SAND77-1165
Unlimited Release

KSPTH: A Subroutine for the K Shortest Paths in a Sabotage Graph

Bernie L. Hulme, Diane B. Holdridge

Prepared by Sandia Laboratories, Albuquerque, New Mexico 87115
and Livermore, California 94550 for the United States Nuclear
Regulatory Commission under ERDA Contract AT(29-1)-789.

Printed August 1977

Sandia Laboratories

Nuclear Fuel Cycle Programs

SF 2900 Q(7-73)

DISCLAIMER

SAND 77-1165
Unlimited Release
Printed August 1977

KSPTH: A SUBROUTINE FOR THE K SHORTEST PATHS IN A SABOTAGE GRAPH

Bernie L. Hulme
Numerical Mathematics Division

Diane B. Holdridge†
Applied Mathematics Division

Sandia Laboratories
Albuquerque, New Mexico 87115

ABSTRACT

Finding shortest paths in a weighted graph model is one way for a safeguards analyst to locate weaknesses in a facility's barrier and alarm system. KSPTH can be used to rank sabotage paths according to path length so that the K shortest paths can be studied by more detailed methods to see which possess additional properties attractive to an adversary. While emphasizing how to use KSPTH, this report explains the K-th shortest path algorithm of Hoffman, Pavley and Dreyfus and contains a sample problem, test results and program listings.

†Currently in Nuclear Waste Technology Division.

Printed in the United States of America

Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, Virginia 22161

Price: $4.00 Microfiche $3.00

TABLE OF CONTENTS

1. Introduction

A sabotage graph is a network of nodes and arcs schematically representing a fixed-site facility. By weighting the nodes and arcs with some quantity to be minimized by an adversary (say, time or detection probability), a safeguards analyst can find shortest paths in the graph as a means of identifying possibly good attack routes for saboteurs. Such paths indicate where the barrier and alarm systems are weakest. A similar, but more difficult, problem is that of finding shortest theft paths, which must include escape routes. Graph-theoretic models for the theft problem were introduced in [6] and for the sabotage problem in [7].

The current shortest sabotage path code is SPTH3 [8]. This subroutine finds all the shortest paths from off-site to every node of the graph using the Dijkstra shortest path algorithm [3] as modified by Yen [10,9]. Those paths terminating at target nodes represent shortest routes for a simultaneous attack by several sabotage teams each having only one target.

Of course, near-shortest adversary paths are also important to the safeguards analyst. One reason is that the data are not known well enough to distinguish among paths of almost the same length. In addition, other factors such as the reliability of the communications systems and the ability of response forces to engage and defeat, or at least delay, the adversary also play an important role in finding "best" adversary paths. The idea behind the use of KSPTH is to rank sabotage paths according to length so that the K shortest can be further evaluated by techniques such a as FESEM [2], the Forcible Entry Safeguard Effectiveness Model, or EASI [1], an Estimate of Adversary Sequence Interruption.

KSPTH begins by finding the shortest sabotage paths just as SPTH3 does. Then the Hoffman-Pavley-Dreyfus algorithm [4,5] is used to find the k-th

shortest paths from the boundary to all the nodes inside the facility, $2 \leq k \leq K$. The sabotage paths are those that lead to target nodes.

Although KSPTH is written to accept time weights (or any other weights for which normal addition is appropriate), a minor modification will allow it to accept detection probability weights and compute cumulative detective probabilities as path lengths. In the remainder of this report we think of the weights as time. Also we assume that the reader is familiar with [8].

2. The Hoffman-Pavley-Dreyfus Algorithm

In 1959 Hoffman and Pavley [5] gave a procedure for finding the k-th shortest paths between two specified nodes in a graph. In 1969 Dreyfus [4] analysed their scheme, compared it with others, and proposed an extention to the problem of k-th shortest paths from all nodes to a common terminal node. In KSPTH we use the Dreyfus modification of Hoffman and Pavley's method to find k-th shortest paths from the set of boundary nodes to all other nodes in a sabotage graph. This is akin to finding paths from a common source node to all the other nodes in a graph. However, instead of one source node, we have a set of boundary nodes which may be thought to be adjacent to a ficticious "off-site node." Consequently, the discussion here will differ from that of Dreyfus in that the common node is taken to be a source rather than a terminal.

To begin it must be noted that different paths can have the same length. Therefore, we must think in terms of a set of shortest paths, a set of second-shortest paths, etc. The problem is to find the K sets of shortest paths from a source node s to each of the other nodes in a graph G.

Let $x_{j,k}$ be the length of the k-th shortest paths from s to j, let $d_{i,j} \geq 0$ be the direct distance from node i to node j ($d_{i,i} = 0$ and $d_{i,j} = \infty$ if $(i,j) \notin G$), and let $n_{i,j,k}$ be the number of different-length paths among the k shortest from s to j which terminate with the arc (i,j). Assume that the shortest paths from s to all other nodes j have already been found and that the $x_{j,1}$ have been set to the corresponding shortest path lengths and the $n_{i,j,1}$ have been set to 1, if (i,j) belongs to a shortest path, otherwise to 0. Then, for k = 2,3,...,K, let j be the node closest to s (along a shortest path), set

$$x_{j,k} = \min_{i \neq j} \left(x_{i,n_{i,j,k-1}+1} + d_{i,j} \right), \tag{1}$$

set $n_{i,j,k} = n_{i,j,k-1} + 1$ for each minimizing i and $n_{i,j,k} = n_{i,j,k-1}$ for all other i, let j be the next closest node to s, and repeat the procedure until all nodes j have received their k-th labels, $x_{j,k}$.

To partially see why this procedure works, it is necessary to understand the role of the quantity $n_{i,j,k}$. When k = 2, every node i adjacent to j is examined in (1). If $n_{i,j,1} = 0$, then $x_{i,1} + d_{i,j}$ is a candidate for $x_{j,2}$ because (i,j) does not belong to a shortest s - j path and hence a shortest s - i path followed by arc (i,j) is a possibly second-shortest s - j path. If $n_{i,j,1} = 1$, however, then (i,j) belongs to a shortest s - j path and in this case $x_{i,2} + d_{i,j}$, the length of a second-shortest s - i path followed by (i,j), is a candidate for second-shortest s - j path. The $x_{i,1}$ are determined before starting the k = 2 stage. Also, any $x_{i,2}$ appearing on the right side of (1) is available when it is needed because i is necessarily closer to s than j and therefore i receives its second label before j does. Similar remarks apply to the k-th stage.

