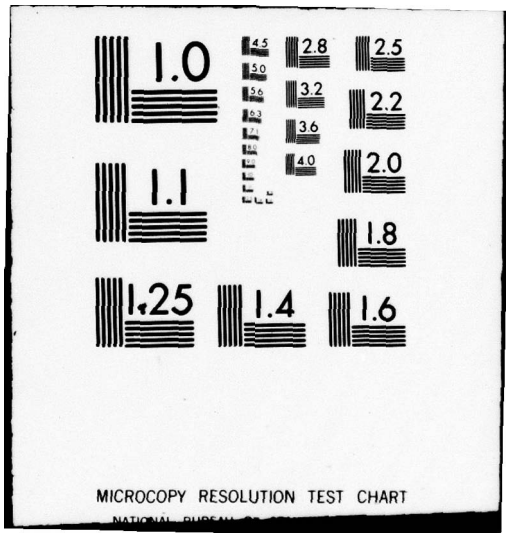
AD-A075 415 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 12/1
DISTRIBUTED SHORTEST PATH ALGORITHMS FOR COMPUTER NETWORKS.(U)
AUG 79 J Y YEN
UNCLASSIFIED NPS55-79-017 NL

| OF |
ADA
075415

END
DATE
FILMED
11 -79
DDC

END
DATE
FILMED
11 -79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
8.0
9.0
MICROCOPY RESOLUTION TEST CHART

NPS55-79-017

NAVAL POSTGRADUATE SCHOOL

Monterey, California

DISTRIBUTED SHORTEST PATH ALGORITHMS FOR COMPUTER NETWORKS

by

Jin Y. Yen

August 1979

Prepared for:
Chief of Naval Research
Arlington, VA 22217

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

Rear Admiral T. F. Dedman
Superintendent

Jack R. Borsting
Provost

The work reported herein was supported by the Foundation Research Program of the Naval Postgraduate School with funds provided by the Chief of Naval Research.

This report was prepared by:

Jin Y. Yen

Jin Y. Yen, Adjunct Professor
Department of Operations Research

Reviewed by:

G. Howard, Acting Chm.

Michael G. Sovereign, Chairman
Department of Operations Research

Released by:

William M. Tolles

William M. Tolles
Dean of Research

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

(14) 1. REPORT NUMBER: NPS55-79-Ø17 ✓ | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER

(6) 4. TITLE (and Subtitle): DISTRIBUTED SHORTEST PATH ALGORITHMS FOR COMPUTED NETWORKS.

(9) 5. TYPE OF REPORT & PERIOD COVERED: Technical rept.

6. PERFORMING ORG. REPORT NUMBER

(10) 7. AUTHOR(s): Jin Y. Yen

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Postgraduate School
Monterey, CA 93940

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
61152N, RR 000-01-01
N0001479WR90027

11. CONTROLLING OFFICE NAME AND ADDRESS
Naval Postgraduate School
Monterey, CA 93940

(11) 12. REPORT DATE: August 1979

13. NUMBER OF PAGES: 36

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
(12) 38

15. SECURITY CLASS. (of this report): Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

(16) RRØØØØ1

(17) RRØØØØ1Ø1

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper presents two distributed algorithms for finding shortest paths from a source node to all other nodes in an N-node network. These algorithms are executed at individual nodes using only local information. Algorithm 1 works in networks where there are no topological changes such as link failures, link recoveries or changes of link lengths. Algorithm 2 is a modification of Algorithm 1 for networks where there are topological changes. Algorithm 1 determines the optimal shortest paths in at most $N^3/4$ steps,

N3/4

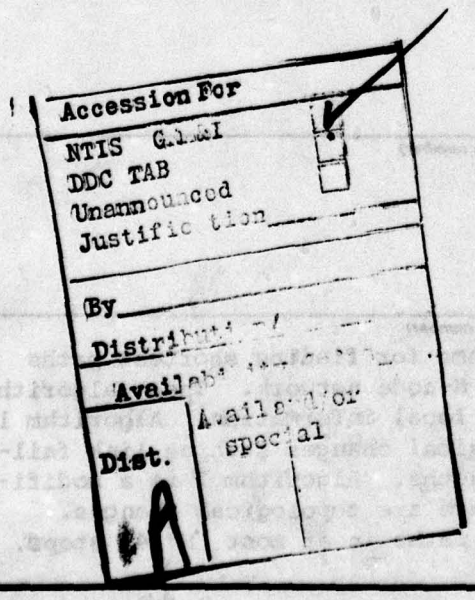
over

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

9 51 450

20. Abstract Continued.
which is only one-half of the computational upper bounds of Abram and Rhodes' and Segall, Merlin and Gallager's algorithms. After the last topological change, Algorithm 2 determines the optimal shortest paths in the same number of steps as Algorithm 1. There are many situations where the present algorithms will work up to N/2 times faster than the algorithms proposed by these authors.



DISTRIBUTED SHORTEST PATH ALGORITHMS FOR COMPUTER NETWORKS

by

Jin Y. Yen

Naval Postgraduate School
Monterey, California

August 1979

DISTRIBUTED SHORTEST PATH ALGORITHMS FOR COMPUTER NETWORKS

by

Jin Y. Yen
Naval Postgraduate School
Monterey, CA 93940

ABSTRACT

This paper presents two distributed algorithms for finding shortest paths from a source node to all other nodes in an N-node network. These algorithms are executed at individual nodes using only local information. Algorithm 1 works in networks where there are no topological changes such as link failures, link recoveries or changes of link lengths. Algorithm 2 is a modification of Algorithm 1 for networks where there are topological changes. Algorithm 1 determines the optimal shortest paths in at most $N^3/4$ steps, which is only one-half of the computational upper bounds of Abram and Rhodes' and Segall, Merlin and Gallager's algorithms. After the last topological change, Algorithm 2 determines the optimal shortest paths in the same number of steps as Algorithm 1. There are many situations where the present algorithms will work up to N/2 times faster than the algorithms proposed by these authors.

