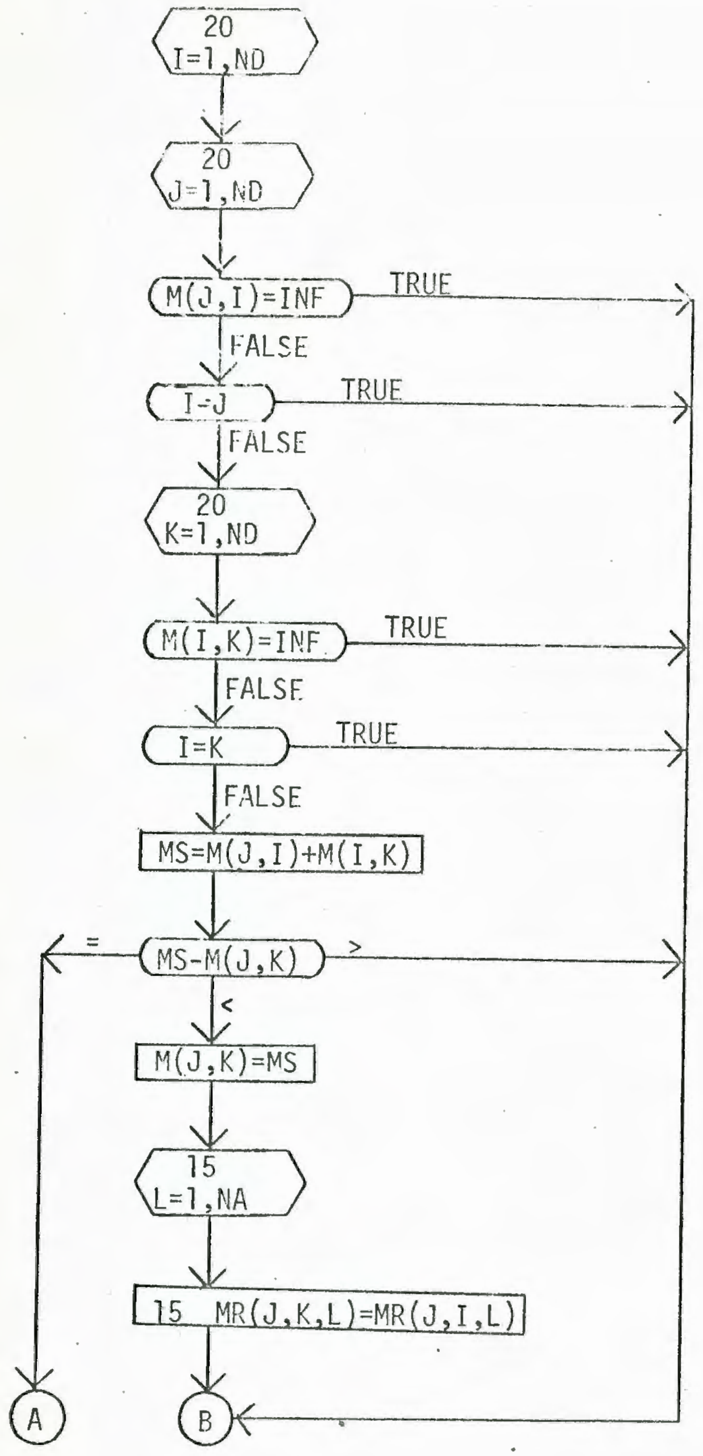
APPENDIX I

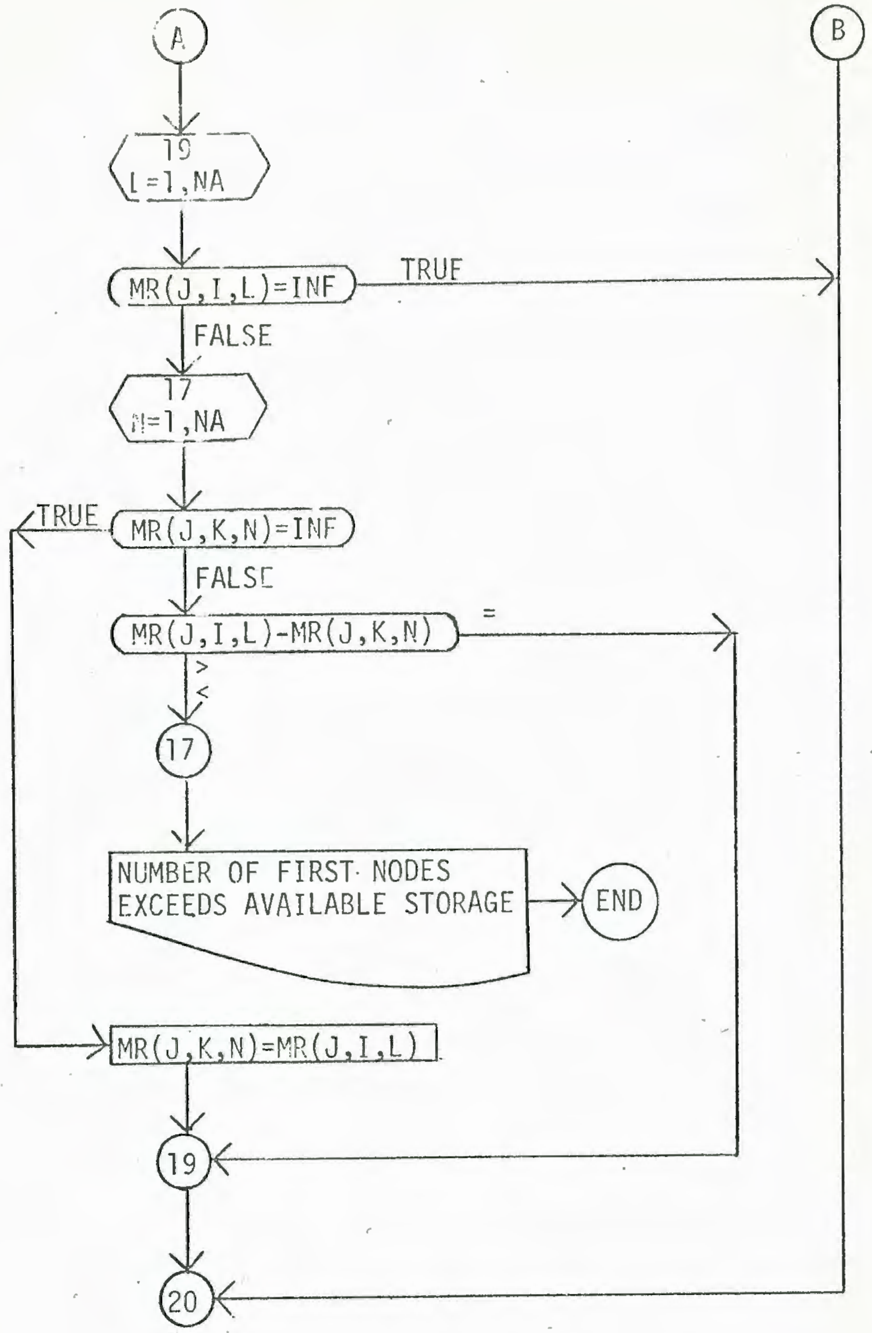
Flow Diagram for the Shortest Path Algorithm



APPENDIX II

Flow Diagram for the All Alternate Path Algorithm





A
B
19
L=1,NA
MR(J,I,L)=INF
TRUE
FALSE
17
N=1,NA
MR(J,K,N)=INF
TRUE
FALSE
MR(J,I,L)-MR(J,K,N)
=
>
<
17
NUMBER OF FIRST NODES EXCEEDS AVAILABLE STORAGE
END
MR(J,K,N)=MR(J,I,L)
19
20