The quantity $n_{i,j,k-1}$ in (1) may be thought of as the number of labels

on node i that have already been used in determining the first k - 1 labels on j. Thus, the $(n_{i,j,k-1}+1)$-st label on i must be used in a candidate for the k-th label on j. By setting $n_{i,j,k} = n_{i,j,k-1} + 1$ for a minimizing i, the algorithm records the fact that one more label on i has been used in finding a label on j. For example, suppose that at the end of the fifth stage $n_{i,j,1} = 0$, $n_{i,j,2} = 1$, $n_{i,j,3} = 1$, $n_{i,j,4} = 1$, and $n_{i,j,5} = 2$. This means that (i,j) does not belong to a shortest s - j path, a second-shortest s - j path begins with a shortest s - i path and ends with arc (i,j), neither a third nor a fourth-shortest s - j path contains (i,j), and a fifth-shortest s - j path begins with a second-shortest s - i path and ends with (i,j).

3. Description of KSPTH

Given an N-node sabotage graph [8, Sec. 3] and a value for K, KSPTH finds all the k-th shortest paths from the boundary to each target node, $1 \le k \le K$. In the process it finds k-th shortest paths to all the barrier nodes as well. The Dijkstra-Yen algorithm [8, Sec. 6.5] is used to obtain the shortest paths, and then the Hoffman-Pavley-Dreyfus method just described is used to produce the second through K-th shortest paths. In both methods the boundary nodes may be viewed as being connected by zero-length arcs to a single, fictitious, source node.

KSPTH does the same calculations as SPTH3 to find the shortest paths. In the same manner as explained in Section 6 of [8], KSPTH tests for triangle inequalities on arc weights, computes direct distances between node centers (using the full node weight for distances to and from boundary and target node centers since they are endpoints of sabotage paths), and stores and retrieves these distances from the arc weight input vector

rather than from an extra matrix.

KSPTH differs from SPTH3 in two ways during the shortest path computation. Addressing of a distance $d_{i,j}$, stored in the arc weight vector, is done by a function subroutine IAD, which the user never calls. Also KSPTH does not retrace and count the shortest paths. Instead, KSPTH stores the predecessors $p_{j,k}$ of nodes j along the k-th shortest paths for k = 1,2,...,K. These predecessors together with the indices $n_{i,j,k}$ described in Section 2 contain all the information needed for the subroutine LKSP to list the K shortest paths as node sequences.

When KSPTH returns, having produced in common working storage the path lengths $x_{j,k}$, the predecessors $p_{j,k}$, and the indices $n_{i,j,k}$, the user may then call LKSP. This subroutine prints the individual node sequences and lengths for each of the k-th shortest paths to each target, $1 \le k \le K$. A k-th shortest path from the boundary to target t is found by forming the backward node sequence

$$t,u,v,w,\ldots,z \ ,$$

where

$$u = p_{t,k} \ ,$$

$$v = p_{u,n_{u,t,k}} \ ,$$

$$w = p_{v,n_{v,u,n_{u,t,k}}} \ ,$$

$$\vdots$$

$$z = \text{a boundary node} \ .$$

Users interested only in shortest paths may still want to use SPTH3. It has slightly less overhead than KSPTH with K = 1, and it represents the set of shortest paths to all targets as an arc list rather than as individual node sequences.

4. How to Use KSPTH

4.1. The Call List

The call list for KSPTH is

KSPTH(N1,N2,N3,NA,W,MR,II,JJ,AWT,KBAR,IFLAG) ,

where the arguments have the following meanings:

N1 - the number of target nodes,

N2 - the number of barrier nodes,

N3 - the number of boundary nodes,

NA - the number of arcs,

W(·) - the node weight vector, dimensioned N = N1 + N2 + N3,

(the next four vectors, dimensioned NA, give the arcs as quadruples consisting of a region, two nodes, and an arc weight)

MR(·) - the region index vector,

II(·) - a node index vector,

JJ(·) - a node index vector,

AWT(·) - the arc weight vector,

KBAR - the upper limit K on k for k-th shortest paths,

IFLAG - 1, for normal return,
0, for return with no output. See the printed message. Either there is a triangle inequality failure, or some node is isolated from the boundary, or else two nodes which were adjacent no longer appear to be adjacent. The third difficulty should never arise. It implies that the array IREG has been inadvertently overwritten, perhaps because of exceeding another array dimension.

4.2. Input

First, construct a weighted sabotage graph as indicated in [8, Sec. 3] and decide on a value of K, the number of different-length paths to be found to each target. In the program which calls KSPTH dimension W by N and MR,II,JJ,AWT by NA. Then store the node weights in W and the arc data in MR,II,JJ and AWT.

The ordering of node and arc data is the same as for SPTH3 [8]. W(I)

is the weight of node I, $1 \le I \le N$. All the arcs of one region are listed consecutively in MR,II,JJ,AWT, and the regions may be given in any order. Within each region, however, the arcs must be listed as if they were taken row by row from the strictly upper triangular part of some node adjacency matrix. For example, if the nodes of one region are {16,9,21,4,7}, then an acceptable arc ordering based on the given node ordering is (16,9), (16,21), (16,4), (16,7), (9,21), (9,4), (9,7), (21,4), (21,7), (4,7). Notice that the arc ordering for a region may be based on any ordering of the region's nodes. This special arc ordering allows KSPTH to quickly address any arc weight in AWT and hence completely eliminates the need for the usual N x N direct distance matrix. The resulting storage economy is very significant for several hundred nodes.

4.3. Work Arrays

The user must set the dimensions of several work arrays before running a job. The following arrays (in unlabeled common storage) must be dimensioned in the subroutines KSPTH, IAD, and LKSP:

IPERM,ITEMP,IPR - (N),

NODE - (N,4),

IREG - (NR,2), where NR = the number of regions,

X,IMP - (N,K),

NP - (2,NA,K),

ICSV,LSV,JSV,I2SV,IR3SV - (5), an unpredictable maximum number of branch points for alternate equal-length paths along a path being traced for output by LKSP. This value has been adequate for all our test problems.

NPOOL - (1000), an initially unpredictable total number of extra predecessors for nodes which have more than one predecessor along k-th shortest paths, $1 \le k \le K$. KSPTH prints a message when this dimension needs to be increased. In this case, the results should be viewed as incomplete, and the problem should be rerun with a larger dimension whose value is printed just before the return from KSPTH. With this dimension NPOOL is guaranteed to be exactly large enough because KSPTH finished solving the problem to see how large NPOOL should be.