1. INTRODUCTION

Finding shortest paths in a network is an important problem and has many applications in computer networks [2], [5], [10]. Two types of algorithms are available for finding the shortest paths in networks. The first type of algorithm is called a centralized shortest path algorithm. In order to apply centralized shortest path algorithms, a network must establish a central node to gather information concerning the complete network topology so that the algorithm can be executed at the central node. Many authors have introduced centralized shortest path algorithms which are very efficient and are widely known [4], [5], [9], [10], [14]. The second type of algorithm is called a distributed (or decentralized) shortest path algorithm. Distributed shortest path algorithms do not require a central node to execute the algorithm; instead, they are executed locally at each individual node using only the limited information received from its neighboring nodes. As compared with centralized shortest path algorithms, distributed shortest path algorithms are less efficient and less well-known. Authors who have introduced distributed shortest path algorithms include: Abram and Rhodes [1], Boehm and Mobley [3], Fultz and Kleinrock [6], McQuillan [7], Naylor [8], Segall, Merlin and Gallager [11] and Tajibnapis [12].

In a computer network, the links or nodes do not always function properly. They often change from a state of functioning to a state of breakdown. Also, the length of a link may increase or decrease from time to time. Therefore it is necessary for a shortest path algorithm to adapt to such topological changes in the network. In our discussion, we will use the length of links to indicate whether links or nodes fail or recover. When a link fails, we will set the length of the link to infinity; and when a link recovers, we will change the link length from infinity to the actual length of the link. Similarly, when a node fails, we will set the length of all links associated with the node to infinity; and, when a node recovers, we will change the associated link lengths from infinity to the actual length of the links. Also, the term, "adaptive algorithm," is used to describe an algorithm that can be applied to a network where the link lengths are subject to periodical changes.

The purpose of this paper is to present an adaptive distributed algorithm for finding shortest path from a source node to all other nodes in an N-node computer network. We will present this algorithm, first, by describing a basic distributed algorithm for finding shortest paths from an origin to all other nodes in a network where there are no topological changes (Algorithm 1). Then we will present the modification of Algorithm 1 that is applicable to topological changes in networks

(Algorithm 2). Also, we will briefly describe how these algorithms can be applied to find all shortest paths between every pair of nodes in the network.

The advantages of Algorithm 2 are as follows:

1) The algorithm is applicable to networks with directed or undirected links.
2) It is executed locally at each individual node using only simple information obtained from the neighborhing nodes.
3) It is adaptive to topological changes such as changes in link lengths and the failure or recovery of links and nodes.
4) The shortest paths obtained at the termination of the algorithm are optimal.
5) The algorithm works much faster than other available algorithms. After the last topological change is made, the algorithm requires in the worst case a total of $N^3/4$ additions and $N^3/4$ comparisons to obtain all optimal shortest paths from a source node to all other nodes. However, experience has shown that in practice the algorithm terminates much faster than these worst bounds.
6) The algorithm is easy to understand and easy to program.

2. ALGORITHM 1

In this section we will present a distributed algorithm for finding the shortest paths from an origin to all other nodes in a network where there are no topological changes. In order to apply the algorithm we require that:

1) Each node is given a node label: node 1, node 2,..., node N, where node 1 is the origin.

2) Each node has knowledge of a set of neighboring TO nodes that are connected to node I by directed links leading from node I to the TO nodes, respectively. And the set of TO nodes are further separated into two subsets: HIGHER TO nodes and LOWER TO nodes. The HIGHER TO nodes are the TO nodes whose node labels are larger than I, and the LOWER TO nodes are the TO nodes whose node labels are smaller than I. For example, node 5 is a HIGHER TO node of node 2, and is a LOWER TO node of node 9. Each node I also has knowledge of the lengths of the directed links going from node I to the TO nodes.

In an N-node directed network, let

$I, J, L = 1,2,\ldots,N$, be the nodes of the network where node 1 is the origin, and the remaining nodes are numbered arbitrarily,

(I,J) be the directed link connecting node I to node J,

$D(I,J) \geq 0$ be the length of link (I,J),

F(J:K), J = 2,3,...,N and K = 1,2,... be the distance of the tentative shortest path from node 1 to node J at the current cycle K of the algorithm,

H(J:K) be the node that precedes node J on a tentative shortest path from node 1 to node J whose distance is F(J:K).

More notation will be introduced later when it becomes necessary.

Algorithm 1 is developed from the following principle. At a pre-arranged time, each node I, I = 2,3,...,N, in turn informs each of its TO nodes J what is the current best shortest distance from node 1 to node J via node I. This procedure allows the shortest distances from node 1 to all other nodes to be updated iteratively in order to reach optimality. The following steps can be taken to achieve this principle.

1) Divide the time horizon into N cycles and divide each cycle into N time periods.

2) In an odd cycle K at time period T: node I (where I = T) sends each of its HIGHER TO nodes J the current best shortest distance from node 1 to node J, i.e., F(J:K).

3) In an even cycle K at time period T: node I (where I = N-T+1) sends each of its LOWER TO nodes J the current best shortest distance from node 1 to node J, i.e., F(J:K).

4) In any cycle K, any node I is not to send its TO nodes the current shortest distances F(J:K)'s unless F(J:K) < F(J:K-2).

A proof of why these steps will determine the optimal shortest distances from node 1 to all other nodes in at most N-1 cycles can be found in [13], [14, pp. 52-71].

We will now describe the distributed algorithm for finding the shortest paths from node 1 to all other nodes in a network where there are no topological changes. Note that the following Algorithm 1 is to be executed by cycles in the order $K = 0,1,2,\ldots,N-1$.

Algorithm 1

Cycle 0 (to initialize F(J:-1)'s and F(J:0)'s)

A. Let $F(J:-1) = \infty$ for all $J = 1,2,\ldots,N$.

B. Let $F(1:0) = 0$, $F(J:0) = D(1,J)$ and $H(J:0) = 1$ for all $J = 2,3,\ldots,N$.

Cycle $K = 1,3,\ldots$ (to update F(J:K)'s for pivot node I's HIGHER TO nodes J)

A. Let $F(J:K) = F(J:K-1)$ for $J = 1,2,\ldots,N$.

B. At each of time T, $T = 1,2,\ldots,N-1$, do the following:

I. (at pivot node I, where $I = T$)

Check if $F(I:K) < F(I:K-2)$.

a. If $F(I:K) \geq F(I:K-2)$, do nothing and go to time T+1.

b. If $F(I:K) < F(I:K-2)$, compute $F(J:K)^*$ by $F(J:K)^* = F(I:K) + D(I,J)$ for each of node I's HIGHER TO nodes J; and, transmit "$F(J:K)^*$" and "I" to the corresponding HIGHER TO nodes J. Go to time T+1.