APPENDIX III

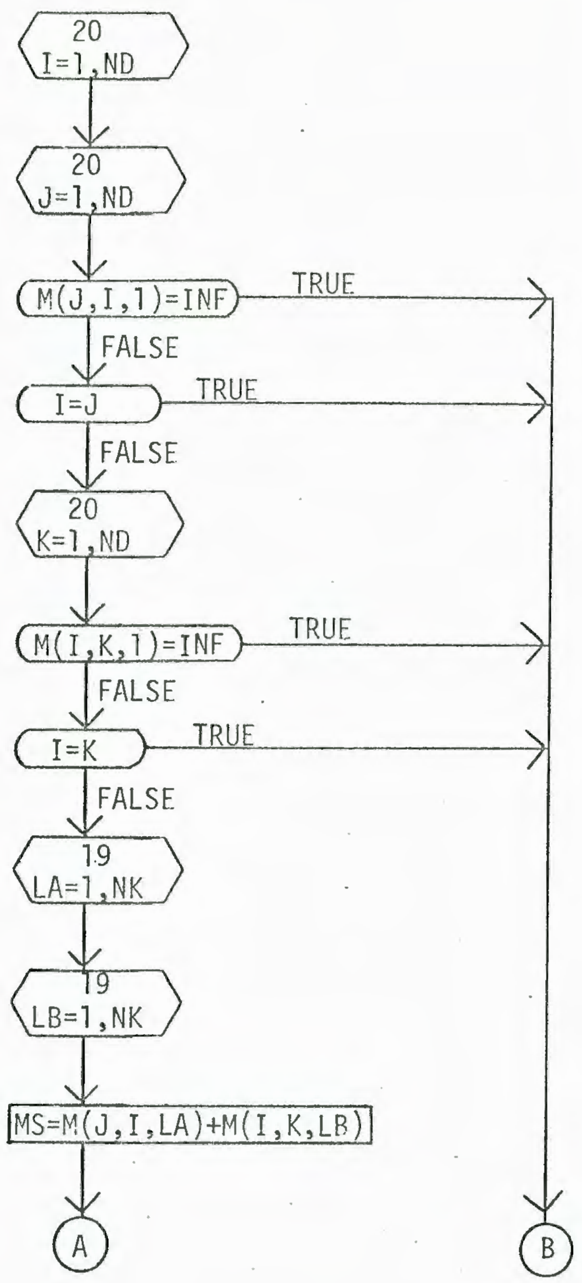
FORTRAN IV COMPUTER PROGRAM FOR THE ALL ALTERNATE PATH ALGORITHM

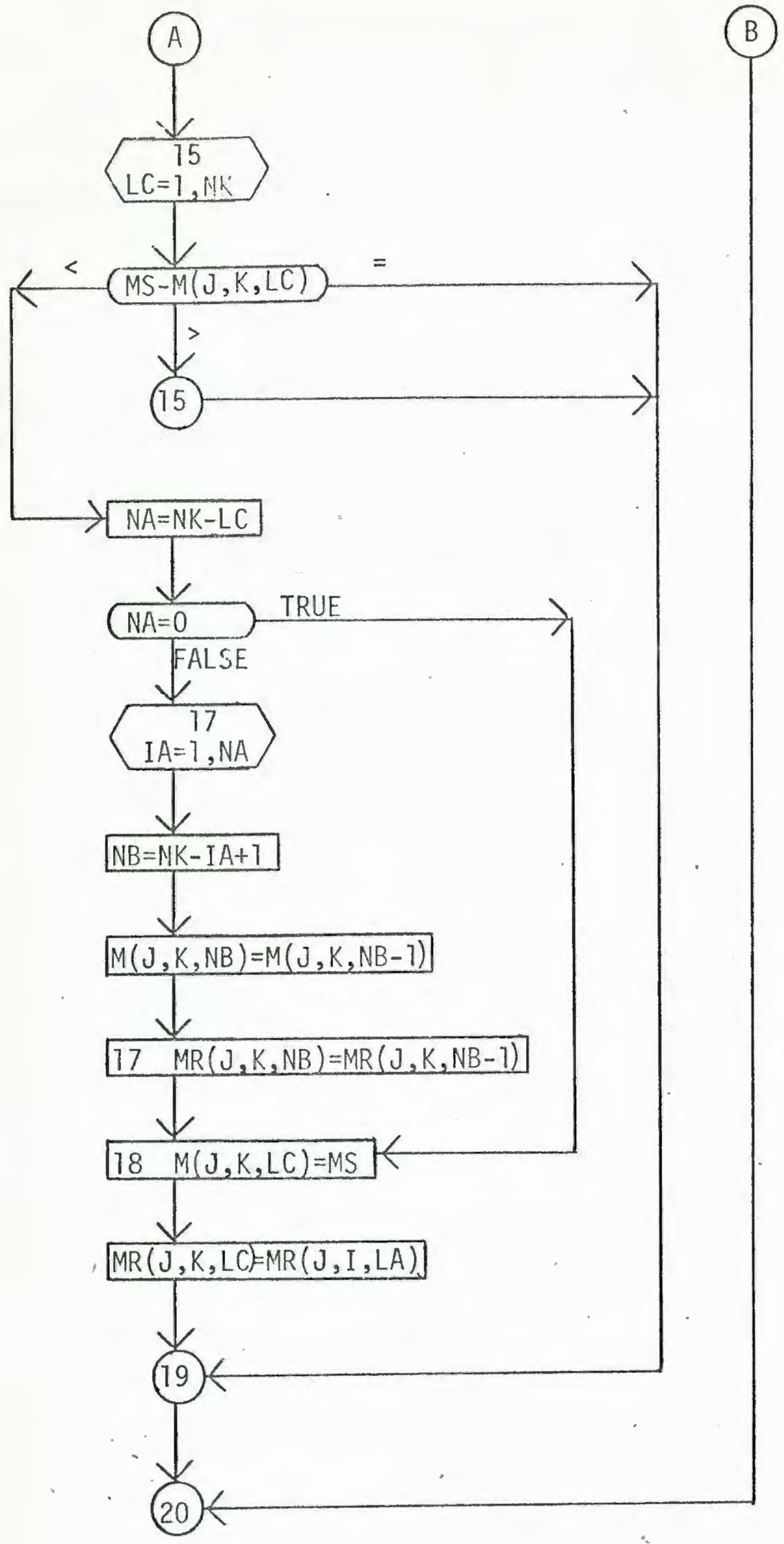
```
C     INF=INFINITY
C     ND=NUMBER OF NODES
C     NA=MAXIMUM NUMBER OF EXPECTED FIRST NODES
C     NGO=FIRST NODE OF PATH TO BE PRINTED OUT
C     NEND=LAST NODE OF PATH TO BE PRINTED OUT
C     NALT=1 IF DESIRE ALL ALTERNATE PATHS FROM NODE NGO TO
C       NODE NEND,0 IF NOT
C     NMAT=1 IF DESIRE ALL ALTERNATE ROUTING MATRICES TO BE
C       PRINTED OUT,0 IF NOT
C     M(I,J)=MATRIX OF DISTANCES FROM I TO J. AS READ IN,IS
C       DISTANCE OF SINGLE ARC FROM I TO J,OR INFINITY IF NO
C       SUCH ARC EXISTS. AS PRINTED OUT,IS SHORTEST DISTANCE
C       OF PATH FROM I TO J
C     MR(I,J,K)=ROUTING MATRIX
C     N1,N2=PARAMETERS USED TO FACILATE PRINT OUT
          COMMON INF,ND,NA,NGO,NEND,M,MR
          DIMENSION M(50,50),MR(50,50,10)
          NAMELIST/INPUT/M
C     READ AND PRINT PARAMETERS
          READ 38,INF,ND,NA,NGO,NEND,NALT,NMAT,N1,N2
          PRINT 38,INF,ND,NA,NGO,NEND,NALT,NMAT,N1,N2
C     ALL ELEMENTS OF M(I,J) ARE SET TO INFINITY EXCEPT
C       DIAGONAL ELEMENTS WHICH ARE SET TO ZERO. MR(I,J,K) IS
C       INITIALIZED
          DO 12 I=1,ND
          DO 11 J=1,ND
          MR(J,I,1)=I
          DO 10 K=2,NA
       10 MR(J,I,K)=INF
       11 M(I,J)=INF
       12 M(I,I)=0
C     READ IN NON-INFINITY ELEMENTS OF M(I,J)
          READ(5,INPUT)
C     PRINT OUT INITIAL DISTANCE MATRIX
          PRINT 33
          PRINT 39,(I,I=1,ND)
          DO 13 I=1,ND
          PRINT 40,I,(M(I,J),J=1,ND),I
       13 CONTINUE
C     CONVERT INITIAL MATRIX TO SHORTEST DISTANCE MATRIX AND
C       TRANSFORM ROUTING MATRIX
          DO 20 I=1,ND
          DO 20 J=1,ND
          IF(M(J,I).EQ.INF) GO TO 20