4.4. Output

The only output variable in KSPTH's call list is IFLAG which must be tested to see if a normal execution took place. The settings of IFLAG are explained in Section 4.1. When IFLAG = 1, the K shortest sabotage paths and their lengths are stored in the work arrays X,IMP,NP and NPOOL. By calling the subroutine LKSP the user obtains a list of these paths and path lengths.

The call list for LKSP is

LKSP(N1,KBAR,MR,LAG) ,

where N1, KBAR and MR have the same meanings as for KSPTH. LAG is an input flag which gives the user two options:

LAG = 0, print all of the K shortest sabotage paths,
= 1, print only those K shortest sabotage paths which have no two adjacent arcs in the same region.

Often the user is uninterested in paths having two adjacent arcs in the same region because such a path differs only slightly from another path of the same or shorter length, namely the path obtained by deleting these two arcs and inserting the third side of the triangle. For each target node t = 1,2,...,N1, LKSP prints the length and the backward node sequence for each of the K shortest sabotage paths, in order of increasing length. When LAG = 0 and some equal-length paths to the same target coincide near the end, the second and subsequent paths will have the coinciding nodes suppressed. For example, the three paths (152,34,31,5), (153,34,31,5) and (154,16,31,5) of length 975.3 would be printed as

975.3	5	31	34	152
975.3				153
975.3			16	154 .

In addition to these outputs, KSPTH prints the number of entries of NPOOL which were used. In the event that the dimension of NPOOL was exceeded and a corresponding message was printed, this number can be used to dimension NPOOL before rerunning the problem.

Like SPTH3, KSPTH changes W and AWT by

$$W(I) \leftarrow W(I)/2. \ , \quad N1 + 1 \le I \le N1 + N2 \ ,$$

$$AWT(K) \leftarrow AWT(K) + W\left(II(K)\right) + W\left(JJ(K)\right) \ , \quad 1 \le K \le NA.$$

In this way the direct distances between node centers are stored in AWT rather than in an N x N matrix.

5. Examples

5.1. A Sample Problem

Suppose that in the sabotage graph of Figure 1, in which the squares are boundary nodes, the circles are barrier nodes, and the shaded circles are target nodes, we seek the five shortest paths from the boundary to each target.

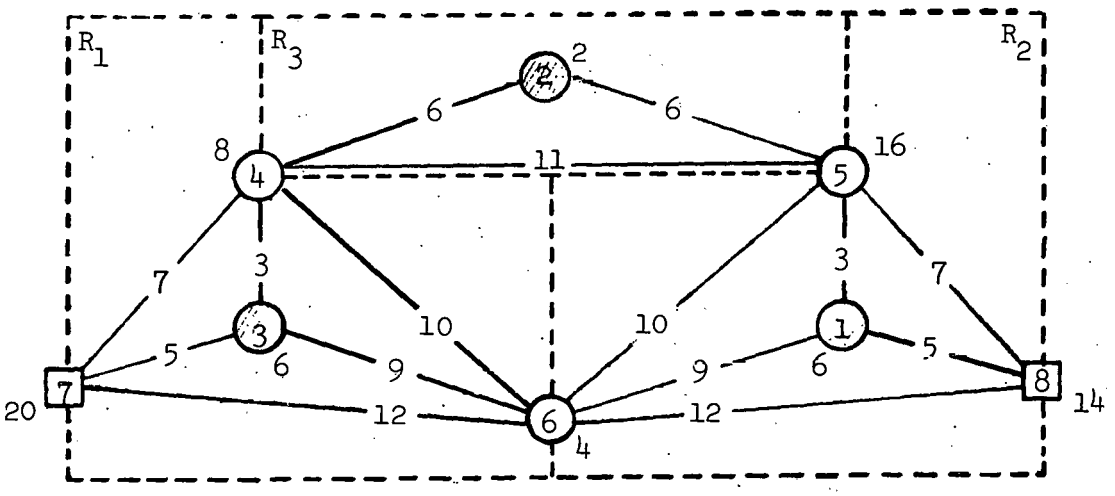

Figure 1 .
A Weighted Sabotage Graph

The input consists of

$$N1 = 3,\ N2 = 3,\ N3 = 2,\ NA = 15,\ KBAR = 5$$

$$W = \{6.,2.,6.,8.,16.,4.,20.,14.\}\ ,$$

MR	II	JJ	AWT
1	3	4	3.
1	3	6	9.
1	3	7	5.
1	4	6	10.
1	4	7	7.
1	6	7	12.
2	1	5	3.
2	1	6	9.
2	1	8	5.
2	5	6	10.
2	5	8	7.
2	6	8	12.
3	2	4	6.
3	2	5	6.
3	4	5	11.

KSPTH returns with IFLAG = 1. A call of LKSP with LAG = 0 yields all five of the shortest paths to each target:

```
25.    1 8
45.    1 6 8
46.    1 5 8
51.    1 6 7
59.    1 5 1 8
59.      6 1 8
43.    2 4 7
45.    2 5 8
56.    2 4 3 7
56.        6 8
58.    2 5 1 8
62.    2 4 6 7
31.    3 7
44.    3 4 7
45.    3 6 8
51.    3 6 7
57.    3 4 3 7
57.        6 8 .
```

Notice that there are two fifth-shortest paths to targets 1 and 3 and two

third-shortest paths to target 2. Also notice that path (8,6,1) is mathematically the second shortest path to target 1, yet, because it involves the needless penetration of node 6, it is physically uninteresting as a sabotage path. In order to automatically delete from LKSP's output any paths containing two consecutive arcs in the same region, the user may call LKSP with LAG = 1. This yields

```
25.    1 8
51.    1 6 7
43.    2 4 7
45.    2 5 8
56.    2 4 6 8
31.    3 7
45.    3 6 8 ,
```

the subset of potentially interesting sabotage paths.

5.2 Run Time and Array Storage Results

KSPTH is both fast and storage efficient. The Dijkstra-Yen shortest path algorithm has a run time of $O(N^2)$, and the Hoffman-Pavley-Dreyfus K-th shortest path algorithm is O(NK). The array storage is

$$2(N+NA)K + 8N + 4NA + 2NR + NPL + 25 ,$$

where NPL is the dimension needed for NPOOL. For a given graph, then, both the run time and the array storage exclusive of NPOOL vary linearly with K. Because of the coefficient 2(N+NA), however, the storage increase with K can be rather rapid.