II. (at each of HIGHER TO nodes J of node I)

After receiving "F(J:K)*" check if $F(J:K)^* < F(J:K)$.

a. If $F(J:K)^* \geq F(J:K)$, do nothing.

b. If $F(J:K)^* < F(J:K)$, F(J:K*) becomes F(J:K) and I becomes H(J:K); or, $F(J:K)^* \leftarrow F(J:K)$ and $H(J:K) \leftarrow I$.

III. (at all other nodes)

Do nothing.

Cycle K = 2,4,... (to update F(J:K)'s for pivot node I's LOWER TO nodes J)

A. Let $F(J:K) = F(J:K-1)$ for $J = 1,2,\ldots,N$.

B. At each of time T, T = 1,2,...,N-2, do the following:

I. (at pivot node I, where $I = N-T+1$)

Check if $F(I:K) < F(I:K-2)$.

a. If $F(I:K) \geq F(I:K-2)$, do nothing and go to next time T+1.

b. If $F(I:K) < F(I:K-2)$ compute $F(J:K)^*$ by $F(J:K)^* = F(I:K) + D(I,J)$ for each of node I's LOWER TO nodes J, and transmit "F(J:K)*" and "I" to the corresponding LOWER TO node J. Go to time T+1.

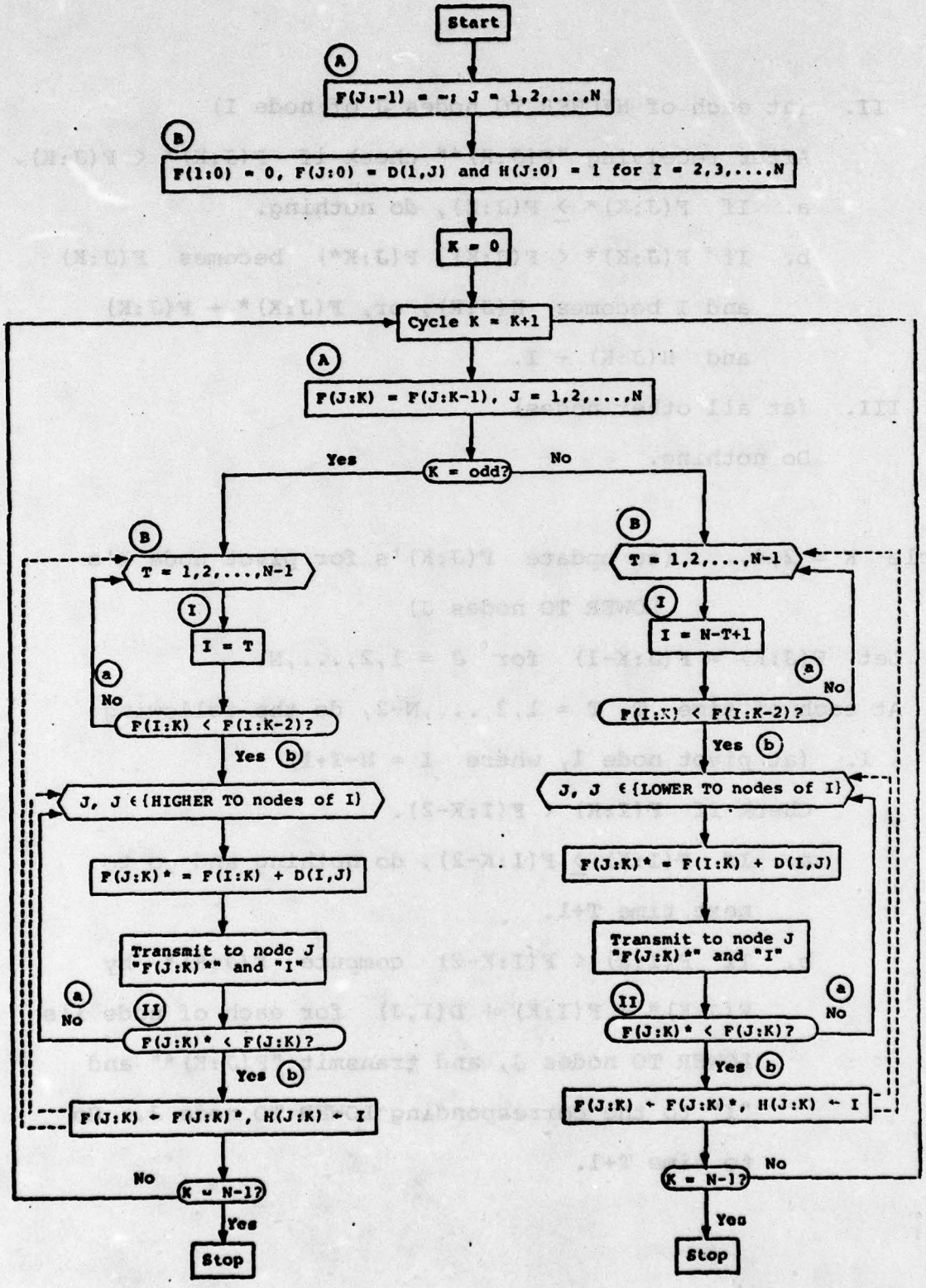


FIGURE 1. A flow chart of Algorithm 1 for finding the shortest paths from node 1 to all other nodes. The circled letters are the step numbers of the algorithms.

II. (at the LOWER TO nodes J of node I)

After receiving "F(J:K)*", check if F(J:K)* < F(J:K).

a. If F(J:K)* $\geq$ F(J:K), do nothing.

b. If F(J:K)* < F(J:K), let F(J:K) $\leftarrow$ F(J:K)*

and H(J:K) $\leftarrow$ I.