          IF(I.EQ.J) GO TO 20
          DO 20 K=1,ND
          IF(M(I,K).EQ.INF) GO TO 20
          IF(I.EQ.K) GO TO 20
          MS=M(J,I)+M(I,K)
          IF(MS-M(J,K)) 14,16,20
       14 M(J,K)=MS
          DO 15 L=1,NA
       15 MR(J,K,L)=MR(J,I,L)
          GO TO 20
       16 DO 19 L=1,NA
          IF(MR(J,I,L).EQ.INF) GO TO 20
          DO 17 N=1,NA
          IF(MR(J,K,N).EQ.INF) GO TO 18
          IF(MR(J,I,L)-MR(J,K,N)) 17,19,17
       17 CONTINUE
```

```
      PRINT 37
      GO TO 32
   18 MR(J,K,N)=MR(J,I,L)
   19 CONTINUE
   20 CONTINUE
C  IF NMAT=1 PRINT OUT MATRICES
      IF(NMAT.EQ.0) GO TO 31
C  DETERMINE MAXIMUM NUMBER OF FIRST NODES
      IA=1
      DO 22 I=1,ND
      DO 22 J=1,ND
      KA=0
      DO 21 K=1,NA
      IF(MR(I,J,K).LT.INF) KA=KA+1
   21 CONTINUE
      IF(KA.GT.IA) IA=KA
   22 CONTINUE
C  FOR ELEMENTS OF M(I,J)=INFINITY MAKE CORRESPONDING
C    ELEMENTS OF MR(I,J,1)=INFINITY
      DO 23 I=1,ND
      DO 23 J=1,ND
      IF(M(I,J).LT.INF) GO TO 23
      MR(I,J,1)=INF
   23 CONTINUE
C  PRINT OUT SHORTEST DISTANCE MATRIX
      PRINT 34
      PRINT 41,(I,I=1,N1)
      DO 24 I=1,ND
      PRINT 42,I,(M(I,J),J=1,N1),I
   24 CONTINUE
      IF(N2.EQ.0) GO TO 26
      PRINT 43,(I,I=N2,ND)
      DO 25 I=1,ND
      PRINT 42,I,(M(I,J),J=N2,ND),I
   25 CONTINUE
C  PRINT OUT ROUTING MATRIX
   26 PRINT 35
      DO 30 I=1,IA
      PRINT 41,(L,L=1,N1)
      DO 27 J=1,ND
      PRINT 42,J,(MR(J,K,I),K=1,N1),J
   27 CONTINUE
      IF(N2.EQ.0) GO TO 29
      PRINT 43,(L,L=N2,ND)
      DO 28 J=1,ND
      PRINT 42,J,(MR(J,K,I),K=N2,ND),J
   28 CONTINUE
   29 IF(IA.EQ.I) GO TO 30
      PRINT 36
   30 CONTINUE
C  IF NALT=1 PRINT OUT ALL ALTERNATE PATHS FROM NODE NGO TO
C    NODE NEND
   31 IF(NALT.EQ.0) GO TO 32
      CALL ALTERN