Table I presents the run times for problems with different sized graphs and K = 1,5,10,20, and Table II gives the corresponding array storage requirements for the larger problems. The sample problem above is Problem 2.

Table I

CDC 6600 Run Times for KSPTH

Problem	N1-N2-N3	Nodes N	Arcs NA	Regions NR	Run Times (seconds)			
					K=1	K=5	K=10	K=20
1	1-5-1	7	13	4	0.004	0.014	0.030	0.055
2	3-3-2	8	15	3	0.003	0.014	0.028	0.055
3	5-1-2	8	16	2	0.005	0.013	0.026	0.052
4	2-6-2	10	23	5	0.006	0.022	0.042	0.084
5	4-8-2	14	34	6	0.009	0.032	0.068	0.130
6	1-8-8	17	32	8	0.011	0.035	0.072	0.132
7	1-10-8	19	40	8	0.015	0.043	0.089	0.164
8	1-31-14	46	112	20	0.060	0.167	0.321	0.508
9	20-58-4	82	514	35	0.223	0.534	0.971	1.748
10	30-120-5	155	656	82	0.637	1.061	1.686	2.752

Table II

Array Storage Requirements

Problem	NPL, dimension of NPOOL				Array Storage (decimal)			
	K=1	K=5	K=10	K=20	K=1	K=5	K=10	K=20
7	40	93	222	492	511	1036	1755	3205
8	40	417	945	1886	1237	2878	4986	9087
9	40	191	409	913	4039	8958	15,136	27,560
10	40	121	297	822	5715	12,284	20,570	37,315

```
      SUBROUTINE KSPTH(N1,N2,N3,NA,W,MR,II,JJ,AWT,KBAR,IFLAG)
      DIMENSION W(1),MR(1),II(1),JJ(1),AWT(1)
      COMMON IPERM(155),ITEMP(155),NODE(155,4),IPR(155),ICSV(5),
     1 LSV(5),JSV(5),I2SV(5),IR3SV(5),NPOOL(1000),IREG(82,2),
     2 X(155,20),IMP(155,20),NP(2,656,20)
      COMMON / A / IR,IR2,LJ,LJ2,M,M2,I12
      COMMON / B / NMX(25),LOW,ISHFT,N12
      DATA NMX / 10H001X,25I4,,10H005X,24I4,,10H009X,23I4,,
     1 10H013X,22I4,,10H017X,21I4,,10H021X,20I4,,10H025X,19I4,,
     2 10H029X,18I4,,10H033X,17I4,,10H037X,16I4,,10H041X,15I4,,
     3 10H045X,14I4,,10H049X,13I4,,10H053X,12I4,,10H057X,11I4,,
     4 10H061X,10I4,,10H065X,09I4,,10H069X,08I4,,10H073X,07I4,,
     5 10H077X,06I4,,10H081X,05I4,,10H085X,04I4,,10H089X,03I4,,
     6 10H093X,02I4,,10H097X,01I4, /
C    K-TH SHORTEST PATHS, 1 LE. K LE. KBAR
C    SIMULTANEOUS SABOTAGE--ONE TEAM PER HARDWARE NODE.
C    GIVEN UNDIRECTED NODES AND ARCS WITH TIME WEIGHTS, KSPTH FINDS THE
C    SHORTEST PATHS FROM OFF-SITE TO ALL NODES IN THE SABOTAGE GRAPHS
C    USING AN INWARD DIJKSTRA-YEN SEARCH.  THEN THE HOFFMAN-PAVLEY-
C    DREYFUS ALGORITHM IS USED TO OBTAIN K-TH SHORTEST PATHS TO ALL
C    NODES FOR K=2,3,...,KBAR.  WHEN KBAR=1, ONLY SHORTEST PATHS ARE
C    OBTAINED.
C    INPUT.
C      N1   NO. OF HARDWARE NODES.
C      N2   NO. OF BARRIER NODES.
C      N3   NO. OF BOUNDARY NODES.
C      NA   NO. OF ARCS.
C       W   NODE WEIGHT VECTOR, DIMENSIONED N=N1+N2+N3.
C           W IS CHANGED.
C    ARC DATA--FOUR NA VECTORS.  ARCS MUST BE LISTED REGION BY REGION.
C              FURTHERMORE, WITHIN EACH REGION HAVING P NODES THERE MUST
C              BE P*(P-1)/2 ARCS LISTED ROW BY ROW IN STRICTLY UPPER
C              TRIANGULAR FORM.  THAT IS, (I1,I2), (I1,I3), ..., (I1,IP),
C              (I2,I3), ..., (I2,IP), (I3,I4), ..., (I3,IP), ...,
C              (IPM1,IP).
C      MR   REGION INDEX VECTOR.
C      II   NODE INDEX VECTOR.
C      JJ   NODE INDEX VECTOR.
C     AWT   ARC WEIGHT VECTOR.  AWT IS CHANGED.
C           SUBROUTINE IAD MAPS NODE INDICES I AND J INTO THE ADDRESS IAD
C           OF ARC(I,J) IN THE ARC LIST, THUS AVOIDING THE STORAGE OF AN
C           N BY N DISTANCE MATRIX.
C    KBAR   MAXIMUM VALUE OF K.
C    OUTPUT.
C     IFLAG   = 0, NO OUTPUT, SEE MESSAGE.
C             = 1, NORMAL RETURN.
C             THE FOLLOWING ARRAYS X, IMP, NP AND NPOOL CONTAIN THE
C             ANSWERS.  X HAS THE PATH LENGTHS AND THE OTHER THREE ARRAYS
C             ARE USED TO CONSTRUCT THE PATHS.  SUBROUTINE LKSP WILL
C             PRINT THE PATHS TO EACH HARDWARE NODE AND THEIR LENGTHS.
C          X  DISTANCE LABELS.  THE LABEL X(J,K) IS THE LENGTH OF THE
C             K-TH SHORTEST PATHS FROM OFF-SITE TO NODE J.
C             DIMENSIONED (N=N1+N2+N3,KBAR).
C IMP(J,K)   IS THE IMMEDIATE PREDECESSOR OF NODE J ALONG A K-TH
C             SHORTEST PATH FROM OFF-SITE TO NODE J.
```

```
C            DIMENSIONED (N,KBAR).
C         NP  NUMBER OF PATHS (OF DISTINCTLY DIFFERENT LENGTHS) AMONG TH