III. (at all other nodes)

Do nothing.

A flow chart of Algorithm 1 is given in Figure 1.

The optimal distances of the shortest paths from node 1 to all other nodes are determined when F(J:K) = F(J:K-1) for all J, J = 2,3,...,N. When this condition occurs, all nodes J will cease transmitting; and, this condition will occur no later than in cycle N-1. We will use an example to illustrate how Algorithm 1 determines the shortest distances from node 1 to all other nodes in no more than N-1 cycles. Suppose that in a 10-node network the shortest path from node 1 to node 9 is 1-3-8-7-5-2-4-6-9. Then the shortest distance from node 1 to node 3 is determined in cycle K = 1, time T = 1 as F(3:1); the shortest distance to node 8 is determined in K = 1, T = 3 as F(8:1); to node 7 in K = 2, T = 3 as F(7:2); to node 5 in K = 2, T = 4 as F(5:2); to node 2 in K = 2, T = 6 as F(2:2); to node 4 in K = 3, T = 2 as F(4:3); to node 6 in K = 3, T = 4 as F(6:3); and, to node 9 in K = 3, T = 6 as F(9:3). As a matter of fact, the optimal distances of all shortest paths from node 1 to other nodes

that include a sequence of successively-larger nodes, such as 1-3 and 1-3-8 in our example, are all determined in cycle K = 1. In cycle K = 2, more shortest paths are determined by attaching a sequence of successively-smaller nodes, such as 7-5 and 7-5-2, to the nodes of the shortest paths already determined in cycle K = 1. In cycle K = 3, more shortest paths are determined by attaching a sequence of successively-larger nodes, such as 4-6 and 4-6-9 in our example, to the nodes of the shortest paths that are determined in cycle K = 2,... . Since a shortest path can include at most N-1 successively-increasing and successively-decreasing sequences of nodes, all shortest paths in the network are determined in no more than N-1 cycles. Actually, the number of cycles it takes the algorithm to determine all shortest paths is equal to the maximum number of successively-increasing and successively-decreasing sequence of nodes in any of the shortest paths from node 1 to all other nodes.

When the algorithm is terminated, say, at the end of cycle N-1, each node J, J = 2,3,...,N, has the following solutions:

1) The optimal distance of the shortest path from node 1 to node J; i.e., F(J:N-1), and
2) The identity of the node that proceeds node J on the shortest path from node 1 to node J; i.e., H(J:N-1).

However, node 1 does not know what are the shortest distances to all other nodes and what are the next nodes on the shortest paths to these nodes. If desired, node 1 can obtain this information by requiring each node J to transmit F(J:N-1) back to node 1. If link (J,I) exists for every link (I,J), node J can easily send F(J:N-1) back to node 1 via the intermediate nodes on the shortest path. However, if link (J,I) does not exist for every link (I,J), node J has to find a new path to send the F(J:N-1) back to node 1, which requires substantial effort.

To examine the efficiency of Algorithm 1, recall that there are N nodes in the network. In the worst case, each of cycles K = 1,3,... requires node 1 to make N-1 additions, comparisons and transmissions; node 2 to make N-2 additions, comparisons and transmissions; ..., and so on. The sum taken over nodes is approximately $N^2/2$ additions, comparisons and transmissions. Similarly, each of cycles K = 2,4,... requires a total of approximately $N^2/2$ additions, comparisons and transmissions. In the worst case, the algorithm has to be executed for N-1 cycles, therefore, approximately $N^3/2$ additions, comparisons and transmissions are necessary overall. However, in each cycle of the algorithm, at least one optimal shortest distance from node 1 to node J will be determined; and the determination of a shortest distance will save N-1 additions, comparisons and transmissions in two cycles to follow.

Consequently, a total of $[(N-1)^2/2][1 + 2 + 3 + \cdots + N-1] \doteq N^3/4$ additions, comparisons and transmissions are saved throughout the $N-1$ cycles of the algorithm. Therefore, in the worst case, Algorithm 1 requires approximately $N^3/2 - N^3/4 = N^3/4$ additions, comparisons and transmissions. In the best case, Algorithm 1 can determine all shortest distances in one cycle using only $N^2/2$ additions, comparions and transmissions. Some empirical studies have found that algorithms similar to Algorithm 1 have required much fewer operations than the worst computational bound of $N^3/4$ [14, pp. 82-89].

The relative efficiency of Algorithm 1 appears to be quite good as compared with other available algorithms. For example, the worst computational bound of Algorithm 1 is only one-half of the worst bounds of Abram and Rhodes' [1] and Segall, Merlin and Gallager's [11] algorithms. However, in most cases Algorithm 1 will work much faster than these two algorithms. For example, in a 10-node network where the shortest paths are: 1-10-9-8-7-6-5-4-3-2, Algorithm 1 will require two cycles using approximately N^2 steps; whereas, Abram and Rhodes' algorithm will require 9 cycles using approximately $5N^2$ steps. When N is large Algorithm 1 can work up to approximately $N/2$ timesfaster than the algorithm of Abram and Rhodes. While Algorithm 1 is expected to work faster than Abram and Rhodes' algorithm in most situations, there is no situation that Algorithm 1 will work slower than their algorithm.

Algorithm 1 represents a basic algorithm for finding the shortest paths *from* node 1 to all other nodes. Algorithm 1 can be modified to improve its use and efficiency. For example, node 1 can be excluded from execution of the algorithm after cycle 1.

Algorithm 1 can also be applied to find the shortest paths from any node to all other nodes in the network. To find the shortest paths from an arbitrary node I to all other nodes, we can modify the algorithm slightly by initializing $F(I:0) = 0$ and $F(J:0) = \infty$ for all $I \neq J$ in Step B of Cycle 0, and execute cycles 1,2,..., in the same manner as Algorithm 1. Of course the $F(J:K)$'s in such a case represent the shortest distances from node I to node J, respectively. However, in this case the algorithm may not start building the shortest paths until cycle $K = 2$, time $T = N-T+1$, consequently, the algorithm should be executed at least until cycle N, time $T = N-I$, to guarantee an optimal solution. For simplicity, when the origin is not node 1, we will terminate the modified Algorithm 1 after cycle N instead of N-1.

