NUREG/CR-0869
SAND79-1205
RS

ADPATH: An Adversary Path Subroutine

Bernie L. Hulme, Christine A. Morgan

Printed July 1979

Sandia Laboratories

Prepared for

U. S. NUCLEAR REGULATORY COMMISSION

NOTICE

The views expressed in this report are not necessarily those of the U. S. Nuclear Regulatory Commission

Available from

National Technical Information Service

Springfield, Virginia 22161

NUREG/CR-0869
SAND79-1205
RS

ADPATH: An Adversary Path Subroutine

Bernie L. Hulme
Christine A. Morgan

Date Published: July 1979

Sandia Laboratories
Albuquerque, New Mexico 87185
operated by
Sandia Corporation
for the
U.S. Department of Energy

Prepared for
Division of Safeguards, Fuel Cycle and Environmental Research
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555
Under Interagency Agreement DOE 40-550-75
NRC FIN No. A1060

ABSTRACT

ADPATH is a subroutine which safeguards analysts can use to find adversary paths in fixed-site facility safeguard studies. It was developed primarily to become a pathfinding module in SAFE, the Safeguards Automated Facility Evaluation process. The subroutine accepts a weighted digraph as a model of the facility and its safeguards system. Given a list of sabotage and/or theft targets and appropriate guard-response times, ADPATH calculates an adversary's minimum interruption probability path for each target. These are paths that minimize the adversary's probability of being detected while the guards still have time to respond to an alarm and interrupt the theft or sabotage.

CONTENTS

FIGURES

TABLES

1. Introduction

The Safeguards Automated Facility Evaluation (SAFE) process [1] currently uses the deterministic pathfinding subroutine MINDPT [3]. This code accepts a network model of the plant layout, its barrier and alarm systems and its sabotage targets and produces single-target paths in the network which correspond to good physical routes to each sabotage target. The subject of the present report is a new subroutine, ADPATH, which finds single-target theft and sabotage paths (adversary paths) in a network model that is considerably improved. ADPATH also makes use of the latest results in our research on implementation of basic pathfinding algorithms [4] by computing the adversary path segments with a faster and more storage efficient shortest path algorithm.

The name MINDPT derives from the chosen objective that an optimal sabotage path minimizes detection probability up to a guard-response time away from the target and minimizes time thereafter. This combined measure has come to be called minimum interruption probability since the path minimizes the probability that the adversary will be detected with sufficient time for the guards to respond and interrupt him before he completes the sabotage of his target. ADPATH uses this same measure. But, where MINDPT finds sabotage paths for outsiders using an undirected facility graph, ADPATH finds theft and sabotage paths for insiders or outsiders using a facility digraph (directed graph). This means not only that ADPATH has the added capability of theft path calculation but also that both kinds of paths are found in a network that is more realistic because it admits barriers whose delay times and detection probabilities depend on the adversary's direction of travel. The simple, but very important, device of allowing the user to specify an arbitrary set of starting nodes allows consideration of both insiders and outsiders as adversaries.† In addition the digraph model is more realistic because it uses two or more nodes (instead of one) to model the barrier penetration points and targets. This allows each side of a barrier or target to be distinguished and multiple alarm systems at a single barrier or target to be modeled.

---

†We are indebted to H. A. Bennett for this valuable suggestion.

2. The Facility Digraph Model

Any facility may be viewed as a boundary, some internal regions separated by barriers, and certain targets, i.e., vital equipment or protected material. The safeguards system consists of these barriers, including the boundary and the target containment, as well as detection and alarm devices such as vibration sensors, closed circuit TV, electrical switches, guards and credential verification procedures. The boundary and the internal barriers divide the facility into regions some of which contain targets.

A facility digraph is a set of nodes and a set of (directed) arcs which model these features of a facility. The nodes represent points on both sides of each boundary and barrier penetration point and on both sides of each target. They are known accordingly as boundary, barrier and target nodes. The nodes are joined together by the arcs, which represent ways to move from one point to another and hence have associated directions.

A safeguards analyst chooses the node set after studying the facility blueprints. Obviously the process of identifying possible penetration points is a crucial discretization process with some degree of arbitrariness, and an analyst can use several different levels of refinement in modeling a given facility.

Once the node set is specified, however, the arc set is completely determined. Assume for the moment that each barrier and boundary penetration point and each target can be modeled by a pair of nodes, one on each side of the obstacle. Then each pair of barrier and each pair of boundary nodes is joined by a pair of oppositely directed penetration arcs along which the barrier or boundary can be penetrated in either direction. The external boundary nodes are called off-site nodes. One of the two nodes for a target is called the target initiation node, and the other is called the target completion node. A single target arc joins a target initiation node to its corresponding target completion node and represents either equipment destruction or material removal. Finally, the region arcs, representing transit, are defined so that (a) a pair of oppositely directed arcs joins every barrier and boundary node to every other barrier and boundary node in the same (internal) region, (b) every barrier and boundary node has a single arc to each target initiation node in the same region, and

(c) every target completion node has a single arc to every barrier, boundary and target initiation node (other than its own) in the same region.

Figure 1 shows a facility digraph having two nodes for each penetration point and each target. Node 14 is an off-site node, 13 is a boundary node, 7 and 8 are barrier nodes, 2 is a target initiation node, and 1 is a target completion node. Arc (5,6) is a penetration arc, (2,1) is a target arc, and (6,12) and (12,2) are region arcs. The interested reader may compare this model with Figure 1 of [3], which models the same facility, to see how closely related yet more general is the new digraph model.