C            K SHORTEST PATHS FROM OFF-SITE TO NODE J THAT END WITH
C            ARC(I,J).  DIMENSIONED (2,NA,KBAR).  SUBROUTINE IAD MAPS
C            I AND J INTO THE FIRST TWO SUBSCRIPTS OF NP(I12,IAD,K).
C      NPOOL  A LINKED LIST IN WHICH ADDITIONAL PREDECESSORS MAY BE
C            STORED WHEN NODE J HAS MORE THAN ONE.  THE LINK FROM
C            IMP(J,K) TO NPOOL(LINK) IS STORED IN THE LEFT 51 BITS OF
C            IMP(J,K).  SIMILARLY, IF THERE IS A THIRD PREDECESSOR OF
C            J, THEN LINK1 FROM NPOOL(LINK) TO NPOOL(LINK1) IS STORED
C            IN THE LEFT 51 BITS OF NPOOL(LINK), ETC.  DIMENSIONED 100C
C            IF THE DIMENSION OF NPOOL IS CHANGED, THEN THE THIRD
C            STATEMENT WHICH DEFINES NPLDP MUST BE CHANGED.  NPLDP
C            IS THE NPOOL DIMENSION PLUS ONE.  IF THE DIMENSION N IS
C            INCREASED TO MORE THAN 511 NODES, THE VALUES OF LTEST,
C            LOW AND ISHFT MUST BE CHANGED TO ALLOW MORE THAN 9 BITS IN
C            THE RIGHT OF EACH MASK.
C     WORK ARRAYS
C      IPERM  NODES WHERE DISTANCE LABELS X(J,1) HAVE BEEN MADE PERMANEN
C      ITEMP  NODES WHERE DISTANCE LABELS X(J,1) ARE STILL TEMPORARY.
C       NODE  REGION AND LOCAL NODE NUMBERS FOR EACH NODE.
C            DIMENSION (N,4).
C            NODE(I,1), NODE(I,3) ARE REGION NUMBERS FOR NODE I.
C            NODE(I,2), NODE(I,4) ARE CORRESPONDING LOCAL NODE NUMBERS.
C       IREG  REGION DATA CONCERNING ARCS.  DIMENSIONED (NO. REGIONS, 2)
C            IREG(R,1) IS THE FIRST WORD ADDRESS MINUS ONE IN THE ARC
C            LIST OF THE ARCS OF REGION R.
C            IREG(R,2) IS THE NUMBER OF NODES IN REGION R, IMPLYING
C            THERE ARE IREG(R,2)*(IREG(R,2)-1)/2 ARCS IN REGION R.
      DATA EPS,BIG / 1.0E-13,1.0E321 /
      DATA LTEST,LOW,ISHFT / 1000B,777B,9 /
      DATA NPLDP / 1001 /
      IFLAG=1
      OMEPS=1.0-EPS
      OPEPS=1.0+EPS
      N12=N1+N2
      N=N12+N3
      NM1=N-1
      N1P=N1+1
      N12P=N12+1
C     CHECK EACH REGION FOR TRIANGLE INEQUALITY ON ARC WEIGHTS
      ICT=0
      IREG=1
 5    IREGP=IREG+1
      IF(MR(IREG) .NE. MR(IREGP)) GO TO 52
      LIKE=II(IREG)
      DO 10 I=IREGP,NA
      IF(II(I) .NE. LIKE) GO TO 15
 10   CONTINUE
      GO TO 52
 15   NM=I-IREG
      IF(NM .LE. 1) GO TO 52
 20   I2=IREG+NM
      IEND=I2-2
      NMT=NM-1
```

```
      DO 50 I1=IBEG,IEND
      AWTOM=AWT(I1)*OMEPS
      I3=I1+1
      DO 45 J=1,NMT
      IF(AWT(I2)+AWT(I3) .GE. AWTOM) GO TO 35
 25   FORMAT(* TRIANGLE*3I3* FAILS.  ARC WEIGHTS--*3E15.5)
 30   PRINT 25,II(I1),JJ(I1),JJ(I3),AWT(I1),AWT(I2),AWT(I3)
      ICT=1
      GO TO 40
 35   IF(AWT(I1)+AWT(I2) .LT. AWT(I3)*OMEPS) GO TO 30
      IF(AWT(I1)+AWT(I3) .LT. AWT(I2)*OMEPS) GO TO 30
 40   I2=I2+1
 45   I3=I3+1
 50   NMT=NMT-1
      NM=NM-1
      IBEG=IEND+2
      IF(NM .GE. 2) GO TO 20
 52   IBEG=IBEG+1
      IF(IBEG .LT. NA) GO TO 5
      IF(ICT .EQ. 0) GO TO 55
      IFLAG=0
      RETURN
C   COMBINE NODE WEIGHTS INTO ARC WEIGHTS
 55   DO 65 I=N1P,N12
      W(I)=0.5*W(I)
 65   CONTINUE
      DO 68 IA=1,NA
      I=II(IA)
      J=JJ(IA)
      AWT(IA)=AWT(IA)+W(I)+W(J)
 68   CONTINUE
C    SET THE ARRAYS NODE, IREG.
      DO 70 I=1,N
      NODE(I,1)=0
 70   NODE(I,3)=0
      L=1
 72   K=1
      IR=MR(L)
      IREG(IR,1)=L-1
      I=II(L)
      IF(NODE(I,1) .EQ. 0) GO TO 73
      NODE(I,3)=IR
      NODE(I,4)=K
      GO TO 74
 73   NODE(I,1)=IR
      NODE(I,2)=K
 74   K=K+1
      J=JJ(L)
      IF(NODE(J,1) .EQ. 0) GO TO 75
      NODE(J,3)=IR
      NODE(J,4)=K
      GO TO 76
 75   NODE(J,1)=IR
      NODE(J,2)=K
 76   IF(L .EQ. NA) GO TO 77
```

```
      L=L+1
      IF((I .EQ. II(L)) .AND. (IR .EQ. MR(L))) GO TO 74
      IREG(IR,2)=K
      L=L-K+K*(K-1)/2+1
      IF(L .LE. NA) GO TO 72
 77   IREG(IR,2)=K
C    DIJKSTRA-YEN SEARCH INWARD.
C    INITIALIZE.
      DO 125 I=1,N12
      X(I,1)=BIG
      ITEMP(I)=I
 125  CONTINUE
      DO 127 I=N12P,N
      X(I,1)=0.0
      ITEMP(I)=I
  127 CONTINUE
      IPL=1
      L=1
C    PERMANENTLY LABEL NODE N.
      IPERM(1)=N
      I=N
      IR=NODE(I,1)
      LJ=NODE(I,2)
      M=IREG(IR,1)+(LJ-1)*IREG(IR,2)-LJ*(LJ+1)/2