The above modification of Algorithm 1 can be applied concurrently to find the shortest paths between all pairs of nodes in N cycles. That is, in each cycle K, the modified algorithm can be executed N times by successively taking node L, $L = 1,2,...,N$, as the source node. When the modified algorithm is applied to find the shortest paths between all

pairs of nodes the worst computational bound is $N^4/4$ additions and $N^4/4$ comparisons. However, the upper bound of the necessary transmissions between the nodes is only $N^3/4$, because each transmission from node I to node J can contain the shortest distances from all N-1 source nodes to node J.

Algorithm 1 can also be modified slightly so that each pivot node I does not have to transmit messages at a pre-arranged time. This can be done as follows: In an odd cycle, node I becomes a pivot node and is ready to transmit whenever it has received messages from all of its neighboring nodes whose node numbers are smaller than I; and, in an even cycle, node I becomes a pivot node and is ready to transmit whenever it has received messages from all of its neighboring nodes whose node numbers are larger than I. The advantages of this modification are that no clock is necessary at each node and that the algorithm can be executed within a shorter time span. This time savings occurs because instead of waiting for its pre-arranged time to transmit messages, node I can transmit messages as soon as it becomes a pivot node.

3. ALGORITHM 2.

In this section we will present Algorithm 2 for finding the shortest paths from node 1 to all other nodes in a network which is adaptive to topological changes. Algorithm 2 is a modification of Algorithm 1 and uses the following additional ideas:

1) When the link length D(I,J) is reduced, the algorithm updates F(J:K) using the reduced D(I,J).
2) When the link length D(I,J) is increased such that the value of F(J:K) may be increased, the algorithm discards all tentative shortest paths that use link (I,J) on their paths, and then rebuilds new paths.
3) When the last link, (I,J), on the tentative shortest path to node I fails, the algorithm discards all tentative shortest paths that use link(I,J) on their paths, and then rebuilds new paths.

The distributed algorithm for finding the shortest paths from node 1 to all other nodes in a network which is adaptive to topological changes is as follows. Note that in Algorithm 2, unless otherwise specified, D(I,J) represents the current link length at the current cycle and time period, which can be different from earlier lengths due to topological changes.

Algorithm 2

Cycle 0 (to initialize F(J:0)'s)

A. Let F(1:0) = 0, F(J:0) = D(1,J) and H(J:0) = 1 for J = 2,3,...,N.

Cycle K = 1,3,... (to update F(J:K)'s for the pivot node I's HIGHER TO nodes J)

A. Let F(J:K) = F(J:K-1) for all J = 1,2,...,N.

B. At each of time T, T = 1,2,...,N, do the following:

I. (at pivot node I, where I = T)

a. Check if F(I:K) = ∞.

i. If yes, go to Step b.

ii. If no, go to Step c.

b. Check if F(I:K) was set to ∞ due to the message "DISMANTLE" received since the last time node I was the pivot node in cycle K-1, time T = N-I+2.

i. If no, go to time T+1.

ii. If yes, send node I's HIGHER TO and LOWER TO node messages: "DISMANTLE" and "I". Then go to time T+1.

c. Check if link (L,I) still exists, where node L, L = H(I:K), is the node from which node I has obtained the current F(I:K).

i. If yes, go t Step d.

ii. If no, let $F(I:K) \leftarrow \infty$ and send each of node I's HIGHER TO and LOWER TO nodes J the messages: "DISMANTLE" and "I". Then go to time T+1.

d. For each of node I's HIGHER TO nodes J, check if D(I,J) has been increased since cycle K-2, time T = I+1.

i. If no, compute F(J:K)* = F(I:K) + D(I,J) and transmit the messages "F(J:K)*" and "I" to node J.

ii. If yes, send node J the messages: "DISMANTLE" and "I". Go to time T+1 after Step d is done.

II. (at each of node I's TO nodes J)

a. If node J receives the messages "F(J:K)*" and "I", do the following:

i. Check if $F(J:K) \leftarrow \infty$ due to the message "DISMANTLE" received since cycle K-1, time T = N-I+2.
If yes, do nothing.
If no, go to Step ii.

ii. Check if $F(J:K)^* < F(J:K)$.

If no, do nothing.

If yes, let $F(J:K) \leftarrow F(J:K)^*$ and $H(J:K) \leftarrow I$.

b. If node J receives the messages: "DISMANTLE" and "I", check if $I = H(J:K)$.

i. If $I \neq H(J:K)$, do nothing.

ii. If $I = H(J:K)$, let $F(J:K) \leftarrow \infty$, and do not update $F(J:K)$ until after node J becomes a pivot node.

III. (all other nodes)

Do nothing.

Cycle K = 2,4.... (to update $F(J:K)$'s for the pivot node I's LOWER TO nodes J)

A. Let $F(J:K) = F(J:K-1)$ for all $J = 1,2,\ldots,N$.

B. At each of time T, $T = 1,2,\ldots,N$, do the following:

I. (at pivot node I, where $I = N-T+1$)

a. Check if $F(I:K) = \infty$.

i. If yes, go to Step b.

ii. If not, go to Step c.

b. Check if $F(I:K)$ was set to ∞ due to the message "DISMANTLE" received since the last time node I was the pivot node in cycle K-1, time $T = I+1$.

i. If no, go to time T+1.

ii. If yes, send node I's LOWER TO and HIGHER TO nodes messages: "DISMANTLE" and "I". Then go to time T+1.

c. Check if link (L,I) still exists, where node L, L = H(I:K), is the node from which node I has obtained obtained the current F(I:K).

i. If yes, go to Step d.

ii. If no, let $F(I:K) \leftarrow \infty$ and send each of node I's LOWER TO and HIGHER TO nodes J the messages: "DISMANTLE" and "I". Then go to time T+1.

d. For each of node I's LOWER TO nodes J, check if D(I,J) has been changed since cycle K-2, time T = I+1.

i. If D(I,J) has not change or if it has decreased, compute $F(J:K)^* = F(I:K) + D(I,J)$ and transmit the messages "F(J:K)*" and "I" to node J.

ii. If D(I,J) has increased, send node J the messages: "DISMANTLE" and "I".

