August, 1976 Report ESL-R-683

SHORTEST ROUTE ALGORITHMS FOR SPARSELY CONNECTED NETWORKS

by

Joe E. Defenderfer

Electronic Systems Laboratory
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

This report studies the shortest route problem for networks that are less than fully connected. Two algorithms are presented which exploit the absence of arcs in solving the shortest route problem. The first, which is designated the NXN algorithm, would tend to be the more applicable to networks typically encountered in practice. The second, which is an improvement on Hu's decomposition shortest route algorithm, is more efficient for a small class of networks; however, it generally requires less memory to hold the required decomposition information in the computer than does the NXN algorithm.

Acknowledgements

I wish to thank Professors John M. Wozencraft and Robert G. Gallager who provided encouragement and many helpful discussions.

This work was supported by a Sloan Research Traineeship and the Advanced Research Projects Agency under Contract ONR/N00014-75-C-1183.

Table of Contents

Figures

Tables

Section I Introduction

The problem of finding all the shortest routes in a directed network has an extensive literature [3,9] due to the number of network problems to which shortest route algorithms are applied. This paper presents two new shortest route algorithms which can significantly reduce the required computation time when the network is less than fully connected. The first is based on original decomposition ideas and is called the node-by-node decomposition (NXN) algorithm.† The second is based on Hu's decomposition algorithm [5,6,11] and is designated the improved Hu (IHU) algorithm.

The shortest route problem is formulated as a shortest distance problem where $D = [d_{ij}]$ is a given matrix. The number d_{ij} represents the length of the directed arc from node i to node j, and thus it is assumed $d_{ii} = 0$. A path P from i to j is an ordered sequence $i = k_o, k_1, \ldots, k_{m-1}, k_m = j$, and the length of the path, L(P), is defined as $L(P) = \sum_{r=1}^{m} d_{k_{r-1}k_r}$. If P is any closed path, then it is assumed $L(P) \geq 0$ so that the shortest distance problem is well defined. Then the problem is to find $D^* = [d^*_{ij}]$ where $d^*_{ij} = \min L(P)$ for P ranging over all paths from i to j. Knowing D* alone does not specify the shortest routes, but is a well documented fact that by appropriate bookkeeping as one calculates D*, the shortest routes can also be established.

---

†After completion of this paper, the equivalence of the NXN algorithm with previous work done at Network Analysis Corporation under ARPA Order No. 1523 [8] was discovered.

Typically D* is calculated as a series of refinements on D. Floyd's algorithm [4] is cited for an N node network:

For every i $\varepsilon\{1,2,\ldots,N\}$, do step a:

a) For every j,k $\varepsilon\{1,2,\ldots,N\}$, do step b:

b) $d_{jk} \leftarrow \min\,(d_{jk}, d_{ji} + d_{ik})$

where " $\leftarrow$" means "is replaced by". The algorithm requires N^3 additions and N^3 comparisons, and it is generally assumed additions and comparisons take about the same amount of time so that one says Floyd's algorithms requires $2N^3$ operations. At the conclusion of the algorithm D* has replaced D. Proof of the algorithm is found elsewhere [6], but the interested reader can easily convince himself that when i has been stepped from 1 though i_o then the current value of d_{jk} is the minimal distance over all paths from j to k under the condition that the intermediate nodes are elements of the set $\{1,2,\ldots, i_o\}$.

No algorithm which solves the shortest route algorithm could be any simpler to encode, but there are a variety of faster algorithms in terms of number of operations [7,10]. The standard against which the new decomposition algorithms will be measured is Yen's implementation of Dijkstra's algorithm [2,11] requiring $\frac{3}{2}N^3$ operations. The algorithms claiming even less operations are not significantly faster, theoretically, for networks of the size for which computational experience is cited in this paper; furthermore, some of the apparent gains of the theoretically faster algorithms would be offset by their additional algorithmic complexity.

Section II The NXN Algorithm

The NXN algorithm for solving the shortest route problem is actually a special case of the following new $2N^3$ operation algorithm:

1) For every $i\varepsilon\{1,2,...,N-2\}$ in order, do step a:
   a) For every $j,k\varepsilon\{i+1,i+2,...,N\}$, do step b:
      b) $d_{jk} \leftarrow \min (d_{jk}, d_{ji}+d_{ik})$
2) For every $i\varepsilon\{N-2,N-3,...,1\}$ in order, do step a:
   a) For every $j,k\varepsilon\{i+1,i+2,...,N\}$, do steps b and c:
      b) $d_{ij} \leftarrow \min (d_{ik}+d_{kj}, d_{ij})$
      c) $d_{ji} \leftarrow \min (d_{jk}+d_{ki}, d_{ji})$

An intuitive proof of this algorithm will be helpful in understanding the NXN algorithm. By inductive reasoning similar to that for Floyd's algorithm, when step 1 has been completed for $i=i_o$, then d_{jk} (for $j,k > i_o$) represents the conditional shortest j to k distance subject to all intermediate nodes being elements of the set $\{1,2,...,i_o\}$. Consequently, when step 1 has been completed, then the d_{jk} (for $j,k > N-2$) represent unconditional shortest distances d^*_{ij}.

Note than an arbitrary i to j path (for $j > i$) must be of the form $i,...,r,...,j$ where r is the first element in the path such that $r > i$; and, if this path is the shortest path, then its length is $d^*_{ir}+d^*_{rj}$. When performing step 2 for $i = N-2$, d^*_{rj} is known and d^*_{ir} must be the same as the minimal d_{ir} conditional on all intermediate nodes being elements of the set $\{1,2,\ldots,N-3\}$; it follows that at the end of step 2 for $i = N-2$, $d_{ij} = d^*_{ij}$, and similarly $d_{ji} = d^*_{ji}$ for every $j\varepsilon\{N-1,N\}$. Clearly, inductive reasoning shows that at the end of the algorithm $D = D^*$.

The NXN algorithm will now be presented. However, in order to simplify the discussion, it is assumed that all of the arcs are duplex, i.e. if $d_{ij}<\infty$ then $d_{ji}<\infty$. Define C_i, called the ith connection set, as follows: $j\varepsilon C_i$ if $j > i$ and there exists a path P from i to j such that $L(P) < \infty$ and every intermediate node k satisfies $k < i$. Notice that the C_i are functions of topology only (implicitly assuming the length assigned to an arc is ∞ if and only if the arc does not exist in some sense).

In step 1 of the above algorithm, $d_{ji} = \infty$ if $j\not\in C_i$ and $d_{ik} = \infty$ if $k\not\in C_i$. Furthermore, in step 2 of the above algorithm, $d_{ik} = \infty$ and $d_{ki} = \infty$ if $k\not\in C_i$. The corresponding operations are clearly unnecessary; the algorithm obtained by deleting them is called the NXN algorithm:

1) For every $i\varepsilon\{1,2,\ldots,N-2\}$ in order, do step a:

   a) For every $j,k\varepsilon C_i$, do step b:

      b) $d_{jk} \leftarrow \min\,(d_{jk}, d_{ji}+d_{ik})$

2) For every $i\varepsilon\{N-2, N-3,\ldots,1\}$ in order, do step a:

   a) For every $j\varepsilon\{i+1,i+2,\ldots,N\}$ and $k\varepsilon C_i$, do steps b and c:

      b) $d_{ij} \leftarrow \min\,(d_{ik}+d_{kj},d_{ij})$

      c) $d_{ji} \leftarrow \min\,(d_{jk}+d_{ki},d_{ji})$

A decomposition is defined as an ordering of the nodes. Since the connection sets are a function of the decomposition, the number of operations which the algorithm requires is also a function of the decomposition, as will be demostrated in the following section.

In the case where some of the arcs are not duplex, two alternatives are available. The first is to change the definition of C_i as follows: $j \varepsilon C_i$ if $j > i$ and there exists a path P from i to j or from j to i such that $L(P) < \infty$ and every intermediate node k satisfies $k < i$. This approach causes unnecessary operations for the algorithm. The alternative is to define two connection sets for each node--one for the incoming connections and one for the outgoing connections. In the latter case, one must alter the NXN algorithm to incorporate the efficiencies of the additional connection sets. The increased algorithmic complexity of the second approach and the resultant additional computer steps must be weighed against the number of unnecessary operations of the first approach for the problem at hand.

Section III Decomposing the Network for the NXN Algorithm

This section is introduced via an example. Consider figures 1 and 2 in which the same network has been decomposed two ways. For the first, $C_i = \{i+1, N-1, N\}$ when $i \varepsilon \{1,2,\ldots,N-3\}$ and $C_{N-2} = \{N-1,N\}$; the number of operations for the NXN algorithm is calculated in a straightforward fashion as:

$$\text{Step 1,} \quad (\sum_{i=1}^{N-3} (2)\ (3)\ (3)) + (2)\ (2)\ (2)$$

$$\text{Step 2,} \quad (\sum_{i=1}^{N-3} (2)\ (2)\ (3)\ (N-i)) + (2)\ (2)\ (2)\ (2)$$

which totals $6N^2+12N-66$. By contrast, for the decomposition of figure 2, $C_i = \{i+1, i+2, \ldots, N\}$ which is exactly the same as if the network was fully connected, and it follows immediately that the NXN algorithm requires $2N^3$ operations. This example makes it clear that the choice of decomposition can have a profound effect of the efficiency of the algorithm.

For an arbitrary network, finding the optimal decomposition in the sense of minimizing the required number of operations for the NXN algorithm is not a trivial problem and probably can only be solved by exhaustive comparision. The method of choosing the decomposition for the examples which are presented later in Section IV deviated only slightly from the following heuristic procedure:

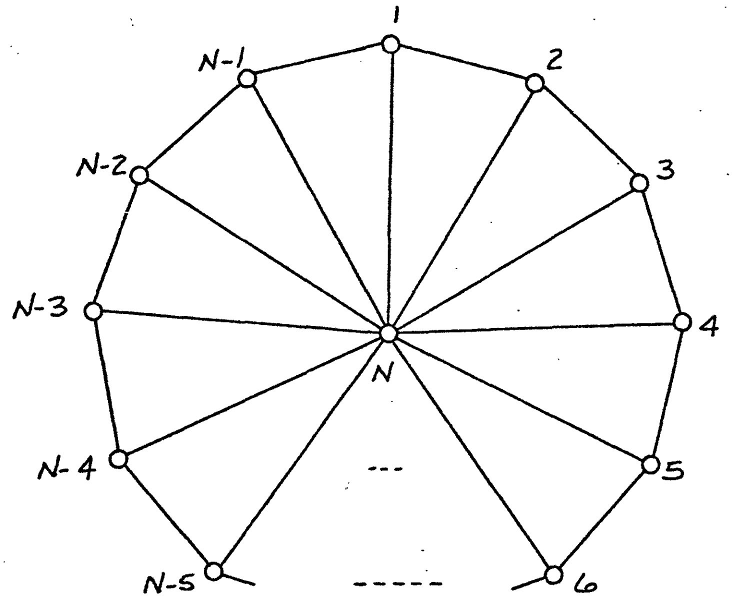

Figure 1. An N node network with an NXN decomposition implied by the numbering of the nodes.

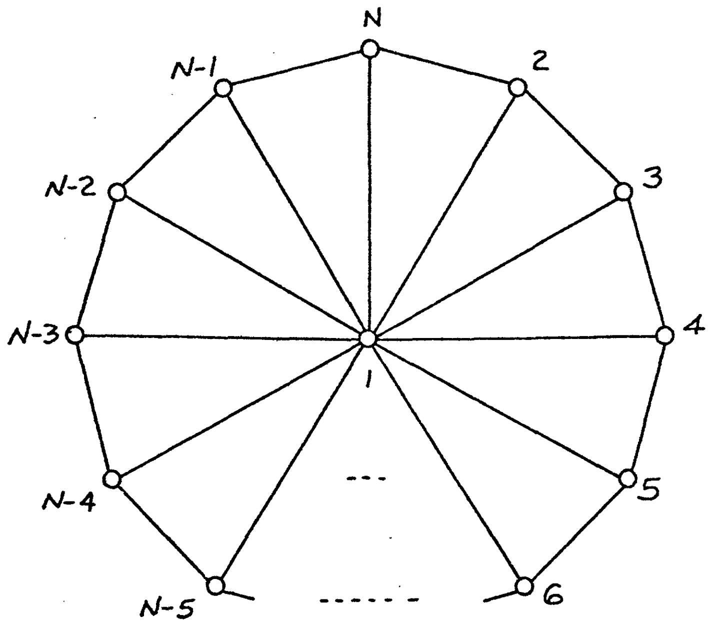

Figure 2. The same N node network as in Figure 1 with a distinct NXN decomposition.

1) Label a node "1" such that the cardinality of C_1 is minimized.

2) For every $i \epsilon \{2,3,\ldots,N\}$ in order, do step a:

   a) Given the nodes which have been labeled "1","2",...,"i-1", label an unlabeled node "i" such that the cardinality of C_i is minimized.