      IR2=0
      K=NM1
      V=BIG
C    TREAT EACH TEMPORARILY LABELED NODE.
C    V IS THE SMALLEST SUCH LABEL.
  130 DO 140 IT=1,K
      J=ITEMP(IT)
      IAR=IAD(J)
      IF(IAR .EQ. 0) GO TO 135
      DIJ=AWT(IAR)
      Z=X(I,1)+DIJ
      XJPEPS=X(J,1)*OPEPS
      IF(Z .GT. XJPEPS) GO TO 135
      XJMEPS=X(J,1)*OMEPS
      IF(Z .GE. XJMEPS) GO TO 300
      X(J,1)=Z
      IMP(J,1)=I
      GO TO 135
 300  IF(IPL-NPLDP) 305,302,340
 301  FORMAT(* NPOOL NEEDS TO STORE MORE LINKS*)
 302  PRINT 301
      GO TO 340
 305  NPRED=IMP(J,1)
 310  IF(NPRED .LT. LTEST) GO TO 320
      LINK=SHIFT(NPRED, -ISHFT)
      NPRED=NPOOL(LINK)
      GO TO 310
 320  NPOOL(IPL)=I
      I1=SHIFT(IPL, ISHFT) .OR. NPRED
      IF(NPRED .EQ. IMP(J,1)) GO TO 330
      NPOOL(LINK)=I1
```

```
      GO TO 340
  330 IMP(J,1)=I1
  340 IPL=IPL+1
  135 IF(X(J,1) .GE. V) GO TO 140
      V=X(J,1)
      IP=J
      IQ=IT
  140 CONTINUE
      IF(V .NE. BIG) GO TO 155
      IFLAG=0
  145 FORMAT(*0SOME NODE HAS NO PATH FROM THE BOUNDARY.*)
      PRINT 145
      RETURN
C   NODE IP IS TO BE PERMANENTLY LABELED.
  155 V=BIG
      L=L+1
      IPERM(L)=IP
      I=IP
      IR=NODE(I,1)
      LJ=NODE(I,2)
      M=IREG(IR,1)+(LJ-1)*IREG(IR,2)-LJ*(LJ+1)/2
      IR2=NODE(I,3)
      LJ2=NODE(I,4)
      IF(IR2 .NE. 0) M2=IREG(IR2,1)+(LJ2-1)*IREG(IR2,2)-LJ2*(LJ2+1)/2
      ITEMP(IQ)=ITEMP(K)
      K=K-1
      IF(K .GT. 0) GO TO 130
C   ALL NODES ARE PERMANENTLY LABELED.
C   STORE THE SHORTEST PATH DATA.
      DO 225 I=1,2
      DO 225 IA=1,NA
  225 NP(I,IA,1)=0
      DO 235 J=1,N12
      IR=NODE(J,1)
      LJ=NODE(J,2)
      IR2=NODE(J,3)
      LJ2=NODE(J,4)
      M=IREG(IR,1)+(LJ-1)*IREG(IR,2)-LJ*(LJ+1)/2
      IF(IR2 .NE. 0) M2=IREG(IR2,1)+(LJ2-1)*IREG(IR2,2)-LJ2*(LJ2+1)/2
      NJ=IMP(J,1)
  230 I=NJ .AND. LOW
      IAR=IAD(I)
      IF(IAR .NE. 0) GO TO 232
  231 FORMAT(*0IR OR IR2 IS WRONG IN FUNCTION IAD*12I5)
      PRINT 231,IR,IR2,I,J,LJ,LJ2,M,M2,(NODE(I,L),L=1,4)
      IFLAG=0
      RETURN
  232 NP(I12,IAR,1)=1
      IF(I .EQ. NJ) GO TO 235
      LINK=SHIFT(NJ, -ISHFT)
      NJ=NPOOL(LINK)
      GO TO 230
  235 CONTINUE
      IF(KBAR .LT. 2) RETURN
C   K-TH SHORTEST PATHS, K GT. 1.  HOFFMAN-PAVLEY-DREYFUS ALGORITHM.
```

```
      DO 370 K=2,KBAR
C   INITIALIZE FOR K-TH STAGE.
      KM1=K-1
      DO 237 I=1,2
      DO 237 IA=1,NA
  237 NP(I,IA,K)=NP(I,IA,KM1)
      DO 240 J=1,N
  240 X(J,K)=BIG
C   TREAT NODES J IN ORDER OF INCREASING DISTANCE FROM OFF-SITE.
      L=N3+1
  245 J=IPERM(L)
C   STORE ALL NEIGHBORS OF J IN ITEMP.
      IR2=NODE(J,1)
      NN=0
      DO 260 KK=1,2
      IF(IR2 .EQ. 0) GO TO 260
      IBEG=IREG(IR2,1)+1
      IEND=IBEG+IREG(IR2,2)-2
      IF(II(IBEG) .EQ. J) GO TO 250
      NN=NN+1
      ITEMP(NN)=II(IBEG)
  250 DO 255 KKK=IBEG,IEND
      IF(JJ(KKK) .EQ. J) GO TO 255
      NN=NN+1
      ITEMP(NN)=JJ(KKK)
  255 CONTINUE
  260 IR2=NODE(J,3)
      IR=NODE(J,1)
      LJ=NODE(J,2)
      LJ2=NODE(J,4)
      M=IREG(IR,1)+(LJ-1)*IREG(IR,2)-LJ*(LJ+1)/2
      IF(IR2 .NE. 0) M2=IREG(IR2,1)+(LJ2-1)*IREG(IR2,2)-LJ2*(LJ2+1)/2
C   USE EACH NEIGHBOR OF J TO ATTEMPT A REDUCTION OF X(J,K).
C   WHENEVER A REDUCTION OCCURS STORE THE NEIGHBOR I IN IMP(J,K) OR
C   NPOOL.
      DO 355 IT=1,NN
      I=ITEMP(IT)
      IAR=IAD(I)
      IF(IAR .NE. 0) GO TO 262
      PRINT 231,IR,IR2,I,J,LJ,LJ2,M,M2,(NODE(I,L),L=1,4)
      IFLAG=0
      RETURN
  262 I2=NP(I2,IAR,K)+1
      Z=X(I,I2)+AWT(IAR)
      XJPEPS=X(J,K)*OPEPS
      IF(Z .GT. XJPEPS) GO TO 355
      XJMEPS=X(J,K)*OMEPS
      IF(Z .GE. XJMEPS) GO TO 265
      X(J,K)=Z
  264 IMP(J,K)=I
      GO TO 355
  265 IF(X(J,K) .EQ. BIG) GO TO 264
      IF(IPL-NPLDP) 275,270,295
  270 PRINT 301
      GO TO 295
```

```
      NPRED=IMP(J,K)
  280 IF(NPRED .LT. LTEST) GO TO 285
      LINK=SHIFT(NPRED, -ISHFT)
      NPRED=NPOOL(LINK)
      GO TO 280
  285 NPOOL(IPL)=I
      I1=SHIFT(IPL,ISHFT) .OR. NPRED
      IF(NPRED .EQ. IMP(J,K)) GO TO 290
      NPOOL(LINK)=I1
      GO TO 295
  290 IMP(J,K)=I1
  295 IPL=IPL+1
  355 CONTINUE