Go to time T+1 when Step d is done.

II. (at each of node I's TO nodes J)

a. If node J receives the messages "F(J:K)*" and "I", do the following:

i. Check if $F(J:K) \leftarrow \infty$ due to the message "DISMANTLE" received since cycle K-1, time T = I+1.

If yes, do nothing.

If no, go to Step ii.

ii. Check if $F(J:K)^* < F(J:K)$.

If no, do nothing.

If yes, let $F(J:K) \leftarrow F(J:K)^*$ and $H(J:K) \leftarrow I$.

b. If node J receives the messages: "DISMANTLE" and "I", check if $I = H(J:K)$.

i. If $I \neq H(J:K)$, do nothing.

ii. If $I = H(J:K)$, let $F(J:K) \leftarrow \infty$, and do not update $F(J:K)$ until node J becomes a pivot node.

III. (at all other nodes)

Do nothing.

The flow chart of odd cycles, $K = 1,3,\ldots$, of Algorithm 2 is given in Figure 2. Even cycles work similarly.

The speed with which Algorithm 2 will terminate depends on when the last topological change is made. In the worst case, Algorithm 2 will terminate in N cycles after the last topological change is made. While Algorithm 1 successively builds shortest paths, Algorithm 2 simultaneously builds and dismantles shortest paths whenever such operations are necessary. When there are no topological changes, Algorithm 2 works similar to Algorithm 1. However, when there are topological changes, Algorithm 2 will adapt to the changes in order to find the shortest paths from node 1 to all other nodes. We will examine how Algorithm 2 adapts to topological changes in cycle K, time T as follows. Since odd cycles work similar to even cycles, we will use only an odd cycle K in our discussion.

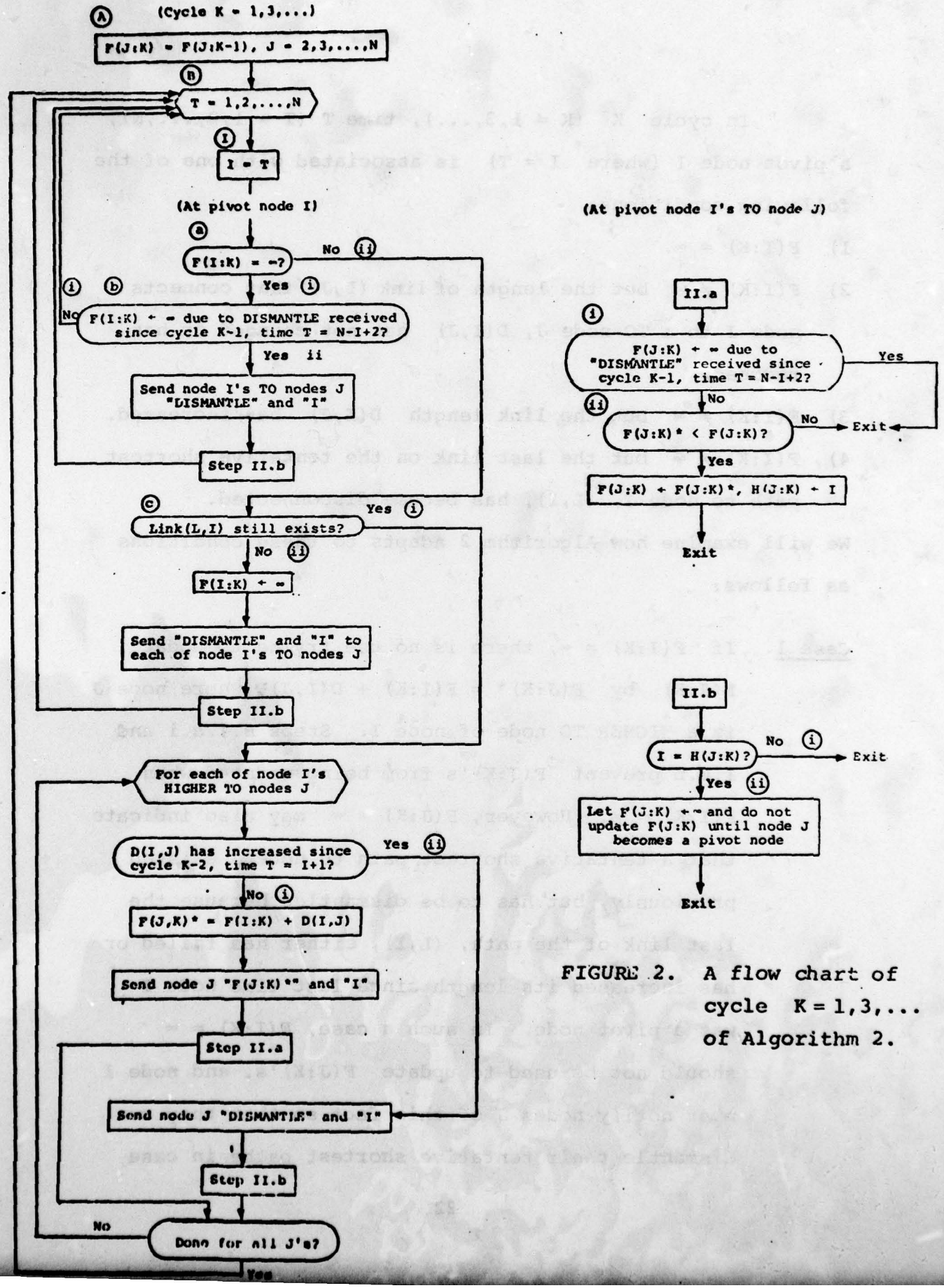


FIGURE 2. A flow chart of cycle K = 1,3,... of Algorithm 2.