   32 CONTINUE
   33 FORMAT(1H1,' INITIAL MATRIX',//)
   34 FORMAT(1H1 ,' SHORTEST DISTANCE MATRIX',//)
   35 FORMAT(1H1 ,' ROUTING MATRIX',//)
   36 FORMAT(1H1 ,' ALTERNATE ROUTING MATRIX',//)
   37 FORMAT(1H1,' NUMBER OF FIRST NODES EXCEEDS STORAGE')
   38 FORMAT(9I8)
   39 FORMAT(6X,50I2,//)
   40 FORMAT(I3,3X,50I2,I6)
   41 FORMAT(6X,25I4,//)
   42 FORMAT(I3,3X,25I4,I5)
   43 FORMAT(1H1,5X,25I4,//)
      END
      SUBROUTINE ALTERN
      COMMON INF,ND,NA,NGO,NEND,M,MR
      DIMENSION M(50,50),MR(50,50,10)
      MIN=M(NGO,NEND)
      PRINT 17,NGO,NEND,MIN
      PRINT 18
      DO 2 I=1,ND
      M(2,I)=0
      DO 1 J=1,NA
      IF(MR(I,NEND,J).GE.INF) GO TO 2
    1 M(2,I)=M(2,I)+1
    2 CONTINUE
      N=0
      J=1
      K=NGO
      DO 3 I=1,ND
      M(1,I)=INF
    3 M(3,I)=INF
      M(1,J)=K
      J=J+1
    4 IF(K-NEND) 5,10,5
    5 IF(M(2,K).LE.1) GO TO 7
      NN=M(2,K)-1
      DO 6 I=1,NN
      N=N+1
    6 M(3,N)=K
      L=K
    7 MAI=M(2,K)
      IF(MAI.EQ.1) GO TO 9
      MAI=1
      DO 8 I=1,ND
      IF(M(3,I).EQ.K) MAI=MAI+1
    8 CONTINUE
    9 NEXT=MR(K,NEND,MAI)
      M(1,J)=NEXT
      K=NEXT
      J=J+1
      GO TO 4
   10 II=J-1
      PRINT 19,(M(1,I),I=1,II)
      IF(N.LE.0) GO TO 16
      DO 11 I=1,ND
      IF(M(1,J-1).EQ.L) GO TO 12
      J=J-1
   11 M(1,J)=INF
   12 CONTINUE
      M(3,N)=INF
      N=N-1
      MAI=M(2,L)
      IF(MAI.EQ.1) GO TO 14
      MAI=1
      DO 13 I=1,ND
      IF(M(3,I).EQ.L) MAI=MAI+1
   13 CONTINUE
   14 NEXT=MR(L,NEND,MAI)
      M(1,J)=NEXT
      J=J+1
      K=NEXT
      IF(N-0) 4,4,15
   15 L=M(3,N)
      GO TO 4
   16 CONTINUE
   17 FORMAT(1H1, 'MINIMUM DISTANCE FROM NODE',I3,' TO NODE',
     1  I3,' IS',I3,' UNITS')
   18 FORMAT(////,' MINIMUM PATHS')
   19 FORMAT(/,30I3)
      RETURN
      END
```