The effort in finding the decomposition via the above procedure is on the same order as doing a shortest route computation via Floyd's algorithm, and as a consequence computer time savings are realized only when the NXN algorithm is iterated several times for the same topology.

There are a large number of networks such that the computation time does not vary widely with the decomposition. Such networks could be termed "locally connected" and have the property that the nodes to which there are direct arcs from any given node are very likely to have direct arcs to one another. In this case, the nodes could be numbered very rapidly by eye with little degradation in efficiency (nodes must at some point in time be assigned a number anyhow in order to communicate the topology to the computer), and in the first shortest route computation the connection sets could be established with very little effort. In fact, the only modification to the NXN algorithm is an additional step which is included just before step 1a:

aa) Initially $C_i = \emptyset$; for $j \epsilon \{i+1,i+2,\ldots,N\}$, do step bb:

bb) Include j in C_i if $d_{ij} < \infty$.

The additional operations required by this step number $\frac{1}{2}N^2$ which is quite modest for the potential gains.

Section IV The IHU Algorithm

The presentation of the IHU algorithm requires some additional definitions. For this algorithm, a network decomposition is defined as a division of the network's nodes into ordered subsets $S_1, S_2, \ldots, S_k$ such that for every $i \in S_m$ and $j \in S_\ell$, $d_{ij} = \infty$ if $|m-\ell| > 1$. Every node of the network belongs to exactly one subset. The submatrix $D_{S_iS_m}$ contains all the distances of arcs from elements of S_i to elements of S_m and has dimension $|S_i| \times |S_m|$ (where $|S|$ means the cardinality of set S). Evidently, $D_{S_iS_m}$ has no finite entries in the case $|i-m| > 1$ (see figure 3).

Various matrix operations will be performed on the submatrices to generate the desired shortest distance matrix. Let $D_{S_iS_i} \leftarrow \xi\, D_{S_iS_i}$ mean $D_{S_iS_i}$ is replaced by the shortest distance matrix computed from the submatrix $D_{S_iS_i}$. Define $A \cdot B = [\min_m (a_{im}+b_{mj})]$ and $\min(A,B) = [\min(a_{ij},b_{ij})]$. Also let $S_1 \cup S_2 \cup \ldots \cup S_m = \Omega_m$, and if m = k (where k is the number of ordered sets) then $\Omega_m = \Omega_k \overset{\Delta}{=} \Omega$. Define the conditional shortest distance submatrix, $D^*_{S_iS_i}(\Omega_m)$, as the shortest distance submatrix under the restriction that all the intermediate nodes on the respective conditional shortest routes are members of Ω_m. In the case m = k, $D^*_{S_iS_i}(\Omega_m) = D^*_{S_iS_j}(\Omega) \overset{\Delta}{=} D^*_{S_iS_j}$.

Under the assumption that an allowable decomposition has been given, the following algorithm generates all the shortest distances in the network (the parenthetical equality to the right of each step is the claim of what each step accomplishes):

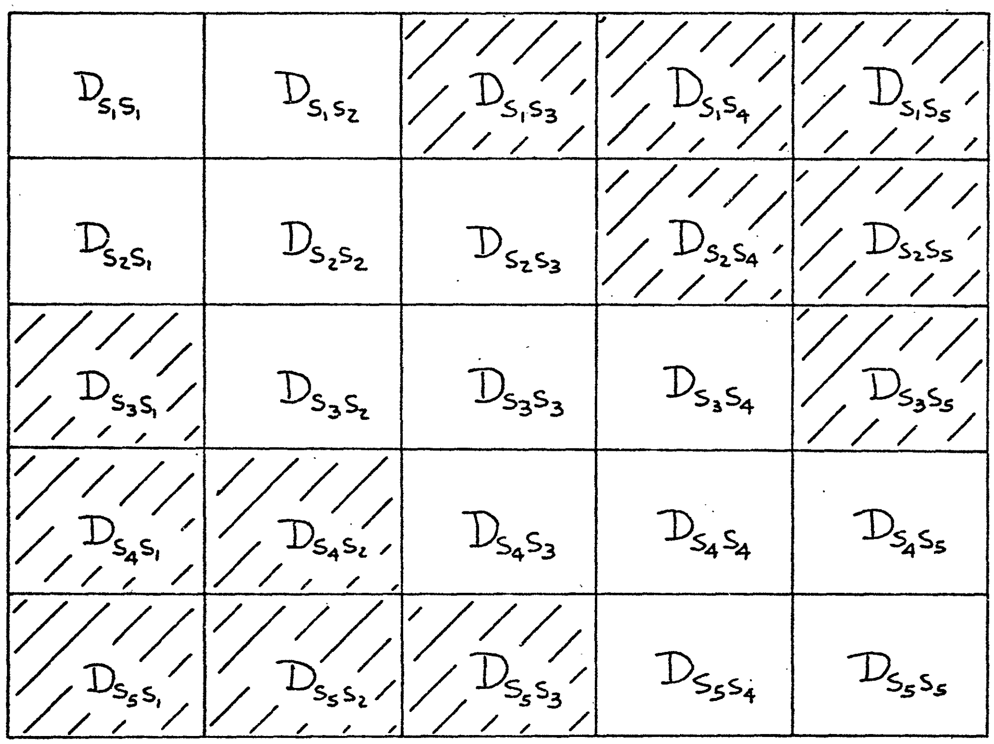

Figure 3. The form of the D matrix for the IHU algorithm in the case $k = 5$. If the decomposition is to be acceptable, the shaded submatrices have no finite entries prior to the algorithmic operations on the D matrix.

1) $D_{S_1S_1} \leftarrow \xi\, D_{S_1S_1}$ $\quad (= D^*_{S_1S_1}(\Omega_1))$

2) For every $i\varepsilon\{1,2,\ldots,k-1\}$in order, do steps a, b, c, and d:

   a) $D_{S_{i+1}S_i} \leftarrow D_{S_{i+1}S_i} \bullet D_{S_iS_i}$ $\quad (= D^*_{S_{i+1}S_i}(\Omega_i))$

   b) $D_{S_iS_{i+1}} \leftarrow D_{S_iS_i} \bullet D_{S_iS_{i+1}}$ $\quad (= D^*_{S_iS_{i+1}}(\Omega_i))$

   c) $D_{S_{i+1}S_{i+1}} \leftarrow \min\,(D_{S_{i+1}S_{i+1}},\, D_{S_{i+1}S_i} \bullet D_{S_iS_{i+1}})$
      $(= D^*_{S_{i+1}S_{i+1}}(\Omega_i))$

   d) $D_{S_{i+1}S_{i+1}} \leftarrow \xi\, D_{S_{i+1}S_{i+1}}$ $\quad (= D^*_{S_{i+1}S_{i+1}}(\Omega_{i+1}))$

3) For every $i\varepsilon\{k,k-1,\ldots,3,2\}$ in order, do steps a, b, and c:

   a) $D_{S_iS_{i-1}} \leftarrow D_{S_iS_i} \bullet D_{S_iS_{i-1}}$ $\quad (= D^*_{S_iS_{i-1}})$

   b) $D_{S_{i-1}S_i} \leftarrow D_{S_iS_{i-1}} \bullet D_{S_iS_i}$ $\quad (=D^*_{S_{i-1}S_i})$

   c) $D_{S_iS_{i-1}} \leftarrow \min\,(D_{S_{i-1}S_i}, D_{S_{i-1}S_i}\, D_{S_iS_{i-1}})$ $\quad (=D^*_{S_{i-1}S_{i-1}})$

4) For every $r\varepsilon\{2,3,\ldots,k-1\}$ in order, do step a:

   a) For every $i,j\varepsilon\{1,2,\ldots k\}$ if $|i-j| = r$, do step b:

      b) $D_{S_iS_j} \leftarrow D_{S_iS_p} \bullet D_{S_pS_j}$ $\quad (=D^*_{S_iS_j})$

where p is an element of the set $Q = \{s+1,s+2,\ldots,t-2,t-2\}$ for $s = \min(i,j)$ and $t = \max(i,j)$ such that $|S_p| \leq |S_m|$ for every $m\varepsilon Q$.

A rigorous proof of the algorithm would be very lengthy and repetitious, and the interested reader is referred to Hu's work [6] for exposition of a similar proof. Steps 1 and 2 are bootstrapping successive diagonal and first off-diagonal submatrices, so that at the end of step 2, $D_{S_kS_k} = D^*_{S_kS_k}$.

Step 3 is essentially a backwards form of step 2 and replaces the diagonal and first off-diagonal submatrices with the respective unconditional shortest distance submatrices. Step 4 is one method for finding the unconditional shortest distance submatrices corresponding to decomposition sets which are separated by at least one intermediate set. The ordering in step 4 allows p to be any element of the set Q, and the particular choice of p minimizes the number of operations.

If one assumes that the shortest distance calculations for submatrices are done via Floyd's method (requiring $2p^3$ operations for a p x p submatrix) and that the pseudo-multiplications are done in a straightforward manner (requiring 2pqr operations to calculate A•B where A is dimension p x q and B is q x r), then the number of operations required by the IHU algorithm is:

Step 1, $2|S_1|^3$

Step 2, $2\sum_{i=1}^{k-1}(2|S_{i+1}|\ |S_i|^2 + |S_i|\ |S_{i+1}|^2 + |S_{i+1}|^3)$

Step 3, $2\sum_{i=2}^{k}(2|S_i|^2\ |S_{i-1}| + |S_{i-1}|^2|S_i|)$

Step 4, $2\sum_{i,j}|S_i|\ |S_j|\ |S_p|$ such that $|i-j| > 1$

The total number of operations is then

$$2\left(\sum_{i=1}^{k-1}|S_i \cup S_{i+1}|^3 - \sum_{i=2}^{k-1}|S|^3 + \sum_{i,j}|S_i|\ |S_j|\ |S_p|\right)$$

such that $|i-j| > 1$

One may compare the IHU algorithm to other versions of Hu's algorithm. For any given decomposition, the IHU algorithm requires fewer operations than the fastest version of Hu's algorithm known to the author, which is that due to Yen [11]. For purposes of comparision, an example which commonly appears in the literature [5,6,11] is presented. Let $|S_i| = \delta$ for i even and $|S_i| = t$ for i odd. Assume $\delta \leq t$, and let k, the number of sets, be odd. Define $m = \frac{k+1}{2}$. In this case, the new algorithm requires $2(mt^3 + (m^2+5m-6)t^2\delta + (2m^2+2m-6)t\delta^2 + (m^2-4m+5)\delta^3)$ operations. Yen's modification requires $2(mt^3 + (m^2+6m-7)t^2\delta + (2m^2+10m-20)t\delta^2 + (m^2+6m-14)\delta^3)$. The new algorithm is faster for the entire range of interest, i.e. $t \geq \delta \geq 1$ and $m \geq 2$. As a particular case, let $\delta = t$ and $m = 3$; the IHU algorithm requires $82t^3$ operations, Yen's modification requires $128t^3$ operations, and Floyd's algorithm requires $250t^3$ operations.

Section V Decomposing the Network for the IHU Algorithm

Perhaps even more important than the numerical gains of the new algorithm are the insights it provides into optimal decomposition of a network. Assume that Floyd's method is used for shortest route computations on submatrices, and that pseudo-multiplications are done by the straightforward technique. It follows that for a given decomposition, if a further decomposition exists by partitioning of existing sets, then the computation time of the further decomposition is less than that of the given decomposition. This "more the better" fact suggests a heuristically good decomposition technique which can be performed by the computer or quickly guessed at by eye. If the decomposition is to be done automatically by the computer, however, it should probably be limited to those cases where many shortest route computations for the same topology will be performed, as in column generating linear programs. An algorithm for finding a good network decomposition for the IHU algorithm is:

a) find two nodes, j and k, such that the minimal number of arcs, d, connecting them is maximal over all pairs of nodes; i.e. find the diameter of the network and an associated pair of nodes;

b) construct d+1 sets by letting $S_1 = \{j\}$ and $S_{i+1} = \{m | m \varepsilon \{\Omega - \Omega_i\}$ and $d_{rm} < \infty$ or $d_{mr} < \infty$ for some $r \varepsilon S_i\}$.

This procedure was used to generate the IHU decomposition sets for the examples of the next section, and the reader may want to look at the figures associated with that section at this point.

Section VI Some Examples Using the IHU and NXN Algorithms

In this section several examples are given which provide insight into the classes of networks for which the NXN and IHU algorithms can substantially reduce shortest route computation time. Although no examples are presented for which the IHU algorithm is faster than the NXN algorithm, they do exist. Such networks form a rather small and special class of networks, and typically may be decomposed in such a manner as to be a variation on the following theme: $|S_i|$ for i odd is large compared to $|S_i|$ for i even, and if $j \varepsilon S_i$ and $k \varepsilon S_i$ then j and k are very likely to have direct arcs to one another.