In cycle K (K = 1,3,...), time T (T = 1,2,...,N), a pivot node I (where I = T) is associated with one of the following conditions:

1) $F(I:K) = \infty$.
2) $F(I:K) \neq \infty$ but the length of link (I,J) that connects node I to a TO node J, D(I,J) has not changed or has reduced.
3) $F(I:K) \neq \infty$ but the link length D(I,J) has increased.
4) $F(I:K) \neq \infty$ but the last link on the tentative shortest path to node I, (L,I), has become disconnected.

We will examine how Algorithm 2 adapts to these conditions as follows:

Case 1. If $F(I:K) = \infty$, there is no use trying to update F(J:K) by $F(J:K)^* = F(I:K) + D(I,J)$, where node J is a HIGHER TO node of node I. Steps B.I.a.i and I.B.b prevent F(J:K)'s from being updated when $F(I:K) = \infty$. However, $F(J:K) = \infty$ may also indicate that a tentative shortest path to node J existed previously, but has to be dismantled because the last link of the path, (L,I), either has failed or has increased its length since last time node I was a pivot node. In such a case, $F(I:K) = \infty$ should not be used to update F(J:K)'s, and node I must notify nodes J of this fact so that they can dismantle their tentative shortest paths in case

these paths go through node I and use link (L,I) on their paths. Steps B.I.b.ii of the algorithm serves to warn its TO nodes J of the fact that node I has just dismantled its tentative shortest path and that nodes J should also dismantle their tentative shortest paths in case these paths go through node I and use link (L,I) on their paths.

Case 2. If $F(I:K) \neq \infty$ and the link length $D(I,J)$ has been reduced, $F(J:K)$ must be updated by $F(J:K)^* = F(I:K) + D(I,J)$ using the smaller $D(I,J)$. The same process must also be performed to update $F(J:K)$ when there is no change in $D(I,J)$ but $F(I:K) < F(I:K-2)$. In order to simplify the description of the algorithm, Step B.I.d.i of the algorthm updates $F(J:K)$ whenever $F(I:K) \neq \infty$ and $D(I,J)$ is the same or has become smaller.

Case 3. When the length of link (I,J) on a tentative shortest path is increased, the current tentative shortest path to node J may have to be replaced. This is necessary because the increase in link length $D(I,J)$ may cause the current shortest path to become longer than other available paths. Step B.I.d of the algorithm first examines if $D(I,J)$ has been increased since cycle K-2, time T = I+1; and, if so,

Step B.I.d.ii sends node J the messages: "DISMANTLE" and "I". When node J receives these messages, Step B.II.b of the algorithm examines if the tentative shortest path to node J goes through node I and uses link (I,J), by checking to see if $I = H(J:K)$. If $I = H(J:K)$, the tentative shortest path to node J goes through node I and uses link (I,J) on its path. In such a case node J dismantles the tentative shortest path by setting $F(J:K) = \infty$ in Step B.II.b.ii, and informs its own TO nodes to do the same the next time node J becomes a pivot node. In the meantime, in order to help node J remember the fact that D(I,J) has increased, Step B.I.b.ii and B.II.a.i of the algorithm keep $F(J:K) = \infty$ from being updated until node J has its opportunity to send its own TO nodes the messages "DISMANTLE" and "J" when J becomes a pivot node. Thus, with the help of Step B.I.b.ii, the dismantling process goes on in the rest of cycle and in later cycles, until all tentative shortest paths that use link(L,I) are dismantled.

Case 4. When link (L,I), the last link on the tentative shortest path to node J, becomes disconnected, node L cannot communicate directly with node J any longer. Therefore, node J must find out about the disruption of link (L,I) by itself. Prior to updating F(J:K)'s

for node I's HIGHER TO nodes J, Step B.I.c of the algorithm checks to see if the link(L,I) still exists. If link (L,I) still exists, the algorithm will proceed to carry out other necessary steps of the algorithm. However, if link (L,I) no longer exists, Step B.I.c.ii of the algorithm will set $F(I:K) = \infty$ and send node I's TO nodes the message "DISMANTLE" and "I". Then, if necessary, nodes J will dismantle the tentative shortest paths to nodes J by setting $F(J:K) = \infty$, and send the messages further on.

A link in the network can be associated only with on of three topological changes: a decrease in link length (including recovery of the link), an increase in link length, and breakdown of the link. As described above, Algorithm 2 can adapt to any such changes at any links, therefore, the algorithm is adaptive to networks where there are topological changes.

The speed with which Algorithm 2 will terminate depends on when the last topological change is made in the network. When there is no topological change in the network, Algorithm 2 has an efficiency similar to that of Algorithm 1. That is, in the worst case, Algorithm 2 will terminated in N-1 cycles using approximately $N^3/4$ steps and, in the best case, will terminate in two cycles using N^2 steps.

When there are topological changes in cycle K of the algorithm, the algorithm will require in the best case practically no additional effort in order to find all shortest paths, simply because these topological changes have no effect on the current solutions. In the worst case, when there are link recoveries and link length reductions in cycle K, all shortest paths have to be determined anew so that the algorithm has to be executed N-1 more cycles. Also in the worst case, when there are link failures or increases in link lengths in cycle K, the algorithm will require up to N-1 additional cycles to dismantle all relevant shortest paths and up to N-1 additional cycles to update the new shortest paths. However, the update cycles start only about one cycle behind the dismantling cycles, and then are executed simultaneously with the dismantling cycles. Therefore, when there are link failures or increases in link lengths in the network, the algorithm requires, in the worst case, an additional N cycles to determine all new shortest paths. Overall speaking, after the last topological change is made, Algorithm 2 determines all shortest paths from node 1 to all other nodes in N cycles using up to $N^3/4$ steps. While the computational upper bound of Algorithm 2 is one-half or less of Abram and Rhodes' and Segall, Merlin and Gallager's algorithms, as illustrated in example in the last part of Section 2, there are situations where Algorithm 2 can run up to N/2 times faster than these two algorithms.