C     ADD ONE TO NP FOR EACH NEIGHBOR I THAT MINIMIZED X(J,K).
      NJ=IMP(J,K)
  360 I=NJ .AND. LOW
      IAR=IAD(I)
      IF(IAR .NE. 0) GO TO 362
      PRINT 231,IR,IR2,I,J,LJ,LJ2,M,M2,(NODE(I,L),L=1,4)
      IFLAG=0
      RETURN
  362 NP(I12,IAR,K)=NP(I12,IAR,K)+1
      IF(I .EQ. NJ) GO TO 365
      LINK=SHIFT(NJ, -ISHFT)
      NJ=NPOOL(LINK)
      GO TO 360
  365 IF(L .GE. N) GO TO 370
      L=L+1
      GO TO 245
  370 CONTINUE
      RETURN
      END
```

```
      FUNCTION IAD(I)
      COMMON IPERM(155),ITEMP(155),NODE(155,4),IPR(155),ICSV(5),
     1 LSV(5),JSV(5),I2SV(5),IR3SV(5),NPOOL(1000),IREG(82,2),
     2 X(155,20),IMP(155,20),NP(2,656,20)
      COMMON / A / IR,IR2,LJ,LJ2,M,M2,I12
C     I IS AN ARGUMENT TO FUNCTION IAD.  J IS AN IMPLICIT ARGUMENT--IR,
C     IR2, LJ, LJ2, M AND M2 ARE DETERMINED BY J AND SET BEFORE CALLING
C     IAD.
C     IF NODES I AND J ARE NOT ADJACENT, THEN IAD=0.  OTHERWISE, IAD IS
C     THE ADDRESS OF ARC(I,J) IN THE ARC LIST.  IN ADDITION, THE INDEX
C     I12 OF COMMON /A/ IS SET TO 1 IF THE LOCAL NODE NUMBERS SATISFY
C     LI LT. LJ AND 2 OTHERWISE.
      IF(NODE(I,1) .NE. IR) GO TO 5
      LI=NODE(I,2)
      GO TO 10
 5    IF(NODE(I,3) .NE. IR) GO TO 20
      LI=NODE(I,4)
 10   IF(LJ .GT. LI) GO TO 15
      IAD=M+LI
      I12=2
      RETURN
 15   IAD=IREG(IR,1)+(LI-1)*IREG(IR,2)-LI*(LI+1)/2+LJ
      I12=1
      RETURN
 20   IF(NODE(I,1) .NE. IR2) GO TO 25
      LI=NODE(I,2)
      GO TO 30
 25   IF(NODE(I,3) .NE. IR2) GO TO 50
      IF(IR2 .EQ. 0) GO TO 50
      LI=NODE(I,4)
 30   IF(LJ2 .GT. LI) GO TO 35
      IAD=M2+LI
      I12=2
      RETURN
 35   IAD=IREG(IR2,1)+(LI-1)*IREG(IR2,2)-LI*(LI+1)/2+LJ2
      I12=1
      RETURN
 50   IAD=0
      RETURN
      END
```

```
      SUBROUTINE LKSP(N1,KBAR,MR,LAG)
      COMMON IPERM(155),ITEMP(155),NODE(155,4),IPR(155),ICSV(5),
     1 LSV(5),JSV(5),I2SV(5),IR3SV(5),NPOOL(1000),IREG(82,2),
     2 X(155,20),IMP(155,20),NP(2,656,20)
      COMMON / A / IR,IR2,LJ,LJ2,M,M2,I12
      COMMON / B / NMX(25),LOW,ISHFT,N12
      DIMENSION IVAR(4),MR(1)
      DATA IVAR / 32H(1X,F12.5,          /(14X,25I4)) /
      DATA LTOP,ICTOP / 5,155 /
C     LIST THE K-TH SHORTEST PATHS, 1 LE. K LE. KBAR, FROM OFF-SITE TO
C     EACH HARDWARE NODE.  PATHS ARE GIVEN AS A NODE SEQUENCE FROM
C     HARDWARE THROUGH BOUNDARY.
C     LAG = 0 MEANS TO PRINT ALL THE KBAR PATHS.
C           1 MEANS TO PRINT ONLY THE PATHS WHICH CONTAIN NO TWO ADJACENT
C             ARCS IN THE SAME REGION.
      DO 25 IH=1,N1
      DO 20 K=1,KBAR
      L=0
      I=IH
      IC=1
      IP=0
      IPR(1)=I
      I2=K
      IR4=0
5     IR3=IR4
      J=I
      NPRED=IMP(J,I2)
10    I=NPRED .AND. LOW
      IC=IC+1
      IF(IC .LE. ICTOP) GO TO 9
8     FORMAT(* PATH LONGER THAN*I5*  INCREASE DIMENSION OF IPR AND VALUE
     1 OF ICTOP*)
      PRINT 8,ICTOP
      GO TO 18
9     IPR(IC)=I
      IF(I .EQ. NPRED) GO TO 11
      L=L+1
      IF(L .LE. LTOP) GO TO 35
30    FORMAT(* STACKING REQUIRES MORE THAN*I5*  INCREASE DIMENSION OF IC
     1SV, LSV, JSV, I2SV, IR3SV AND VALUE OF LTOP*)
      PRINT 30,LTOP
      RETURN
35    JSV(L)=J
      LSV(L)=SHIFT(NPRED, -ISHFT)
      ICSV(L)=IC-1
      I2SV(L)=I2
      IR3SV(L)=IR3
11    IR=NODE(J,1)
      LJ=NODE(J,2)
      IR2=NODE(J,3)
      LJ2=NODE(J,4)
      M=IREG(IR,1)+(LJ-1)*IREG(IR,2)-LJ*(LJ+1)/2
      IF(IR2 .NE. 0) M2=IREG(IR2,1)+(LJ2-1)*IREG(IR2,2)-LJ2*(LJ2+1)/2
      IAR=IAD(I)
      IF(IAR .NE. 0) GO TO 13
```

```
12    FORMAT(*0FROM LKSP IR OR IR2 IS WRONG IN FUNCTION IAD*12I5)
      PRINT 12,IR,IR2,I,J,LJ,LJ2,M,M2,(NODE(I,LL),LL=1,4)
      RETURN
13    IF(LAG .EQ. 0) GO TO 14
      IR4=MR(IAR)
      IF(IR4 .EQ. IR3) GO TO 18
14    IF(I .GT. N12) GO TO 15
      I2=NP(I12,IAR,I2)
      GO TO 5
15    IPP=MIN0(IP+1,25)
      IVAR(2)=NMX(IPP)
      PRINT IVAR,X(IH,K),(IPR(KK),KK=IPP,IC)
18    IF(L .EQ. 0) GO TO 20
      J=JSV(L)
      LINK=LSV(L)
      IC=ICSV(L)
      I2=I2SV(L)
      IR3=IR3SV(L)
      IP=IC
      IF(LAG .NE. 0) IP=0
      NPRED=NPOOL(LINK)
      L=L-1
      GO TO 10
20    CONTINUE
25    CONTINUE
      RETURN
      END
```