The first example is an old version of the ARPA net which is shown in figure 4. In that figure, the NXN decomposition is defined by the numbering of the nodes, and the IHU decomposition is defined by the partitioning of the nodes with broken lines. This network lends itself to NXN decomposition due to the high number of nodes which have arcs directly to only two other nodes--a fact which keeps the cardinality of connection sets very low.

The second example is the 47 node symmetric network shown in figure 5. This network is not "locally connected" to a very high degree, but still the NXN algorithm is (perhaps surprisingly) efficient.

The final example is the 64 node network displayed in figure 6. The density of arcs is perhaps greater here than in the other examples, but a high degree of local connectivity promotes the efficiency of the NXN algorithm.

Efficiency is measured with Yen's implementation of Dijkstra's algorithm as the standard. Theoretical efficiency refers to the relative savings in the number of operations required to perform a shortest route computation. The computation times for the IBM 370-168 to execute the Fortran programs of various algorithms were noted, and relative savings are referred to as the measured efficiency. The comparisions of the various algorithms in performing shortest route calculations on the three sample networks are summarized in table 1. The Fortran programs were complied by the IBM G1 compiler; and each algorithm not only computed the shortest distance matrix, but also computed a routing matrix which specified the next node from each node on the shortest route to any other node.

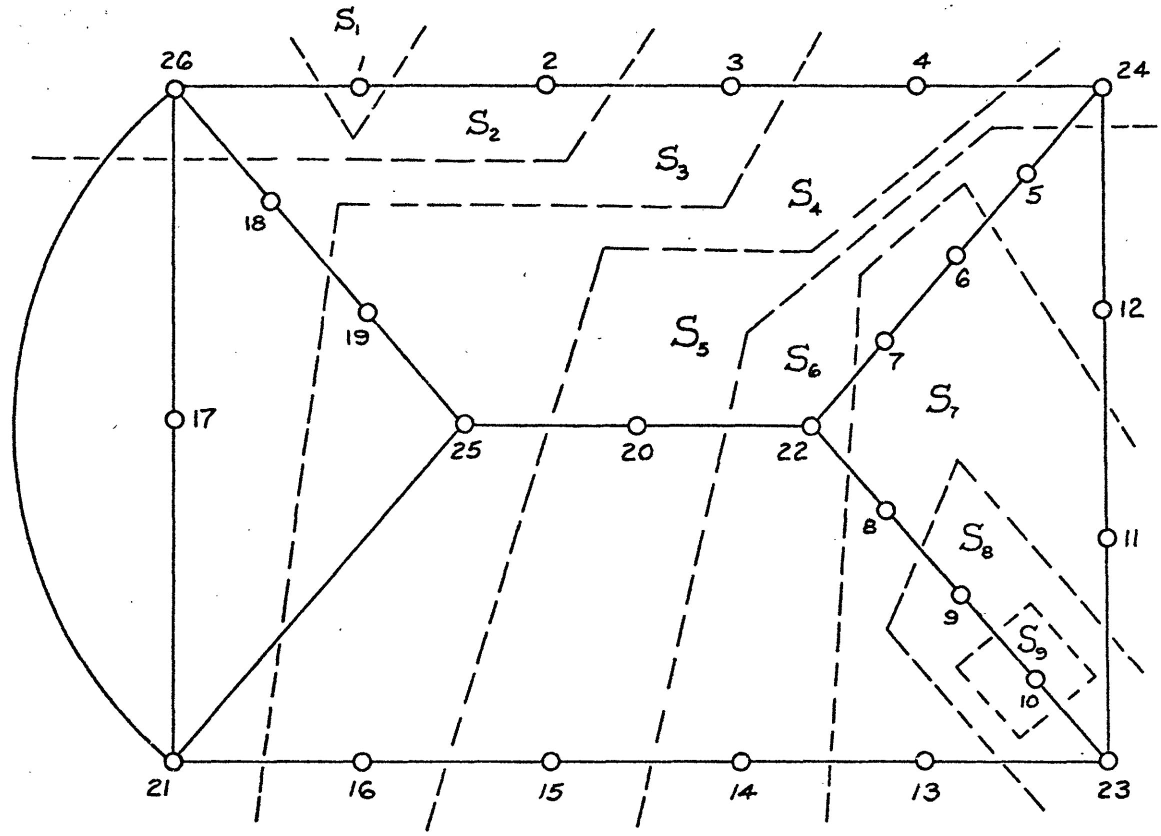

Figure 4. The topology of ARPA network (at one stage of its evolution) with decomposition information.

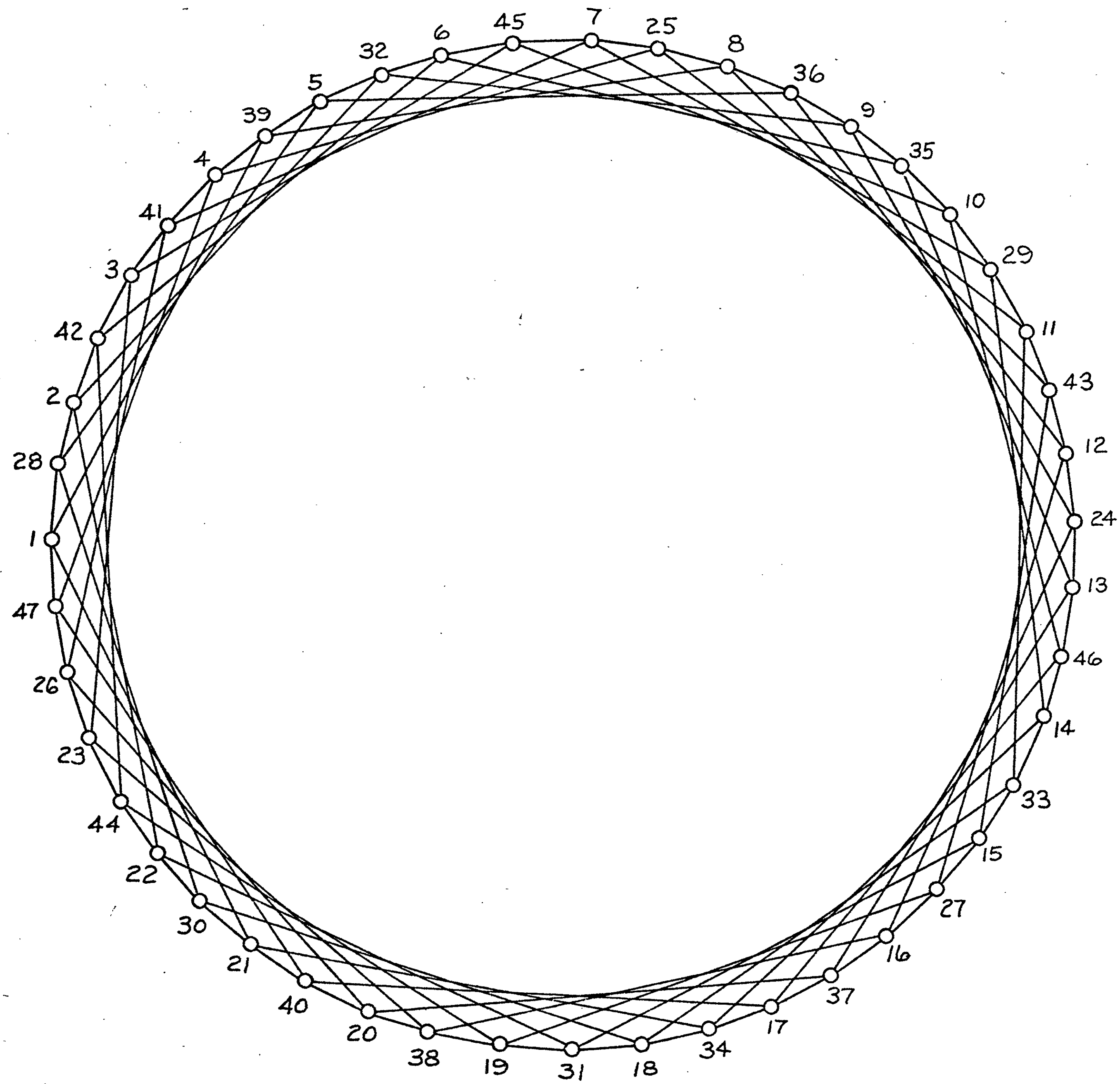

Figure 5. A 47 node symmetric network (nodes are connected to first and seventh nearest neighbors by arcs). NXN decomposition is indicated by node labeling. IHU decomposition sets: S_1 = {1}, S_2 = {21,47,28,39}, S_3 = {34, 40,30,26,2,4,5,8}, S_4 = {14,17,18,29,22,23,42,41,32,25, 36,43}, S_5 = {46,33,15,37,31,38,44,3,6,7,9,29,11,12}, and S_6 = {13,27,16,19,45,35,10,24}.

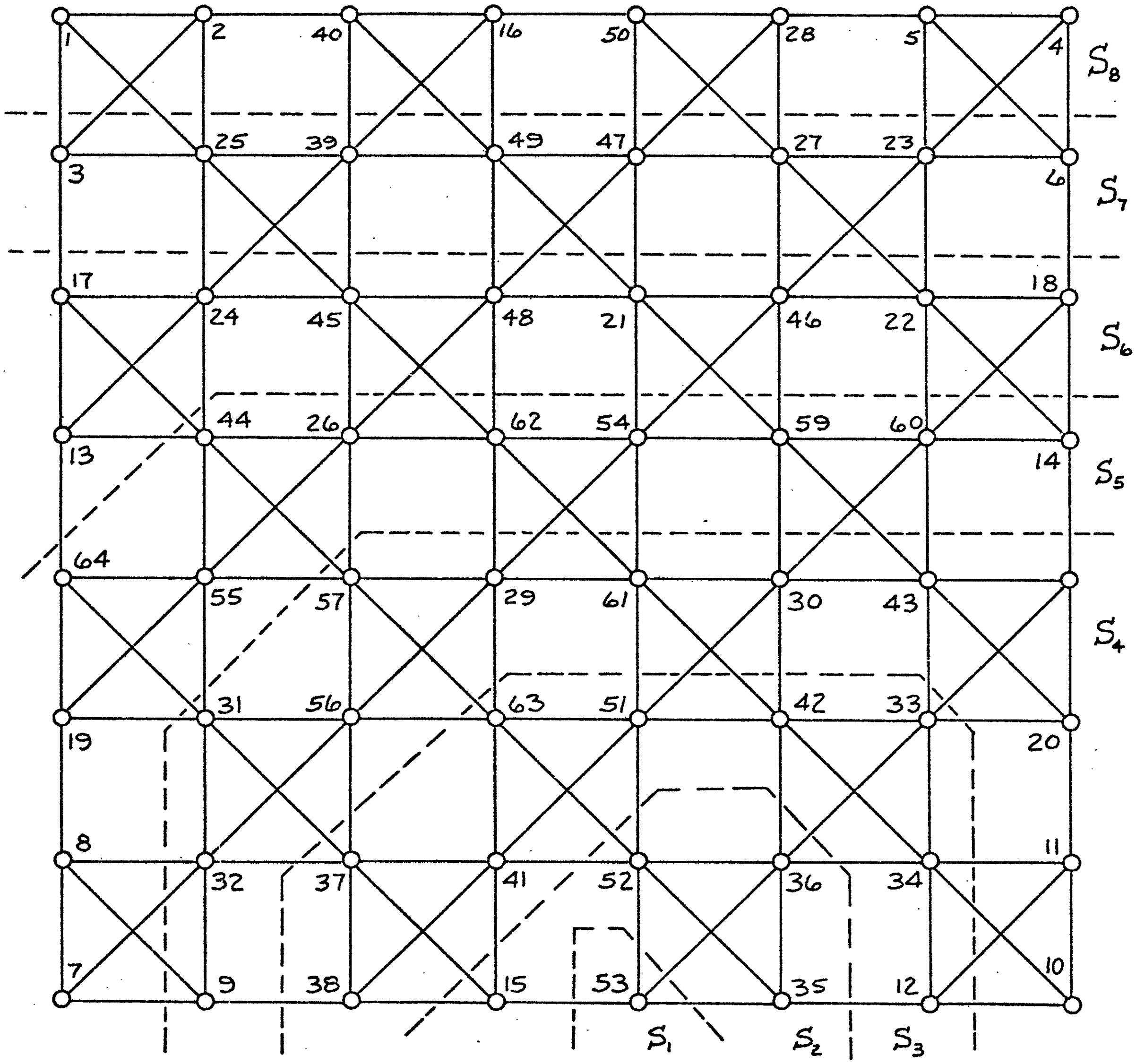

Figure 6. A 64 node network with decomposition information.