APPENDIX IV

Flow Diagram for the Nth Best Path Algorithm




A
B
15
LC=1,NK
<
=
MS-M(J,K,LC)
>
15
NA=NK-LC
NA=0
TRUE
FALSE
17
IA=1,NA
NB=NK-IA+1
M(J,K,NB)=M(J,K,NB-1)
17 MR(J,K,NB)=MR(J,K,NB-1)
18 M(J,K,LC)=MS
MR(J,K,LC)=MR(J,I,LA)
19
20

APPENDIX V

FORTRAN IV COMPUTER PROGRAM FOR THE NTH BEST
PATH ALGORITHM

```
C   INF=INFINITY
C   ND=NUMBER OF NODES
C   NK=NUMBER OF K BEST PATHS
C   NGO=FIRST NODE OF PATH TO BE PRINTED OUT
C   NEND=LAST NODE OF PATH TO BE PRINTED OUT
C   NKTH=1 IF DESIRE ALL K BEST PATHS FROM NODE NGO TO NODE
C     NEND,0 IF NOT
C   NMAT=1 IF DESIRE ALL K BEST ROUTING MATRICES TO BE
C     PRINTED OUT,0 IF NOT
C   M(I,J,K)=MATRIX OF DISTANCES FROM I TO J. AS READ IN,IS
C     DISTANCE OF SINGLE ARC FROM I TO J,OR INFINITY IF NO
C     SUCH ARC EXISTS. AS PRINTED OUT,IS KTH BEST DISTANCE OF
C     PATH FROM I TO J
C   MR(I,J,K)=ROUTING MATRIX
C   N1,N2=PARAMETEFS USED TO FACILATE PRINT OUT
      DIMENSION M(50,50,15),MR(50,50,15),MI(50,50)
      COMMON INF,ND,NK,NGO,NEND,M,MR,MI
      NAMELIST/INPUT/M
C   READ AND PRINT PARAMETERS
      READ 30,INF,ND,NK,NGO,NEND,NKTH,NMAT,N1,N2
      PRINT 30,INF,ND,NK,NGO,NEND,NKTH,NMAT,N1,N2
C   ALL ELEMENTS OF M(I,J,K) ARE SET TO INFINITY EXCEPT
C     DIAGONAL,ELEMENTS FOR K=1,WHICH ARE SET TO ZERO.
C     MR(I,J,K) IS INITIALIZED
      DO 11 I=1,ND
      DO 10 J=1,ND
      MR(J,I,1)=I
      M(I,J,1)=INF
      DO 10 K=2,NK
      M(I,J,K)=INF
   10 MR(J,I,K)=INF
   11 M(I,I,1)=0
C   READ IN NON-INFINITY ELEMENTS OF M(I,J,K)
      READ(5,INPUT)
C   STORE M(I,J,1) IN MI(I,J) FOR USE IN SUBROUTINE KBEST
      DO 12 I=1,ND
      DO 12 J=1,ND
   12 MI(I,J)=M(I,J,1)
C   PRINT OUT INITIAL DISTANCE MATRIX
      PRINT 27
      PRINT 31,(I,I=1,ND)
      DO 13 I=1,ND
      PRINT 32,I,(M(I,J,1),J=1,ND),I
   13 CONTINUE
C   FOR ELEMENTS OF M(I,J)=INFINITY MAKE CORRESPONDING
C     ELEMENTS OF MR(I,J,1)=INFINITY
      DO 14 I=1,ND
      DO 14 J=1,ND
      IF(M(I,J,1).LT.INF) GO TO 14
      MR(I,J,1)=INF
   14 CONTINUE
C   CONVERT INITIAL MATRIX TO KTH BEST DISTANCE MATRIX AND
C     TRANSFORM ROUTING MATRIX
      DO 20 I=1,ND
      DO 20 J=1,ND
      IF(M(J,I,1).EQ.INF) GO TO 20
      IF(I.EQ.J) GO TO 20
      DO 20 K=1,ND
      IF(M(I,K,1).EQ.INF) GO TO 20