Acknowledgements

We want to thank S. L. Daniel, 5411, and G. B. Varnado, 5412, for supplying us with some of the test problems.

REFERENCES

[1] H. A. Bennett, The "EASI" Approach to Physical Security Evaluation, SAND76-0500, Sandia Laboratories, Albuquerque, New Mexico, January 1977.

[2] L. D. Chapman, Effectiveness Evaluation of Alternative Fixed-Site Safeguard Security Systems, SAND75-6159, Sandia Laboratories, Albuquerque, New Mexico, July 1976.

[3] E. W. Dijkstra, A Note on Two Problems in Connexion with Graphs, Numer. Math. 1, 269-271 (1959).

[4] S. E. Dreyfus, An Appraisal of Some Shortest-Path Algorithms, Operations Res. 17, 395-412 (1969).

[5] W. Hoffman and R. Pavley, A Method for the Solution of the Nth Best Path Problem, J. Assoc. Comput. Mach. 6, 506-514 (1959).

[6] B. L. Hulme, Graph Theoretic Models of Theft Problems. I. The Basic Theft Model, SAND75-0595, Sandia Laboratories, Albuquerque, New Mexico, November 1975.

[7] _____ Pathfinding in Graph-Theoretic Sabotage Models. I. Simultaneous Attack by Several Teams, SAND76-0314, Sandia Laboratories, Albuquerque, New Mexico, July 1976.

[8] _____ and D. B. Holdridge, SPTH3: A Subroutine for Finding Shortest Sabotage Paths, SAND77-1060, Sandia Laboratories, Albuquerque, New Mexico, July 1977.

[9] T. A. Williams and G. P. White, A Note on Yen's Algorithm for Finding the Length of All Shortest Paths in N-Node Nonnegative-Distance Networks, J. Assoc. Comput. Mach. 20, 389-390 (1973).

[10] J. Y. Yen, Finding the Lengths of All Shortest Paths in N-Node Nonnegative-Distance Complete Networks Using $\frac{1}{2}N^3$ Additions and N^3 Comparisons, J. Assoc. Comput. Mach. 19, 423-424 (1972).

DISTRIBUTION:

USNRC Distribution Section
Attn: Robert Wade
Washington, DC 20555
NRC-13 (208)

1000 G. A. Fowler
1140 W. D. Weart
1141 L. R. Hill
1141 D. B. Holdridge (10)
1230 W. L. Stevens
1233 R. E. Smith
1233 M. D. Olman
1310 A. A. Lieber
Attn: W. F. Roherty, 1311
1700 O. E. Jones
1710 V. E. Blake
1712 J. W. Kane
1738 J. Jacobs
1739 J. D. Williams
1739 D. L. Mangan
1750 J. E. Stiegler
1750A M. N. Cravens
1750A J. M. Demontmollin
1751 T. A. Sellers
1751 J. L. Darbey
1751 B. R. Fenchel
1751 L. C. Nogales
1751 A. E. Winblad
1752 M. R. Madsen
1754 J. F. Ney
1754 J. L. Todd, Jr.
1758 T. J. Hoban
1758 D. D. Boozer
1758 R. C. Hall
1758 G. A. Kinemond
1758 W. K. Paulus
1758 I. G. Waddoups
1758 R. B. Worrell
4010 C. Winter
5000 A. Narath, Attn: Directorates 5200, 5800
5100 J. K. Galt
5110 F. L. Vook
5120 G. J. Simmons
5121 R. J. Thompson
5121 P. J. Slater
5122 L. F. Shampine
5122 B. L. Hulme (50)
5130 G. A. Samara
5150 J. E. Schirber
5160 W. Herrman
5400 A. W. Snyder
5410 D. J. McCloskey
5411 S. L. Daniel
5412 J. W. Hickman
5412 D. E. Bennett
5412 D. M. Ericson, Jr.
5412 G. B. Varnado
5700 J. H. Scott
5740 V. L. Dugan
5741 L. D. Chapman
5741 K. G. Adams
5741 H. A. Bennett
5741 D. Engi
5741 L. M. Grady
5741 R. D. Jones
5741 R. G. Roosen
5741 D. W. Sasser
5741 A. A. Trujillo
5742 S. G. Varnado
8300 B. F. Murphey
8320 T. S. Gold
8321 R. L. Rinne
8266 E. A. Aas (2)
3141 C. A. Pepmueller (Actg.) (5)
3151 W. L. Garner (3)
For ERDA/TIC (Unlimited Releas
3171-1 R. Campbell (25)
For ERDA/TIC