		ARPA network of figure 4	47 node network of figure 5	64 node network of figure 6
Dijkstra's shortest route algorithm	Number of operations	27040	154630	393216
	Theoretical efficiency	1.00	1.00	1.00
	Computation time (seconds)	.090	.450	1.115
	Measured efficiency	1.00	1.00	1.00
IH algorithm	Number of operations	6948	83880	124722
	Theoretical efficiency	3.89	1.84	3.15
	Computation time (seconds)	.020	.195	.290
	Measured efficiency	4.50	2.31	3.84
NXN algorithm	Number of operations	2828	28608	42416
	Theoretical efficiency	9.56	5.41	9.27
	Computation time (seconds)	.015	.115	.165
	Measured efficiency	6.00	3.91	6.76

Table 1. Comparative performance of three different shortest route algorithms on the three sample networks.

Appendix

Section IA Introduction

This appendix describes and lists the program which provided the computational experience cited in this paper. The program of section VIIA reads the topology of the network, finds a decomposition for the IHU and NXN algorithms, solves a sample shortest route problem via each algorithm and the Dijkstra algorithm in order to compare computation times, and calculates the number of operations required by each. Typically, an application of these programs requires at most two of the listed subroutines--one to decompose the network and one to calculate all the shortest routes. The decomposition subroutine needs to be called only one time for any given topology since a new set of data cards are punched by the decomposition subroutines which record the appropriate decomposition information. In this appendix, a hybrid notation will be employed which is a combination of that used in the body of this report and that used in the Fortran programs. The definitions of all Fortran terms are given in the comment cards at the beginning of the program listing that is found in section VIIA.

Section IIA Bookkeeping for Shortest Routes

The algorithms which are listed not only find the shortest distances between every pair of nodes in the network, but they also record the shortest routes. The method which is used for this purpose is establishing a "next node" matrix where NX(I,J) is the next node on the shortest path from node I to node J. Initially, NX(I,J) = J for every existing arc (I,J), and every time the operation, $d_{ij} \leftarrow \min (d_{ij}, d_{ik}+d_{kj})$ is performed such that $d_{ik}+d_{kj}$ is the distinct minimum, then the algorithm makes the replacement NX(I,J) $\leftarrow$ NX(I,K). For the remainder of this appendix, the algorithms are discussed only in terms of the shortest distance problem.

Section IIIA <u>The Main Program</u>

The main program reads in the topology, assigns arc numbers and provides the control for its specific purpose, i.e. to compare the various algorithms. In figure 7, an example network is presented. Table 2 lists the data cards which communicate the topology of the network to the program. The first card is a header which provides the name of the network and the values for NN, MIHU, MNXN, MAXPRI and NFORBD. The second card says that node "1" has "2" outgoing arcs which terminate on nodes "2" and "3". There is one such card for each node in succession.

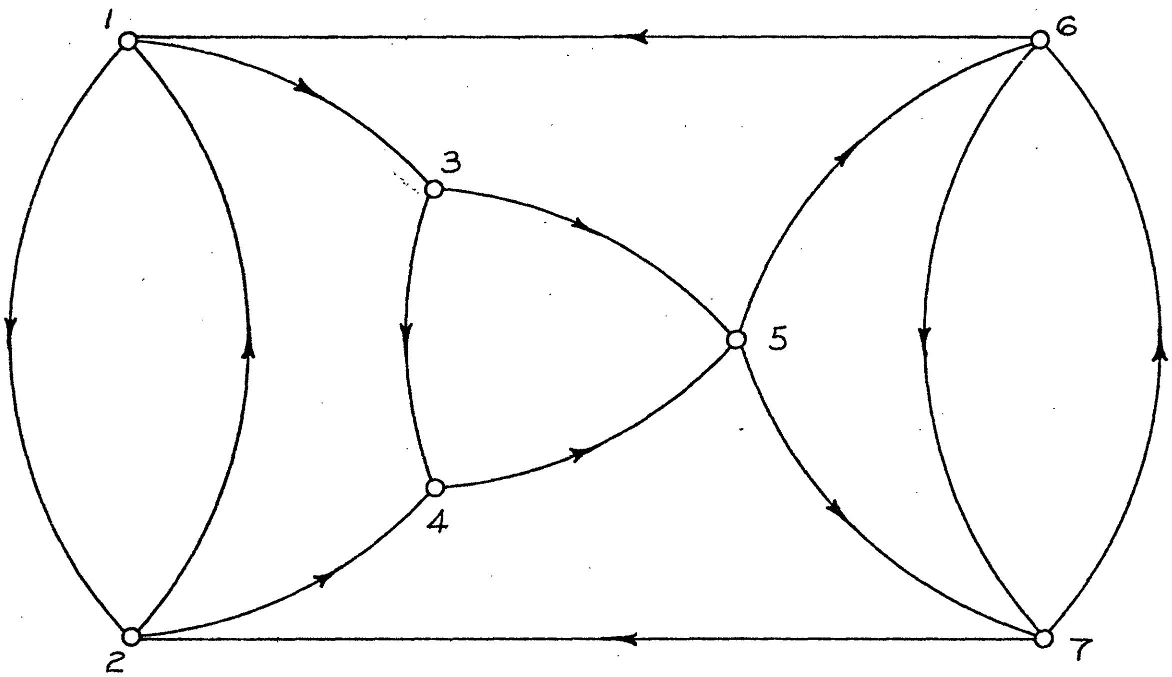

Figure 7. An Example Network With Seven Nodes And Thirteen Arcs.

7 NODE, 13 ARC EXAMPLE NT 7 2 2 7 0

1	2	2	3
2	2	1	4
3	2	4	5
4	1	5	
5	2	6	7
6	2	1	7
7	2	2	6

Table 2. Topology Cards For The Network of Figure 7

Section IVA Subroutine DIJKST

Subroutine DIJKST is an implementation of Dijkstra's algorithm suggested by Yen [12]. The algorithm can be floated to perform the operation, $D_{S_iS_i} \leftarrow \xi D_{S_iS_i}$, in the case that if j is a node number such that $\min_{k\varepsilon S_i} k \leq j \leq \max_{k\varepsilon S_i} k$, then $j\varepsilon S_i$. When the call to the subroutine DIJKST is made, for this case, then $NB = \min_{k\varepsilon S_i} k$ and $NF = \max_{k\varepsilon S_i} k$. If the operation, $D \leftarrow D^*$, is to be performed via Dijkstra's algorithm, then NB = 1 and NF = NN.

Section VA Subroutines DECIHU and IHU

Subroutine DECIHU decomposes the network for the IHU algorithm which is implementated in subroutine IHU. The method of decomposition is that of Section V. Figure 8 shows the network as decomposed by DECIHU with the new node numbers as printed out. Table 3 shows the cards punched by DECIHU which record the decomposition information and describe the topology in terms of the new node numbers. Again, the first card is a header with the title of the network, a "1" which says the cards were punched by DECIHU and a "3" which is the number of IHU sets. The second card says that node "1" has "2" outgoing arcs, is a member of set number "1" (the next two zeros have no significance), and the outgoing nodes are to nodes "2" and "3"; and so forth. The ninth card is a header for NTWIXT which starts on the next card. From them, NTWIXT(1,1) = "0", NTWIXT(1,2) = "0", NTWIXT(1,3) = "2", NTWIXT(2,1) = "0", etc. The information on these cards define the variables found in the common block IHUSTF, and these values are given in Table 4.

Subroutine IHU is a straightforward implementation of the IHU algorithm as presented in Section IV. The operations, $D_{S_i S_i} \leftarrow \xi D_{S_i S_i}$, are performed via subroutine DIJKST.

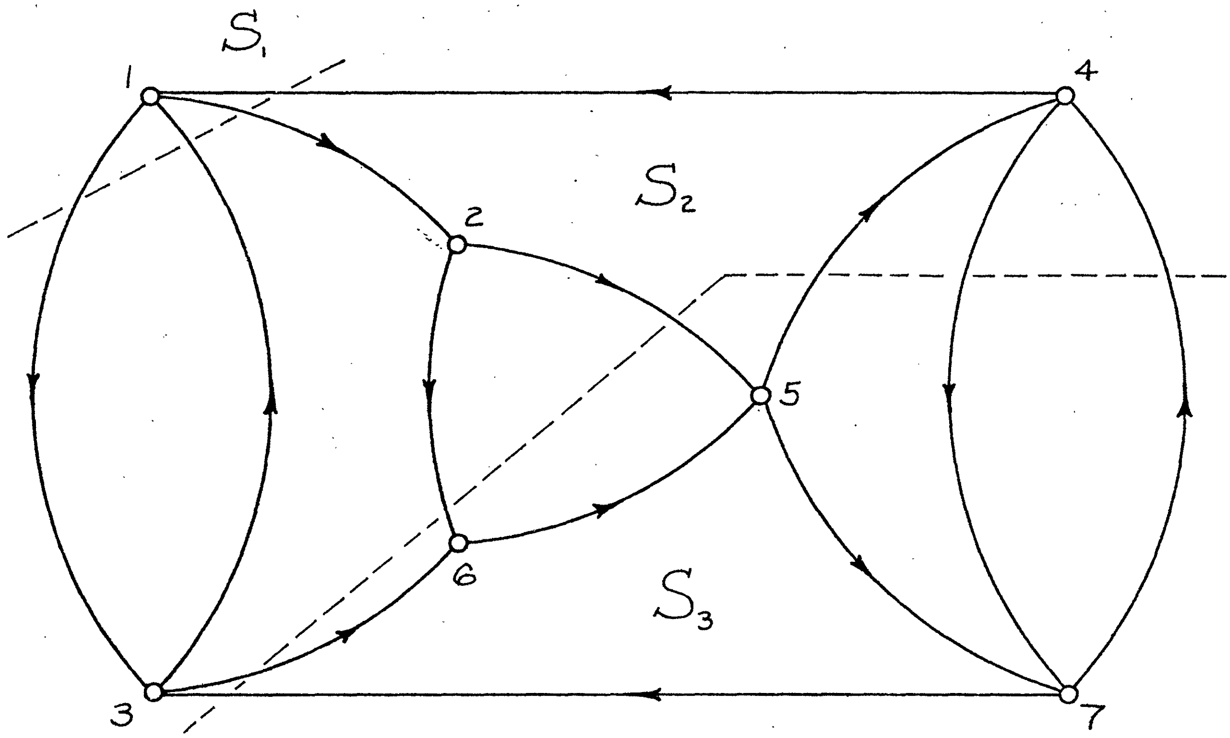

Figure 8. Node Renumbering and Partitioning by Subroutine DECIHU for the Network of Figure 7.

```
7  NODE, 13  ARC  EXAMPLE  NT   7   1   3
   1  2   1   0   0  2  3
   2  2   2   0   0  1  6
   3  2   2   0   0  6  5
   4  2   2   0   0  1  7
   5  2   3   0   0  4  7
   6  1   3   0   0  5
   7  2   3   0   0  2  4
  NTWIXT  FOR  7  NODE,13  ARC  EXAMPLE NT
   0  0   2   0   0  0  3   0   0
```

Table 3. Cards punched by subroutine DECIHU which relate the IHU decomposition information and the topology in terms of the new node numbers for the network of Figure 7.

N1(1) = 1	N1(2) = 2	N1(3) = 5
N2(1) = 1	N2(2) = 4	N2(3) = 7
NTWIXT(1,1) = 0	NTWIXT(2,2) = 0	NTWIXT(1,3) = 2
NTWIXT(2,1) = 0	NTWIXT(2,2) = 0	NTWIXT(2,3) = 0
NTWIXT(3,1) = 3	NTWIXT(3,2) = 0	NTWIXT(3,3) = 0
NS = 3		

Table 4. Values of the variables in labeled common block IHUSTF which may be deduced from cards in Table 3 for the network of Figure 8.

Section VIA Subroutines DECNXN and NXN

Subroutine NXN is a general implementation of the NXN algorithm for the case in which all the arcs in the network are not necesarily duplex. Two connection sets are established for each node--one for outgoing connections and one for incoming connections. Define C_i^O as the outgoing connection set, i.e. $j \varepsilon C_i^O$ if there exists a path P from i to j such that $C(P) < \infty$ and every intermediate node k satisfies $k < i$. Similarly, define C_i^I as the ith incoming connection set. The NXN algorithm takes this form:

1) For every $i \varepsilon \{1,2,\ldots,NN-2\}$ in order, do step a:
   a) For every $j \varepsilon C_i^I$ and $k \varepsilon C_i^O$, do step b:
      b) $d_{jk} \leftarrow \min (d_{jk}, d_{ji}+d_{ik})$
2) For every $i \varepsilon \{NN-2,NN-3,\ldots,1\}$ in order, do step a:
   a) For every $j \varepsilon \{i+1, i+2,\ldots,NN\}$, do steps b and c:
      b) For every $m \varepsilon C_i^O$, $d_{ij} \leftarrow \min (d_{im}+d_{mj}, d_{ij})$
      c) For every $k \varepsilon C_i^I$, $d_{ji} \leftarrow \min (d_{ji}, d_{jk}+d_{ki})$

The method DECNXN uses for decomposing the network is given in Section III with the alteration that nodes are chosen in order to sucessively minimize $|C_i^O| + |C_i^I|$. For the network of Figure 7, the new node numbering which implies the decomposition is shown in Figure 9. The cards punched by DECNXN which contain topology information in terms of new node numbers and the decomposition information are shown in Table 5.