Algorithm 2 represents a basic adaptive algorithm for finding the shortest path from node 1 to all other nodes. Algorithm 2 can be modified to improve its use and efficiency. For example, each node I can store in its memory all F(I:K)'s that node I has received from all other nodes. Then when node I has to dismantle the best tentative shortest path to node I, it can replace this path immediately with the path that has the next smallest F(I:K). Of course, node I can compute F(J:K)'s and send them to its TO nodes J immediately after the best F(J:K) is replaced by the second-best F(J:K). However, the messages "DISMANTLE" and "I" still should be sent to nodes J so that, if necessary, they can dismantle their best tentative shortest paths and replace them with the second-best tentative shortest paths.

Algorithm 2 can also be modified so that it can be used to find the shortest paths between all pairs of nodes in N cycles. Or, it can be modified so that pivot nodes I do not have to transmit messages at a prearranged time. We have already discussed how to make these two modifications with respect to Algorithm 1.

Acknowledgments. The author would like to thank Professors Mike Sovereign, Jack Wozencraft and Don Gaver for their helpful discussion and encouragement of this research.

REFERENCES

[1] Abram, Jeffrey M. and Ian B. Rhodes, "A decentralized shortest path algorithm," to appear in Proceedings of the 16th Annual Allerton Conference on Communication, Control and Computing, University of Illinois, Oct. 4-6 (1978).

[2] Bertsekas, Dimitri, Eli Gafni, and Kenneth S. Vastola, "Validation of algorithms for optimal routing of flow in networks," Proceedings, 1978 IEEE Conference on Decision and Control, San Diego, CA, Jan. 1979, pp. 220-227.

[3] Boehm, B.W. and R.L. Mobley, "Adaptive routing techniques for distributed communication systems," IEEE Trans. Comm. Tech., COM-17, No. 3, pp. 340-349 (June 1969).

[4] Dreyfus, S.E., "An appraisal of some shortest path algorithms," Oper. Res., Vol. 17, pp. 395-412 (1969).

[5] Frank, H. and I.T. Frisch, Communication, Transmission and Transportation Networks, Addison-Wesley, Reading, Massachusetts (1971).

[6] Fultz, G. F. and L. Kleinrock, "Adaptive routing techniques for store-and-forward computer-communication networks," Proc. IEEE Int. Conf. on Communications, Montreal, pp. 39-1-39-8 (June 1971).

[7] McQuillan, J.M., Adaptive Routing Algorithms for Distributed Computer Networks, Bolt, Beranek, and Newman Report No. 2831, May 1974 (Harvard Ph.D. thesis); available from National Technical Information Service, AD781467.

[8] Naylor, W. E., "A loop-free adaptive routing algorithm for packet switched networks," Proc. 4th Data Communication Symposium, Quebec City, pp. 7-9 to 7-14 (October 1975).

[9] Pierce, A. R., "Bibliography on algorithms for shortest path, shortest spanning tree, and related circuit routing problems (1956-1974)," Networks, Vol. 5, pp. 129-149 (1975).

[10] Schwartz, Mischa, Computer Communication Network Design and Analysis, Prentice-Hall, Englewood Cliffs, N.J. (1977).

[11] Segall, A., P. M. Merlin and R. G. Gallager, "A recoverable protocol for loop-free distributed routing," Proc. of ICC 1978, Toronto.

[12] Tajibnapis, William D., "A correctness proof of a topology information, maintenance protocol for a distributed computer network," Comm. ACM, Vol. 20, No. 7, pp. 477-485 (July 1977).

[13] Yen, J.Y., A Shortest Path Algorithm, Ph.D. dissertation, University of California, Berkeley, CA (1970).

[14] Yen, J.Y., Shortest Path Network Problems, Verlag Anton Hain, Meisenheim, Am Glan, W. Germany (1975).

INITIAL DISTRIBUTION LIST

	NO. OF COPIES
Defense Documentation Center Cameron Station Alexandria, VA 22314	2
Library, Code 0212 Naval Postgraduate School Monterey, CA 93940	2
Library, Code 55 Naval Postgraduate School Monterey, CA 93940	1
Dean of Research Code 012 Naval Postgraduate School Monterey, CA 93940	1
Technical Information Division Naval Research Laboratory Washington, D.C. 20375	1
Office of Naval Research San Francisco Area Office 760 Market Street San Francisco, CA 94102	1
Technical Library Naval Ordnance Station Indian Head, MD 20640	1
Dr. J. Chandra U. S. Army Research P.O. Box 12211 Research Triangle Park, N.C. 27706	1
Dr. M. J. Fischer Defense Communcations Agency 1860 Wiehle Ave., Reston, VA 22070	1
Dr. Richard Lau, Director Office of Naval Research Branch Office 1030 East Green Street Pasadena, CA 91101	1

NO. OF COPIES

Dr. A. R. Laufer, Director 1
Office of Naval Research Branch Office
1030 East Green St
Pasadena, CA 91101

Dr. James R. Maar 1
National Security Agency
Fort Meade, MD 29755

Miss B. S. Orleans 1
Naval Sea Systems Command
(SEA 03F)
Rm 10S08
Arlington, VA 29360

Mr. F. R. Priori 1
Code 224, Operational Test and ONRS
Evaluation Force (OPTEVFOR)
Norfolk, VA 20360

Dr. A. L. Slafkosky 1
Scientific Advisor
Commandant of the Marine Corps
Washington, D.C. 20380

Mr. Glenn F. Stahly 1
National Security Agency
Fort Meade, MD 29755

Mr. David A. Swick 1
Advanced Projects Group
Code 8103
Naval Research Lab.
Washington, D.C. 29375

Navy Library 1
National Space Technology Lab
Attn: Navy Librarian
Bay
St. Louis, MO 39522

Naval Electronic Systems Command 1
Navelex 320
National Center No. 1
Arlington, VA 29360

NO. OF COPIES

Director Naval Research Lab 1
Attn: Library (ONRL)
Code 2029
Washington, D.C. 20375

Naval Postgraduate School
Monterey, CA 93940

Attn:	D. P. Gaver, Code 55Gv	1
	M. G. Sovereign, Code 55Zo	1
	J. M. Wozencraft, Code 74	1
	J. Y. Yen, Code 55Ye	50
	R. J. Stampfel, Code 55	1
	G. T. Howard, Code 55Hk	1
	A. McMasters, Code 55Mg	1