      IF(I.EQ.K) GO TO 20
      DO 19 LA=1,NK
      DO 19 LB=1,NK
      MS=M(J,I,LA)+M(I,K,LB)
      DO 15 LC=1,NK
      IF(MS-M(J,K,LC)) 16,19,15
   15 CONTINUE
      GO TO 19
   16 NA=NK-LC
      IF(NA.EQ.0) GO TO 18
      DO 17 IA=1,NA
      NB=NK-IA+1
      M(J,K,NB)=M(J,K,NB-1)
   17 MR(J,K,NB)=MR(J,K,NB-1)
   18 M(J,K,LC)=MS
      MR(J,K,LC)=MR(J,I,LA)
   19 CONTINUE
   20 CONTINUE
C     IF NMAT=1 PRINT OUT MATRICES
      IF(NMAT.EQ.0) GO TO 25
C     PRINT OUT K BEST DISTANCE MATRICES
      DO 22 J=1,NK
      PRINT 28,J
      PRINT 33,(I,I=1,N1)
      DO 21 I=1,ND
      PRINT 34,I,(M(I,K,J),K=1,N1),I
   21 CONTINUE
      IF(N2.EQ.0) GO TO 22
      PRINT 35,(I,I=N2,ND)
      DO 22 I=1,ND
      PRINT 34,I,(M(I,K,J),K=N2,ND),I
   22 CONTINUE
C     PRINT OUT K BEST ROUTING MATRICES
      DO 24 J=1,NK
      PRINT 29,J
      PRINT 33,(I,I=1,N1)
      DO 23 I=1,ND
      PRINT 34,I,(MR(I,K,J),K=1,N1),I
   23 CONTINUE
      IF(N2.EQ.0) GO TO 24
      PRINT 35,(I,I=N2,ND)
      DO 24 I=1,ND
      PRINT 34,I,(MR(I,K,J),K=N2,ND),I
   24 CONTINUE
   25 IF(NKTH.EQ.0) GO TO 26
C     IF NKTH=1 PRINT OUT ALL K BEST PATHS FROM NODE NGO TO
C       NODE NEND
      CALL KBEST
   26 CONTINUE
   27 FORMAT(1H1,' INITIAL MATRIX',//)
   28 FORMAT(1H1 ,I4,'TH BEST DISTANCE MATRIX',//)
   29 FORMAT(1H1 ,I4,'TH BEST ROUTING MATRIX',//)
   30 FORMAT(9I8)
   31 FORMAT(6X,50I2,//)
   32 FORMAT(I3,3X,50I2,I6)
   33 FORMAT(6X,25I4,//)
   34 FORMAT(I3,3X,25I4,I5)
   35 FORMAT(1H1,5X,25I4,//)
      END
      SUBROUTINE KBEST
      COMMON INF,ND,NK,NGO,NEND,M,MR,MI
      DIMENSION M(50,50,15),MR(50,50,15),MI(50,50),MK(30)
      PRINT 7
      DO 6 I=1,NK
      ITH=I
      MDIS=M(NGO,NEND,I)
      MDIST=MDIS
      DO 1 L=1,30
    1 MK(L)=INF
      KA=NGO
      N=1
      MK(N)=KA
    2 KB=MR(KA,NEND,ITH)
      N=N+1
      MK(N)=KB
      IF(KB.EQ.NEND) GO TO 5
      MDIS=MDIS-MI(KA,KB)
      DO 3 J=1,NK
      IF(M(KB,NEND,J).EQ.MDIS) GO TO 4
    3 CONTINUE
    4 ITH=J
      KA=KB
      GO TO 2
    5 PRINT 8,I,NGO,NEND,MDIST
      PRINT 9,(MK(L),L=1,N)
    6 CONTINUE
    7 FORMAT(1H1)
    8 FORMAT(///,I4,'TH BEST DISTANCE FROM NODE',I3,' TO NODE',
     1  I3,' IS',I5,' UNITS')
    9 FORMAT(//,30I3)
      RETURN
      END
```