The interpretation of the cards is now more difficult but should be clear by the program in Table 6 which reads in the cards of Table 5, sets up arc numbers, and prepares the decomposition information for DECNXN.

One feature of the program not yet discussed is that of NFORBD which is an input variable. If a network is "locally connected" except for a few nodes, they should be numbered last and suppressed from being assigned new node numbers which are low by establishing NFORBD as the cardinality of the set of such nodes.

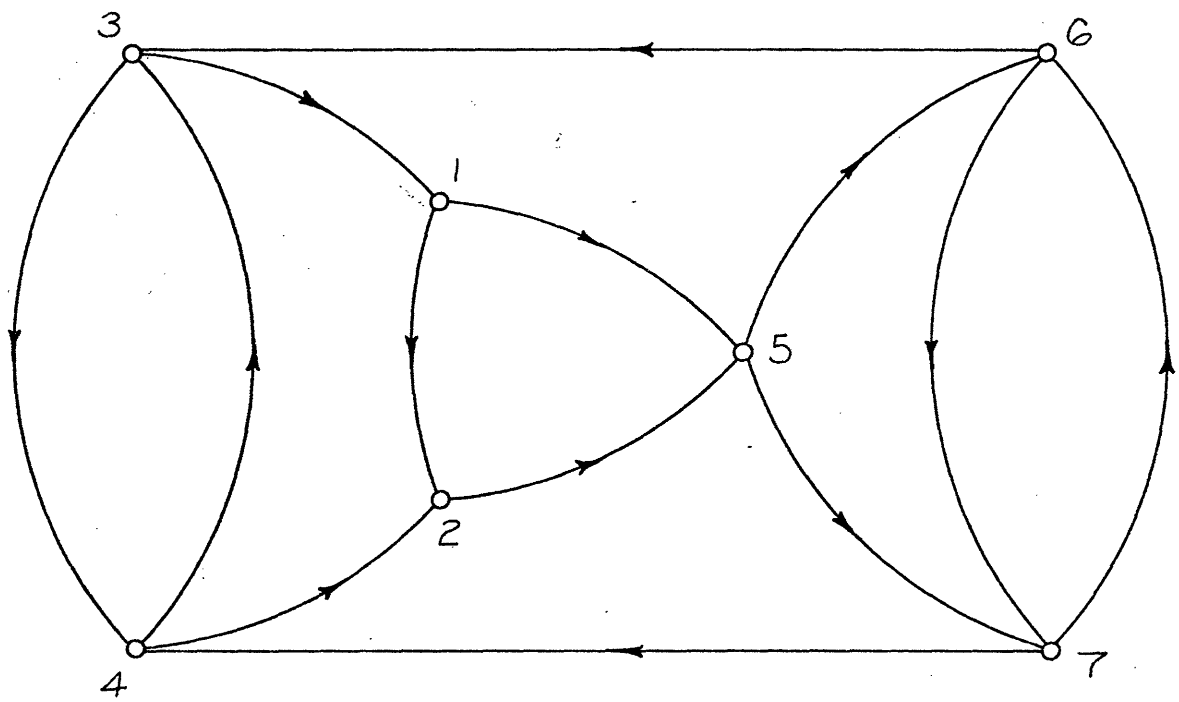

Figure 9. The topology of Figure 7 with new node numbers as assigned by DECNXN

7 NODE, 13 ARC EXAMPLE NT 7 2

1	2	3	2	3	2	3	2	5	3
2	1	2	1	3	5	5	4	3	
3	2	2	2	3	4	1	5	4	6
4	2	2	1	3	3	2	5	6	7
5	2	1	2	2	6	7	6	7	
6	2	1	0	0	3	7			
7	2	1	0	0	4	6			

Table 5. Cards punched by subroutine DECNXN for the network of Figure 7 which contain decomposition information for the NXN algorithm and topology information in terms of the new node numbers.

```
      SUBROUTINE REDNXN
C   THIS IS A SAMPLE SUBROUTINE THAT COULD READ IN CARDS PUNCHED BY
C   DECNXN, ASSIGN ARC NUMBERS, AND DEFINE THE MEMBERS OF THE COMMON
C   BLOCK /NXNSTF/.
      IMPLICIT INTEGER*2 (I-N)
      COMMON /FREE/ F(64),G(64),MA(64),MB(64),MC(64),
     1              A,B,C,X,Y,Z,LA,LB,LC,LD,LE,LF,LU,LV,LW,LX,LY,LZ
      COMMON /STRTSF/ D(64,64),NX(64,64),NN,NB,NF
      COMMON /MAPSTF/ LNK$OR(400),LNK$DS(400),LNKLST(64),
     1                NUMNEW(64),NUMOLD(64),NA
      COMMON /CNTRSF/ TITLE,MIHU,MNXN,NFORBD,MAXNS,MAXCON
      COMMON /NXNSTF/ NC(1024),NO(64),ND(64),NI(64)
      REAL*8 TITLE(3)
      READ 100,TITLE(1),TITLE(2),TITLE(3),NN
  100 FORMAT (3A8,I3)
      IF (LA.EQ.2) GO TO 108
      PRINT 104
  104 FORMAT (' CARDS NOT PUNCHED BY DECNXN')
  108 LX=0
      LY=0
      DO 128 I=1,NN
      READ 112,LA,LB,LC,LD,LE,(MA(J),J=1,LB),(NC(LY+J),J=1,LE)
  112 FORMAT (26I3)
      IF (LA.EQ.I) GO TO 120
      PRINT 116
  116 FORMAT (' INPUT ERRCR')
      STCP
  120 DO 124 J=1,LB
      LX=LX+1
      LNK$OR(LX)=I
  124 INK$DS(IX)=MA(J)
      NO(I)=LC+LY
      ND(I)=LD+IY
      LY=LY+LE
  128 NI(I)=IY
      NA=LX
      RETURN
      END
```

Table 6. A FORTRAN Program that demonstrates the interpretation of the cards punched by DECNXN as shown in Table 5.

Section VIIA Program Listing

The program of this section provided the computational results of this paper. The program is generously commented and should be transparent when studied along with this appendix. In general, clarity was sacrificed for speed only in the subroutines DIJKST, NXN, and IHU.

```
C     THIS PROGRAM DECOMPOSES A NETWORK FOR THE NXN AND IHU ALGORITHMS,
C     CALCULATES THE NUMBER OF COMPUTATION STEPS FOR EACH, AND SOLVES A
C     SAMPLE SHORTEST ROUTE PROBLEM TO COMPARE COMPUTATION TIME.  *** NOTE
C     THAT THE SUBROUTINE TIMING, WHICH IS CALLED ONLY IN THE MAIN PROGRAM
C     AND WHICH KEEPS TRACK OF ACTUAL CPU TIME FOR EACH ALGORITHM, MAY
C     NOT BE AVAILABLE ON ALL MACHINES, PARTICULARLY BY THAT NAME.***
C
C
C     DEFINITIONS-------------
C       DEFINITIONS ARE GIVEN ACCORDING TO LABELED COMMON AREA.  VARIABLES NOT
C       INCLUDED IN COMMON AREAS PERFORM SOME ADMINISTRATIVE FUNCTION WHICH
C       SHOULD BE CLEAR FROM CONTEXT.  REQUIRED DIMENSIONING OF MATRICES IS GIVEN
C       AS A FUNCTION OF:
C           MAXNN- MAXIMUM NUMBER OF NODES.
C           MAXNA- MAXIMUM NUMBER OF ARCS.
C           MAXNS- MAXIMUM NUMBER OF IHU SETS.
C           MAXCON- MAXIMUM NUMBER OF ENTRIES IN NC.
C
C       /FREE/
C            ALL ENTRIES OF FREE HAVE LOCAL DEFINITIONS.  REQUIRED DIMENSIONS ARE:
C               F(MAXNN), G(MAXNN), MA(MAXNN), MB(MAXNN), MC(MAXNN).
C
C       /STRTSF/
C            D(MAXNN,MAXNN)- D(I,J) IS DISTANCE FROM NODE I TO NODE J.  UPON
C               ENTERING SUBROUTINES DIJKST, IHU & NXN, IT REPRESENTS DISTANCE
C               OF THE I TO J ARC (NONEXISTENT ARCS SHOULD HAVE DISTANCE 1.E70;
C               ALSO D(I,I)=0.);  AND UPON LEAVING, IT IS THE SHORTEST I TO J
C               DISTANCE ALONG ANY PATH.
C            NX(MAXNN,NAXNN)- NX(I,J) IS THE NEXT NODE FROM I TO J ALONG THE
C               SHORTEST PATH; UPON ENTERING DIJKST, IHU & NXN, NX(I,J)=J FOR ANY
C               EXISTING I TO J ARC AND NX(I,I)=I.
C            NN- NUMBER OF NODES IN NETWORK.
C            NB- BEGIN NODE FOR SUBROUTINE DIJKST.
C            NF- FINISH NODE FOR SUBROUTINE DIJKST.
C
C       /MAPSTF/
```

```
C       LNK$OR(MAXNA)- LNK$OR(I) IS THE ORIGIN OF THE ITH ARC.
C       LNK$DS(MAXNA)- LNK$DS(I) IS THE DESTINATION OF THE ITH ARC.
C       LNKLST(MAXNN)- LNKLST(I)=K IF K IS THE GREATEST ARC NUMBER
C          SUCH THAT LNK$OR(K)=I.
C       NUMNEW(MAXNN)- SUBROUTINES DECIHU & DECNXN ASSIGN NEW NODE NUMBERS,
C          AND NUMNEW(I) IS THE NEW NODE NUMBER OF OLD NODE NUMBER I.
C       NUMOLD(MAXNN)- NUMOLD(I) IS THE OLD NODE NUMBER OF NEW NODE NUMBER I.
C       NA- NUMBER OF EXISTING ARCS IN NETWORK.
C
C    /CNTRSF/
C       TITLE(3)- NAME OF NETWORK.
C       MIHU- OPTION FOR IHU ALGORITHM (SEE MNXN BELOW).
C       MNXN- OPTION FOR NXN ALGORITHM (OPTIONS: 0- DOES NOTHING WITH
C          RESPECT TO APPROPRIATE ALGORITHM, 1- PERFORMS DECOMPOSITION PLUS
C          SAMPLE SHORTEST ROUTE PROBLEM, 2- SAME AS 1 PLUS PUNCHES
C          DECOMPOSITION INFORMATION).
C       NFORBD- NUMBER OF NODES NOT ALLOWED TO CHANGE NODE NUMBERS IN
C          DECNXN (THEY MUST BE NUMBERED LAST) UNLESS ALL OTHER
C          NODES ARE EXHAUSTED.
C       MAXNS- SEE ABOVE.
C       MAXCON- SEE ABOVE.
C
C    /IHUSTF/
C       NTWIXT(MAXNS,NAXNS)- NTWIXT(I,J) IS SET NUMBER OF LOWEST
C          CARDINALITY BETWEEN SETS I AND J (IFJ>I) AND IS RESPECTIVE
C          CARDINALITY (IF J<I).
C       N1(MAXNS)- N1(I) IS FIRST NODE WHICH IS ELEMENT OF IHU SET I.
C       N2(MAXNS)- N2(I) IS LAST NODE WHICH IS ELEMENT OF IHU SET I.
C       NS- NUMBER OF IHU SETS.
C
C    /NXNSTF/
C       NC(MAXCON)- STORES CONNECTION SETS FOR NXN ALGORITHMS.
C       NO(MAXNN)- NO(I) IS LOCATION IN NC OF FIRST NODE OF ITH CONNECTION
C          SET SUCH THAT IT IS AN INWARD NODE.
C       ND(MAXNN)- ND(I) IS LOCATION IN NC OF LAST NODE OF ITH CONNECTION
C          SET SUCH THAT IT IS AN OUTWARD NODE.
```

```
C         NI(MAXNN)- NI(I) IS LOCATION IN NC OF LAST NODE OF ITH CONNECTION
C            SET.
C
C     INPUT VARIABLE,NOT MENTIONED ABOVE:
C         MAXPRI- SAMPLE SHORTEST DISTANCE AND SAMPLE NX MAXTRICES
C            ARE PRINTED UP THROUGH THE MAXPRI'TH ROW (ONE MAY SET
C            MAXPRI EQUAL TO ZERO).
C
C     ASSUMING REALS ARE REAL*4 AND INTEGERS ARE INTEGER*2, THEN
C        THE MEMORY CONSUMED BY MATRICES IS:
C           26*MAXNN+6*MAXNN**2+4*MAXNA+4*MAXNS+2*MAXNS**2+2*MAXCON
C        WHICH IS ABOUT 32K BYTES IF MAXNN=64 MAXNA=400, MAXNS=26 AND
C        MAXCON=1024.
```

```
      IMPLICIT INTEGER*2 (I-N)
      COMMON /FREE/ F(64),G(64),MA(64),MB(64),MC(64),
     1              A,B,C,X,Y,Z,LA,LB,LC,LD,LE,LF,LU,LV,LW,LX,LY,LZ
      COMMON /STRTSF/ D(64,64),NX(64,64),NN,NB,NF
      COMMON /MAPSTF/ LNK$OR(400),LNK$DS(400),LNKLST(64),
     1                NUMNEW(64),NUMOLD(64),NA
      COMMON /CNTRSF/ TITLE,MIHU,MNXN,NFORBD,MAXNS,MAXCON
      COMMON /IHUSTF/ NTWIXT(26,26),N1(26),N2(26),NS
      COMMON /NXNSTF/ NC(1024),NO(64),ND(64),NI(64)
      INTEGER*4 ISTOP,ISTART,NSTEPS
      REAL*8 TITLE(3),ALGORM(4)
      DATA ALGORM/'DIJKSTRA','IHU','NXN'/
C
C  LIMITS ON THE PROGRAM (MAXNN LIMITS # OF NODES;  MAXNA LIMITS # OF
C    ARCS;  MAXNS LIMITS # OF IHU SETS;,  MAXCON LIMITS # OF ENTRIES IN
C    NC WHICH HOLDS NXN CONNECTION SETS).
      MAXNN=64
      MAXNA=400
      MAXNS=26
      MAXCON=1024
C
C  READ IN NETS (THE NUMBER OF NETWORKS TO BE DECOMPOSED).
  100 FOFMAT (26I3)
      READ 100,NETS
      IF (NETS.LE.0) NETS=1
  108 IF (NETS.IE.0) STOP
C
C  READ IN TOPOLOGY OF THE NETWORK.
      READ 116,TITLE(1),TITLE(2),TITLE(3),NN,MIHU,MNXN,MAXPRI,NFORBD
  116 FORMAT (3A8,10I3)
      IZ=0
      DO 128 I=1,NN
      READ 100,LA,IB,LC,LD,LE,(MA(J),J=1,LB)
      IF (LA.NE.I) GO TO 2088
      DO 124 J=1,IB
      LZ=IZ+1
```

```
      LNK$OR(IZ)=I
  124 LNK$DS(LZ)=MA(J)
  128 LNKLST(I)=IZ
      NA=LZ
      IF (NN.GT.MAXNN.OR.NA.GT.MAXNA) GO TO 2072
      PRINT 132,TITLE(1),TITLE(2),TITLE(3)
  132 FORMAT ('1',3A8)
C
C
C  CONSTRUCT A SAMPLE DISTANCE MATRIX (AS IF EVERY ARC IS DUPLEX,
C    LOOKING FORWARD TO DECIHU) AND FIND THE SHORTEST ROUTES VIA
C    DIJKSTRA'S ALGORITHM.
      LOCPNT=1
      DO 148 I=1,NN
      DO 140 J=1,NN
  140 D(I,J)=1.E70
      NX(I,I)=I
  148 D(I,I)=0.
      DO 156 I=1,NA
      LA=LNK$OR(I)
      LB=LNK$DS(I)
      D(LB,LA)=1.
      NX(LB,LA)=LA
      D(LA,LB)=1.
  156 NX(LA,LB)=LB
      CALL TIMING (ISTART)
      NB=1
      NF=NN
      CALL DIJKST
      CALL TIMING (ISTOP)
      ISTOP=ISTOP-ISTART
      NSTEPS=NN*NN*(NN-(NN+1)/4)*2
      PRINT 164,ALGORM(1),TITLE(1),TITLE(2),TITLE(3)
  164 FORMAT (///' THE FOLLOWING INFORMATION RELATES TO THE ',A8,
     1  ' SHORTEST ROUTE ALGORITHM FOR ',3A8,':')
      GO TO 2000
```

```
C
C
C  THIS SECTION CONTROLS THE DECOMPOSITION FOR THE IHU ALGORITHM.
  200 IF (MIHU.LE.0) GO TC 300
      LOCPNT=2
      PRINT 164,ALGORM(2),TITLE(1),TITLE(2),TITLE(3)
      CALL DECIHU
      IF (NS.GT.MAXNS) GO TO 300
C
C  CONSTRUCT A SAMPLE DISTANCE MATRIX AND FIND SHORTEST ROUTES VIA IHU.
  204 DO 216 I=1,NN
      DO 208 J=1,NN
  208 D(I,J)=1.E70
  216 D(I,I)=0.
      DO 224 I=1,NA
      LA=NUMNEW(LNK$OR(I))
      LB=NUMNEW(LNK$DS(I))
      D(LA,LB)=1.
  224 NX(LA,LB)=LB
      CALL TIMING (ISTART)
      IF (LOCPNT.GT.2) GO TO 340
      CALL IHU
      CALL TIMING (ISTOP)
      ISTOP=ISTOP-ISTART
C
C  COMPUTE NSTEPS FOR IHU
      NSTEPS=0
      DO 240 I=1,NS
      LA=N2(I)-N1(I)+1
      NSTEPS=NSTEPS-LA*LA*LA
  240 MA(I)=LA
      LB=MA(1)
      DO 248 I=2,NS
      LC=MA(I)
      LB=LB+LC
      NSTEPS=NSTEPS+LB*LB*LB
```

```
  248 LB=LC
      LA=1
      IF (NS.LE.2) GO TO 272
      DO 264 I=3,NS
      DO 256 J=1,IA
  256 NSTEPS=NSTEPS+MA(I)*MA(J)*NTWIXT(I,J)*2
  264 LA=IA+1
  272 NSTEPS=NSTEPS*2
      GO TO 2000
C
C
C  THIS SECTION CONTROLS DECOMPOSITION FOR THE NXN ALGORITHM.
  300 IF(MNXN.LE.0) GO TO 400
      LOCPNT=3
      PRINT 164,ALGORM(3),TITLE(1),TITLE(2),TITLE(3)
      CALL DECNXN
      IF (MNXN.LT.0) GO TO 2064
C
C  COMPUTE NUMBER OF COMPUTATION STEPS FOR NXN ALGORITHM.
      NSTEPS=0
      LC=NN-2
      IF (LC.LT.1) GO TO 332
      DO 324 I=1,LC
      IA=MA(I)
      LB=MB(I)
  324 NSTEPS=NSTEPS+LA*(LB-1)+(NN-I)*(LA+LB)
  332 NSTEPS=NSTEPS*2
C
C  CONSTRUCT A SAMPLE DISTANCE MATRIX & SOLVE VIA THE NXN ALGORITHM.
      GO TO 204
  340 CALL NXN
      CALL TIMING (ISTOP)
      ISTOP=ISTOP-ISTART
      GO TO 2000
  400 NETS=NETS-1
      GO TO 108
```

```
C
C
C  THIS SECTION CONTROLS THE MAJORITY OF THE PRINT OUT.
 2000 PRINT 2008,NSTEPS,ISTOP
 2008 FORMAT ('ONUMBER OF COMPUTATION STEPS=',I10,16X,
     1    'COMPUTATION TIME=',I6)
      IF (LOCPNT.LE.1.OR.MAXPRI.LE.0)  GO TO 2064
      PRINT 2016,(I,I=1,NN)
 2016 FORMAT ('0SAMPLE DISTANCE MATRIX IN TERMS OF NEW NODE NUMBERS:'//
     1   6(5X,25(I3,'D')/))
      IF (MAXPRI.GT.NN)  MAXPRI=NN
      DO 2032 I=1,MAXPRI
      DO 2024 J=1,NN
 2024 MA(J)=IFIX(D(I,J)+.5)
 2032 PRINT 2040,I,(MA(J),J=1,NN)
 2040 FORMAT (1X,I3,'S',25I4/5(5X,25I4/))
      PRINT 2048,(I,I=1,NN)
 2048 FORMAT ('0SAMPLE NEXT NODE MATRIX IN TERMS OF NEW NODE NUMBERS:'
     1     //6(5X,25(I3,'D')/))
      DO 2056 I=1,MAXPRI
 2056 PRINT 2040,I,(NX(I,J),J=1,NN)
 2064 GO TO (200,300,400), LOCPNT
C  COME HERE IN CASE OF EXCEEDING PROGRAM LIMITS
 2072 PRINT 2080
 2080 FORMAT(' PROGRAM LIMITS HAVE BEEN EXCEEDED BY # OF NODES OR ARCS')
      GO TO 108
C  COME HERE IN CASE OF AN INPUT ERROR AND STOP THE PROGRAM
 2088 PRINT 2096
 2096 FORMAT (' INPUT ERROR IN READING OF TOPOLOGY')
      STOP
      END
```

```
      SUBROUTINE DIJKST
C
C
C  THIS IS A STRAIGHTFORWARD INTERPRETATION OF DIJKSTRA'S SHORTEST ROUTE
C    ALGORITHM <NUMERISCHE MATHEMATIC,VOL#1,PP.269,1959> AS ENCODED BY
C    YEN <J.ASSOC.COMPUT.MACH.,VOL.19,NO.3,PP.423,JULY 1972> WITH THE
C    ADDITION THAT A ROUTING MATRIX, NX, IS KEPT , AND THE PROVISION
C    FOR FLOATING ALGORITHM VIA NB AND NF.
C
C  THESE VARIABLES MUST BE DEFINED UPON ENTRANCE TO THIS SUBROUTINE:
C          D,NX,NB,NF.
C  THESE VARIABLES ARE DEFINED OR REDEFINED BY THIS SUBROUTINE:
C          D,NX.
C
      IMPLICIT INTEGER*2 (I-N)
      COMMON /FREE/ F(64),G(64),MA(64),MB(64),MC(64),
     1              A,B,C,X,Y,Z,LA,LB,LC,LD,LE,LF,LU,LV,LW,LX,LY,LZ
      COMMON /STRTSF/ D(64,64),NX(64,64),NN,NB,NF
  900 IF (NF.LE.NB) GO TO 918
      LU=NB+1
      DO 916 N=NB,NF
      DO 904 M=LU,NF
      MA(M)=M
      F(M)=1.E70
  904 MB(M)=N
      MA(N)=NB
      LZ=N
      Z=0.
      DO 916 M=LU,NF
      A=1.E70
      DO 914 L=M,NF
      B=Z+D(MA(I),IZ)
      C=F(L)
      IF (B.GE.C) GO TO 910
      C=B
      F(I)=B
```

```
      MB(L)=LZ
  910 IF (C.GE.A) GO TO 914
      A=C
      LA=L
  914 CONTINUE
      Z=A
      LZ=MA(LA)
      D(LZ,N)=A
      NX(LZ,N)=NX(IZ,MB(LA))
      MB(LA)=MB(M)
      MA(IA)=MA(M)
  916 F(LA)=F(M)
  918 RETURN
      END
```

```
      SUBROUTINE DECIHU
C
C  THIS SUBROUTINE PERFORMS THE DECOMPOSITION FOR THE IHU SHORTEST ROUTE
C    ALGORITHM.  UPON ENTERING THIS SUBROUTINE IT IS ASSUMED THAT D(I,J)
C    IS THE MINIMUM NUMBER OF ARCS BETWEEN NODES I AND J WHERE ARCS ARE
C    CONSIDERED AS UNDIRECTED.
C
C  THESE VARIABLES MUST BE DEFINED UPON ENTRANCE TO THIS SUBROUTINE:
C          D,NN,TITLE,MIHU,MAXNS,LNKLST,LNK$DS.
C  THESE VARIABLES ARE DEFINED OR REDEFINED BY THIS SUBROUTINE:
C          NTWIXT,N1,N2,NS,NUMNEW,NUMOLD.
C
      IMPLICIT INTEGER*2 (I-N)
      COMMON /FREE/ F(64),G(64),MA(64),MB(64),MC(64),
     1              A,B,C,X,Y,Z,LA,LB,LC,LD,LE,LF,LU,LV,LW,LX,LY,LZ
      COMMON /IHUSTF/ NTWIXT(26,26),N1(26),N2(26),NS
      COMMON /STRTSF/ D(64,64),NX(64,64),NN,NB,NF
      COMMON /MAPSTF/ LNK$OR(400),LNK$DS(400),LNKLST(64),
     1                NUMNEW(64),NUMOLD(64),NA
      COMMON /CNTRSF/ TITLE,MIHU,MNXN,NFORBD,MAXNS,MAXCON
      REAL*8 TITLE(3)
C
C  FIND THE DIAMETER OF THE NET, A, AND AN ASSOCIATED NODE, LA.
      A=0.
      DO 3012 I=1,NN
      DO 3012 J=1,NN
      B=D(I,J)
      IF (B.LE.A) GO TO 3012
      LA=I
      A=B
 3012 CONTINUE
      NS=IFIX(A+1.5)
      IF (NS.GT.MAXNS) GO TO 3148
C
C  ESTABLISH NEW NODE NUMBERS AND HU DECOMPOSITION SETS;  MA(I) STORES
C    MINIMAL # OF ARCS TO NODE LA.
```

```
      DO 3020 I=1,NN
      MA(I)=IFIX(D(IA,I)+.5)
 3020 NUMOLD(I)=I
      LU=1
      LV=1
      LW=0