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	20
2.	Library Naval Postgraduate School Monterey, California 93940	2
3.	Director, Systems Analysis Division (OP 96) Office of the Chief of Naval Operations Washington, D. C. 20350	1
4.	Department of the Army Civil Schools Branch, OPO, OPD Washington, D. C. 20315	1
5.	Professor Harold Greenberg Department of Operations Analysis Naval Postgraduate School Monterey, California 93940	1
6.	Captain Robert J. Bohls 346 Ardennes Circle Fort Ord, California 93941	1
7.	Department of Operations Analysis Naval Postgraduate School Monterey, California 93940	1

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Naval Postgraduate School Monterey, California 93940	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

On Network Analysis

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Thesis

5. AUTHOR(S) (Last name, first name, initial)

BOHLS, Robert J., Captain, United States Army

6. REPORT DATE June 1968	7a. TOTAL NO. OF PAGES 46	7b. NO. OF REFS 19

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

This document has been approved for public release and sale; its distribution is unlimited. [illegible] 1/26/70

10. AVAILABILITY/LIMITATION NOTICES

[redacted]

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Naval Postgraduate School Monterey, California

13. ABSTRACT

This thesis reviews four of the existing matrix methods for finding the shortest path in a network, including the little known matrix method by Floyd. Floyd's method is then extended to determine all best paths. After a brief review of the nth best path problem, Floyd's method is again extended to determine the nth best path. Finally, the nth best path problem is investigated to determine its applicability to the traveling salesman problem.

DD FORM 1 JAN 64 1473

UNCLASSIFIED
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Shortest Path Problem						
All Alternate Best Paths Problem						
Nth Best Path Problem						
Traveling Salesman Problem						