      DO 3036 I=1,NS
      N1(I)=LU
      DO 3028 J=LV,NN
      IF (MA(J).NE.LW) GO TO 3028
      MA(J)=MA(IU)
      MA(LU)=I
      LB=NUMOLD(IU)
      NUMOLD(LU)=NUMOLD(J)
      NUMOLD(J)=LB
      LU=IU+1
 3028 CONTINUE
      MB(I)=LU-IV
      LW=I
      IV=IU
      N2(I)=LU-1
      LB=I+1
      IF (LB.GT.MAXNS) LB=I
      NTWIXT(I,I)=0
      NTWIXT(I,LB)=0
 3036 NTWIXT(LB,I)=0
C
C  FIND NTWIXT(I,J).  IF J>I, THEN NTWIXT(I,J) IS THE SET OF MINIMUM
C    CARDINALITY BETWEEN SETS I AND J;  AND NTWIXT(J,I) IS THE RESPECTIVE
C    CARDINALITY.  FROM ABOVE, MB(I) IS CARDINALITY OF SET I.
      IE=NS-2
      IF (LE.LE.0) GO TO 3060
      DO 3052 I=1,IE
      LA=32000
      LD=I+1
      IF=I+2
```

```
      DO 3052 J=LF,NS
      IC=MB(LD)
      IF (LC.GE.IA) GO TO 3044
      LB=LD
      LA=LC
 3044 ID=J
      NTWIXT(I,J)=LB
 3052 NTWIXT(J,I)=IA
 3060 DO 3068 I=1,NN
 3068 NUMNEW(NUMOLD(I))=I
C
C
C  PRINT OUT AND PUNCH OUT DECOMPOSITION DATA;  NOTE THAT MA(I) IS NOW
C     THE SET NUMBER OF THE NEW NODE NUMBER I.
      PRINT 3076
 3076 FORMAT ('0NODE CONVERSION DATA FOR IHU DECOMPOSITION:')
      LB=0
 3078 LA=LB+1
      LB=LB+25
 3080 IF (NN.LT.LB) LB=NN
      PRINT 3084,(I,I=LA,LB)
 3084 FORMAT ('0NEW NODE NUMBER',5X,25I4)
      PRINT 3088,(NUMOLD(I),I=LA,LB)
 3088 FORMAT (' OLD NODE NUMBER',5X,25I4)
      PRINT 3092,(MA(I),I=LA,LB)
 3092 FORMAT (' IHU SET NUMBER',6X,25I4)
      IF (LB.LT.NN) GO TO 3078
 3100 IF (MIHU.LE.1) GO TC 3140
      IA=1
      PUNCH 3108,TITLE(1),TITLE(2),TITLE(3),NN,LA,NS
 3108 FORMAT (3A8,3I3)
      LA=0
      DO 3112 I=1,NN
      LF=NUMOLD(I)
      LB=1
      IF (LF.GT.1) LB=LNKLST(LF-1)+1
```

```
      LC=INKIST(IF)
      LD=LC-LB+1
 3112 PUNCH 3116,I,LD,MA(I),LA,LA,(NUMNEW(LNK$DS(J)),J=LB,LC)
 3116 FORMAT (26I3)
     'PUNCH3132,TITLE(1),TITLE(2),TITLE(3),((NTWIXT(I,J),J=1,NS),I=1,NS)
 3132 FORMAT (' NTWIXT FOR ',3A8/26(26I3/))
 3140 RETURN
 3148 PRINT 3156
 3156 FORMAT (' TOO MANY IHU SETS')
      GO TO 3140
      END
```

```
      SUBROUTINE IHU
C  THIS IS THE IHU ALGORITHM FOR FINDING ALL THE SHORTEST ROUTES IN A
C    DIRECTED GRAPH.  STEP NUMBERS REFER TO THOSE IN 'SHORTEST ROUTE
C    ALGORITHMS FOR SPARSELY CONNECTED NETWORKS' BY J.E. DEFENDERFER.
C
C  THESE VARIABLES MUST BE DEFINED UPON ENTRANCE TO THIS SUBROUTINE:
C           NTWIXT,N1,N2,NS,D,NX,NN.
C  THESE VARIABLES ARE DEFINED OR REDEFINED BY THIS SUBROUTINE:
C           D,NX.
C
      IMPLICIT INTEGER*2 (I-N)
      COMMON /FREE/ F(64),G(64),MA(64),MB(64),MC(64),
     1              A,B,C,X,Y,Z,LA,LB,LC,LD,LE,LF,LU,LV,LW,LX,LY,LZ
      COMMON /IHUSTF/ NTWIXT(26,26),N1(26),N2(26),NS
      COMMON /STRTSF/ D(64,64),NX(64,64),NN,NB,NF
      LOGICAL STEP3
C
C  STEP # 1 OF THE IHU ALGORITHM:
  700 NB=N1(1)
      NF=N2(1)
      IF (NF.GT.NB) CALL DIJKST
C
C  STEPS # 2 AND 3 OF THE IHU ALGORITHM (STEP3=.TRUE. IMPLIES THAT THE
C    ALGORITHM IS IN STEP # 3, OTHERWISE STEP # 2):
      STEP3=.FALSE.
      IF (NS.LT.2) GO TO 744
  702 DO 728 M=2,NS
      I=M
      IF (STEP3) I=KV-M
  704 LB=NB
      LF=NF
      NB=N1(I)
      NF=N2(I)
      IF (LB.GE.LF) GO TO 715
      DO 714 J=LB,LF
      DO 714 K=NB,NF
```

```
      A=1.E70
      Z=1.E70
      DO 712 L=LB,LF
      B=D(J,L)+D(L,K)
      IF (B.GE.A) GO TO 708
      A=B
      LA=L
708   B=D(K,L)+D(L,J)
      IF (B.GE.Z) GO TO 712
      Z=B
      IZ=L
712   CONTINUE
      D(J,K)=A
      IF (LA.NE.J) NX(J,K)=NX(J,LA)
      D(K,J)=Z
714   NX(K,J)=NX(K,LZ)
715   IF (NB.GE.NF) GO TO 728
      LD=NB+1
      LE=NB
      DO 726 J=LD,NF
      DO 724 K=NB,LE
      A=D(J,K)
      LA=K
      Z=D(K,J)
      LZ=J
      DO 722 L=LB,LF
      B=D(J,L)+D(L,K)
      IF (B.GE.A) GO TO 718
      A=B
      LA=L
718   B=D(K,L)+D(L,J)
      IF (B.GE.Z) GO TO 722
      Z=B
      LZ=L
722   CONTINUE
      D(J,K)=A
```

```
      D(K,J)=Z
      NX(J,K)=NX(J,LA)
  724 NX(K,J)=NX(K,LZ)
  726 IE=J
      IF (STEP3) GO TO 728
      CALL DIJKST
  728 CONTINUE
      IF (STEP3) GO TO 732
      KV=NS+1
      STEP3=.TRUE.
      GO TO 702
C
C  STEP #4 OF THE IHU ALGORITHM:
  732 IF (NS.LT.3) GO TO 744
      LV=NS-1
      DO 742 LW=2,LV
      LU=NS-LW
      DO 742 J=1,IU
      I=J+LW
      NB=N1(I)
      NF=N2(I)
      LC=N1(J)
      LD=N2(J)
      K=NTWIXT(J,I)
      LE=N1(K)
      LF=N2(K)
      DO 742 K=NB,NF
      DO 742 L=LC,LD
      A=1.E70
      Z=1.E70
      DO 740 M=IE,IF
      B=D(K,M)+D(M,L)
      IF (B.GE.A) GO TO 736
      A=B
      LA=M
  736 B=D(L,M)+D(M,K)
```

```
      IF (B.GE.Z) GO TO 740
      Z=B
      LZ=M
  740 CONTINUE
      D(K,L)=A
      D(I,K)=Z
      NX(K,L)=NX(K,LA)
  742 NX(I,K)=NX(I,LZ)
  744 RETURN
      END
```

```
      SUBROUTINE DECNXN
C
C
C  THIS SUBROUTINE PERFORMS THE DECOMPOSITION FOR THE NXN SHORTEST ROUTE ALGRTHM
C
C  THESE VARIABLES MUST BE DEFINED UPON ENTRANCE TO THIS SUBROUTINE:
C          LNK$DS,INK$OR,INKLST,TITLE,MNXN,MAXCON,NFORBD.
C  THESE VARIABLES ARE DEFINED OR REDEFINED BY THIS SUBROUTINE:
C          NC,NO,ND,NI.
C
      IMPLICIT INTEGER*2 (I-N)
      COMMON /FREE/ F(64),G(64),MA(64),MB(64),MC(64),
     1              A,B,C,X,Y,Z,LA,LB,LC,LD,LE,LF,LU,LV,LW,LX,LY,LZ
      COMMON /STRTSF/ D(64,64),NX(64,64),NN,NB,NF
      COMMON /MAPSTF/ LNK$OR(400),LNK$DS(400),LNKLST(64),
     1                NUMNEW(64),NUMOLD(64),NA
      COMMON /CNTRSF/ TITLE,MIHU,MNXN,NFORBD,MAXNS,MAXCON
      COMMON /NXNSTF/ NC(1024),NO(64),ND(64),NI(64)
      EQUIVALENCE (D(1,1),E(1,1))
      LOGICAL E(64,64),FORBID
      REAL*8 TITLE(3)
C
C
C  SET UP E, THE LOGICAL INCIDENCE MATRIX.
      DO 3208 I=1,NN
      DO 3208 J=1,NN
 3208 E(I,J)=.FALSE.
      DO 3216 I=1,NA
      LA=LNK$OR(I)
      LB=INK$DS(I)
 3216 E(IA,LB)=.TRUE.
C
C  ESTABLISH THE CURRENT CARDINALITY OF THE FIRST CONNECTION SET
C    CONDITIONAL ON NODE I BEING LABELED # 1, AND PLACE IN MC(I).
      DO 3232 I=1,NN
      LA=0
```

```
      NUMOLD(I)=I
      NUMNEW(I)=I
      DO 3224 J=1,NN
      IF (E(I,J)) LA=LA+1
      IF (E(J,I)) LA=LA+1
 3224 CONTINUE
 3232 MC(I)=LA
      NALLOW=NN-NFORBD
C
C
C  DECOMPOSE THE NETWORK, LABELING NODES IN ORDER TO MINIMIZE CARDINALITY OF
C    NEXT CONNECTION SET.
      INDEX2=0
      NSTOP=NN-2
      DO 3336 I=1,NSTOP
C
C  FIND THE NODE SUCH THAT NEXT CONNECTION SET HAS MINIMUM CARDINALITY.
      MINCRD=32000
      IF (I.GT.NALLOW) NALLOW=NN
      DO 3248 J=I,NALLOW
      LC=MC(J)
      IF (LC.GE.MINCRD) GO TO 3248
      LA=J
      MINCRD=LC
 3248 CONTINUE
C
C  FIND THE OUT, DUPLEX & IN NODES AND STORE THEM IN NC, MA & MB RESPECTIVELY.
      LX=0
      LY=0
      INDEX1=INDEX2+1
      DO 3280 J=I,NN
      IF (E(J,LA)) GO TO 3264
      IF (.NOT.E(LA,J)) GO TO 3280
      INDEX2=INDEX2+1
      NC(INDEX2)=J
      MC(J)=MC(J)-1
```

```
      GO TO 3280
 3264 IF(E(LA,J)) GO TO 3272
      LY=LY+1
      MB(LY)=J
      MC(J)=MC(J)-1
      GO TO 3280
 3272 LX=LX+1
      MA(LX)=J
      MC(J)=MC(J)-2
 3280 CONTINUE
      NO(I)=INDEX2-INDEX1+1
      ND(I)=NO(I)+LX
      NI(I)=ND(I)+LY
C
C  NOW PLACE THE ENTIRE CONNECTION SET IN NC.
      LU=INDEX2+1
      IF (LX.LE.0) GO TO 3292
      DO 3288 J=1,IX
      INDEX2=INDEX2+1
 3288 NC(INDEX2)=MA(J)
 3292 LV=INDEX2
      IF (LY.LE.0) GO TO 3300
      DO 3296 J=1,LY
      INDEX2=INDEX2+1
 3296 NC(INDEX2)=MB(J)
 3300 IF (INDEX2.GT.MAXCON) GO TO 3438
C
C  NOW CHANGES IN CONNECTIONS DUE TO CHOICE OF LA AS NEXT LABELED NODE.
      IF (LU.GT.INDEX2.OR.INDEX1.GT.LV) GO TO 3312
      DO 3304 J=LU,INDEX2
      LB=NC(J)
      DO 3304 K=INDEX1,LV
      LC=NC(K)
      IF (E(LB,LC).OR.LC.EQ.LB) GO TO 3304
      MC(LB)=MC(LB)+1
      MC(LC)=MC(LC)+1
```

```
      E(LB,LC)=.TRUE.
 3304 CONTINUE
C
C  ESTABLISH THE ENTRIES IN NC AS THE OLD NODE NUMBERS RATHER THAN AS THE
C     STILL CHANGING NEW NODE NUMBERS.
 3312 DO 3320 J=INDEX1,INDEX2
 3320 NC(J)=NUMOLD(NC(J))
C
C  TRANSFER THE IDENTITY OF THE ITH NODE TO THAT OF THE LATH NODE.
      IF (LA.LE.I) GO TO 3336
      DO 3328 J=I,NN
      E(LA,J)=E(I,J)
 3328 E(J,LA)=E(J,I)
      MC(LA)=MC(I)
      MC(I)=MINCRL
      LC=NUMOLD(I)
      LD=NUMCLD(LA)
      NUMOLD(I)=LD
      NUMOLD(LA)=LC
      NUMNEW(LC)=LA
      NUMNEW(LD)=I
 3336 E(LA,LA)=.FALSE.
C
C  UPDATE NC IN TERMS OF NEW NODE NUMBERS.
      LB=NN-1
      DO 3352 J=LB,NN
      NO(J)=0
      ND(J)=0
 3352 NI(J)=0
      DO 3360 I=1,INDEX2
 3360 NC(I)=NUMNEW(NC(I))
C
C
C NOW THE DECOMPOSITION INFORMATION IS PRINTED OUT AND PUNCHED OUT.
      PRINT 3376
 3376 FORMAT (' NODE CONVERSION DATA FOR NXN DECOMPOSITION:')
```

```
      LB=0
 3378 LA=LB+1
      LB=LB+25
 3380 IF (NN.LT.LB) LB=NN
      PRINT 3384, (I,I=LA,LB)
 3384 FORMAT ('ONEW NODE NUMBER',5X,25I4)
      PRINT 3388,(NUMOLD(I),I=LA,LB)
 3388 FORMAT (' OLD NODE NUMBER',5X,25I4)
      IF (LB.LT.NN) GO TO 3378
      FORBID=.FALSE.
      IF (MNXN.LE.1) FORBID=.TRUE.
      LA=2
      IF (.NOT.FORBID) PUNCH 3408,TITLE(1),TITLE(2),TITLE(3),NN,LA
 3408 FORMAT (3A8,3I3)
      LA=0
      DO 3416 I=1,NN
      LF=NUMOLD(I)
      LB=1
      IF (LF.GT.1) LB=LNKLST(LF-1)+1
      LC=LNKLST(LF)
      LD=LC-LB+1
      LX=NO(I)+1
      LY=ND(I)
      LZ=NI(I)
      LU=LA+1
      LV=LA+LZ
      IF (FORBID) GO TO 3414
      IF (LV.GE.LU) PUNCH 3412,I,LD,LX,LY,LZ,
     1     (NUMNEW(LNK$DS(J)),J=LB,LC),(NC(J),J=LU,LV)
      IF (LV.LT.LU) PUNCH 3412,I,LD,LX,LY,LZ,
     1     (NUMNEW(LNK$DS(J)),J=LB,LC)
 3412 FORMAT (26I3)
 3414 NO(I)=LA+LX
      ND(I)=LA+LY
      NI(I)=LV
      LA=LV
```

```
      MA(I)=IY
 3416 MB(I)=LZ-LX+1
 3430 RETURN
 3438 PRINT 3446
 3446 FORMAT (' TOO MANY CONNECTIONS IN DECNXN')
      MNXN=-1
      GO TO 3430
      END
```

```
      SUBROUTINE NXN
C
C   THIS IS THE NXN ALGORITHM FOR FINDING ALL THE SHORTEST ROUTES IN A
C     DIRECTED GRAPH.  STEP NUMBERS REFER TO THOSE IN 'SHORTEST ROUTE
C     ALGORITHMS FOR SPARSELY CONNECTED NETWORKS' BY J.E. DEFENDERFER.
C
C   THESE VARIABLES MUST BE DEFINED UPON ENTRANCE TO THIS SUBROUTINE:
C           NC,NO,ND,NI,D,NX,NN.
C   THESE VARIABLES ARE DEFINED OR REDEFINED BY THIS SUBROUTINE:
C           D,NX.
C
      IMPLICIT INTEGER*2 (I-N)
      COMMON /FREE/ F(64),G(64),MA(64),MB(64),MC(64),
     1             A,B,C,X,Y,Z,LA,LB,LC,LD,LE,LF,LU,LV,LW,LX,LY,LZ
      COMMON /STRTSF/ D(64,64),NX(64,64),NN,NB,NF
      COMMON /NXNSTF/ NC(1024),NO(64),ND(64),NI(64)
C
C   STEP # 1 OF NXN ALGORITHM:
      IF (NN.LE.2) GO TO 872
      NT=NN-2
      LC=1
      DO 816 I=1,NT
      MC(I)=LC
      LU=NO(I)
      LV=ND(I)
      LW=NI(I)
      IF (LU.LE.LC) GO TO 804
      LD=LU-1
      DO 802 J=LC,LD
      IA=NC(J)
      C=D(I,LA)
      DO 802 K=IU,IW
      LB=NC(K)
      A=D(LB,I)+C
      IF (A.GE.D(LB,LA)) GO TO 802
      D(LB,LA)=A
```

```
      NX(LB,LA)=NX(LB,I)
  802 CONTINUE
  804 IF(LW.LE.LV.OR.LV.LT.LU) GO TO 808
      LD=IV+1
      DO 806 J=LD,LW
      LA=NC(J)
      C=D(LA,I)
      DO 806 K=LU,IV
      LB=NC(K)
      A=C+D(I,LB)
      IF (A.GE.D(LA,LB)) GO TO 806
      D(LA,LB)=A
      NX(LA,LB)=NX(LA,I)
  806 CONTINUE
  808 IF (LV.LE.LU) GO TO 816
      LF=LU+1
      LE=LU
      DO 814 J=LF,LV
      LA=NC(J)
      B=D(I,LA)
      C=D(LA,I)
      DO 812 K=LU,LE
      LB=NC(K)
      A=D(LB,I)+B
      IF (A.GE.D(LB,LA)) GO TO 810
      D(LB,LA)=A
      NX(LB,LA)=NX(LB,I)
  810 A=C+D(I,LB)
      IF (A.GE.D(LA,LB)) GO TO 812
      D(LA,LB)=A
      NX(LA,LB)=NX(LA,I)
  812 CONTINUE
  814 LE=J
  816 IC=IW+1
C
C  STEP # 2 OF NXN ALGORITHM:
```

```
      NB=NT+1
      LF=NB
      DO 864 L=1,NT
      I=NB-L
      IC=MC(I)
      LU=NO(I)
      LV=ND(I)
      LW=NI(I)
      DO 832 J=IC,IW
      IA=NC(J)
      F(IA)=D(I,IA)
  832 G(LA)=D(LA,I)
      DO 856 J=LF,NN
      Z=1.E70
      B=1.E70
      IF (LU.LE.LC) GO TO 836
      LD=LU-1
      DO 834 K=LC,LD
      LA=NC(K)
      A=F(LA)+D(LA,J)
      IF (A.GE.B) GO TO 834
      B=A
      LB=IA
  834 CONTINUE
  836 IF (LV.LT.IU) GO TO 842
      DO 840 K=LU,LV
      LA=NC(K)
      A=F(LA)+D(LA,J)
      IF (A.GE.B) GO TO 838
      B=A
      LB=IA
  838 Y=D(J,LA)+G(LA)
      IF (Y.GE.Z) GO TO 840
      Z=Y
      LZ=LA
  840 CONTINUE
```

```
842 NX(I,J)=NX(I,LB)
    D(I,J)=B
    IF (LW.LE.LV) GO TO 852
    ID=IV+1
    DO 848 K=LD,LW
    LA=NC(K)
    Y=D(J,LA)+G(LA)
    IF (Y.GE.Z) GO TO 848
    Z=Y
    LZ=LA
848 CONTINUE
852 IF (LZ.NE.J) NX(J,I)=NX(J,LZ)
    D(J,I)=Z
856 CONTINUE
864 LF=I
872 RETURN
    END
```

BIBLIOGRAPHY

1. G. Dantzig, All Shortest Routes in a Graph, Operations Research House, Stanford University, Technical Report 66-3, Nov. 1966.

2. E. Dijkstra, "A note on two problems in connexion with graphs", Numerische Mathematik, Vol. 1, pp. 269-271, 1959.

3. S. Dreyfus, "An appraisal of some shortest-path algorithms", Opns. Res., Vol. 17, pp. 395-411, 1969.

4. R. Floyd, "Algorithm 97, shortest path", Comm. ACM, Vol. 5, pp. 345, 1962.

5. T. C. Hu, "A decomposition algorithm for shortest paths in a network", Opns. Res., Vol. 19, No. 3, pp. 983-983, 1971.

6. T. C. Hu, Integer Programming and Network Flows, Addison-Wesley, Reading, Mass., 1969.

7. A. Hoffman and S. Winograd, "Finding all shortest distances in a directed network", IBM J. Res. Develop., Vol. 16, pp. 412-414, July 1972.

8. NAC 4th Semiannual Tech. Report Project "Analysis and optimization of store-and forward computer networks", Def. Doc. Cen., Alexandria, Va., Dec. 1971.

9. A. R. Pierce, "Biblography on algorithms for shortest path, shortest spanning tree, and related circuit routing problems", Networks, Vol. 5, No. 2, April 1975.

10. P. Spira, "A new algorithm for finding all shortest paths in a graph of positive arcs in average time $o(n^2 \log^2 n)$", SIAM J. Comput., Vol. 2, No. 1, March 1973.

11. J. Yen, "On Hu's decomposition algorithm for shortest paths in a network", Opns. Res., Vol. 19, No. 3, pp. 983-985, 1971.

12. J. Yen, "Finding the lengths of all shortest paths in n-node non-negative-distance complete networks using $\frac{1}{2} n^3$ additions and n^3 comparisons", J. Assoc. Comput. Mach., Vol. 19, No. 3, pp. 423-424, July 1972.

Distribution List

Defense Documentation Center Cameron Station Alexandria, Virginia 22314	12 copies
Assistant Chief for Technology Office of Naval Research, Code 200 Arlington, Virginia 22217	1 copy
Office of Naval Research Information Systems Program Code 437 Arlington, Virginia 22217	2 copies
Office of Naval Research Code 1021P Arlington, Virginia 22217	6 copies
Office of Naval Research Branch Office, Boston 495 Summer Street Boston, Massachusetts 02210	1 copy
Office of Naval Research Branch Office, Chicago 536 South Clark Street Chicago, Illinois 60605	1 copy
Office of Naval Research Branch Office, Pasadena 1030 East Green Street Pasadena, California 91106	1 copy
New York Area Office (ONR) 715 Broadway - 5th Floor New York, New York 10003	1 copy
Naval Research Laboratory Technical Information Division, Code 2627 Washington, D.C. 20375	6 copies

Dr. A. L. Slafkosky | 1 copy
Scientific Advisor
Commandant of the Marine Corps (Code RD-1)
Washington, D.C. 20380

Office of Naval Research | 1 copy
Code 455
Arlington, Virginia 22217

Office of Naval Research | 1 copy
Code 458
Arlington, Virginia 22217

Naval Electronics Laboratory Center | 1 copy
Advanced Software Technology Division
Code 5200
San Diego, California 92152

Mr. E. H. Gleissner | 1 copy
Naval Ship Research & Development Center
Computation and Mathematics Department
Bethesda, Maryland 20084

Captain Grace M. Hopper | 1 copy
NAICOM/MIS Planning Branch (OP-916D)
Office of Chief of Naval Operations
Washington, D.C. 20350

Mr. Kin B. Thompson | 1 copy
Technical Director
Information Systems Division (OP-91T)
Office of Chief of Naval Operations
Washington, D.C. 20350

Advanced Research Projects Agency | 1 copy
Information Processing Techniques
1400 Wilson Boulevard
Arlington, Virginia 22209