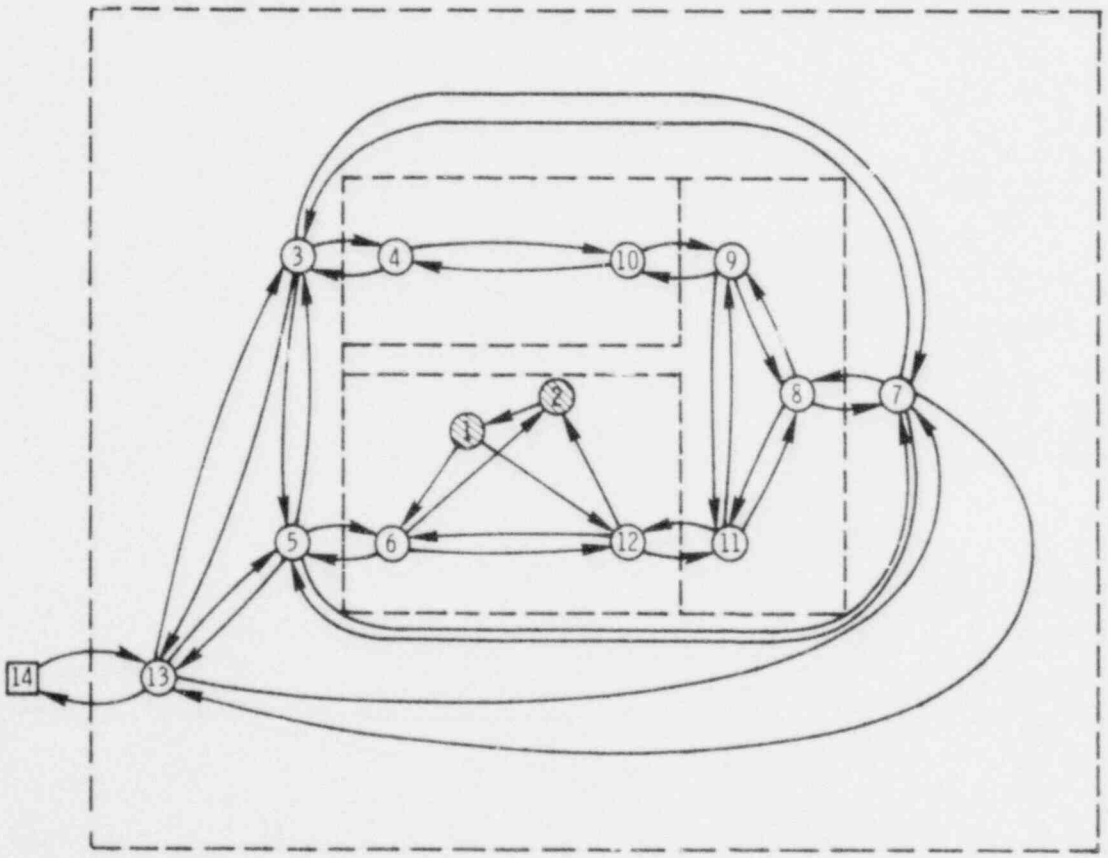

Figure 1. A Facility Digraph

The weights for the digraph consist of both delay times and detection probabilities associated with the tasks involved in traversing the arcs and nodes. Since a node is simply a point, passing through a node always involves zero time. However, each node has a detection

probability which is positive if an effective alarm is located at that point and is zero otherwise. Every arc is assigned both a delay time and a detection probability consistent with the arc's meaning. Since passing through a node requires no time, a node's positive detection probability weight indicates a jump in the cumulative probability of detection with respect to time for any path through the node. Similarly, an arc having a zero delay time, and a positive detection probability also contributes a jump in the cumulative detection probability for any path containing that arc. However, along any arc having a positive traversal time the cumulative detection probability is assumed to increase uniformly with time.

Some portals may have more than one alarm system and occasionally such portals must be represented by more than two nodes and two arcs. Any combination of alarms at a barrier or boundary can be represented by an appropriately weighted series of nodes having two oppositely directed arcs between successive pairs of nodes. For example, a door having (i) a lock on the right side but none on the left, (ii) a switch which activates an alarm with probability .6 when the door opens, and (iii) a closed circuit TV with detection probability .4 located on the right side can be modeled by four nodes and six arcs as shown in Figure 2.

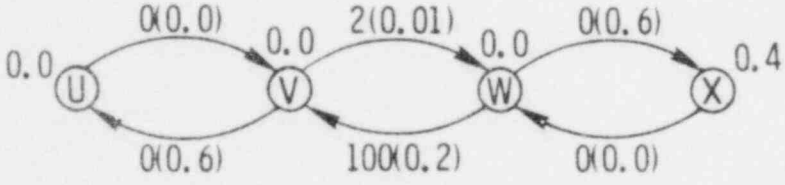

Figure 2. A Multiply-Alarmed Penetration Point

The central arcs (v,w) and (w,v) represent the tasks of penetrating the door in opposite directions. The final arcs (w,x) and (v,u) represent the task of opening the door, and node x is where the TV is located. The first and last nodes in the series, u and x, lie on opposite sides of the penetration point and are joined by arcs to other nodes in their respective regions. The intermediate nodes in the series are not connected directly to any other nodes in the digraph (see nodes 7,10 and 11 in Figure 3). Multiple alarms at a target are similarly modeled,

but there is only one path through the node series, beginning at the target initiation node and ending at the target completion node.

The only other values involved are the guard-response times. Such a time is specified for each sabotage target completion node since interruption of the adversary in this case involves guard arrival at the target before the sabotage task is completed. Theft interruption involves guard arrival at the thief's boundary exit point before the thief escapes. So when theft targets are given, guard-response times for each off-site node must be specified.

Figure 3 shows an example of a weighted facility digraph.

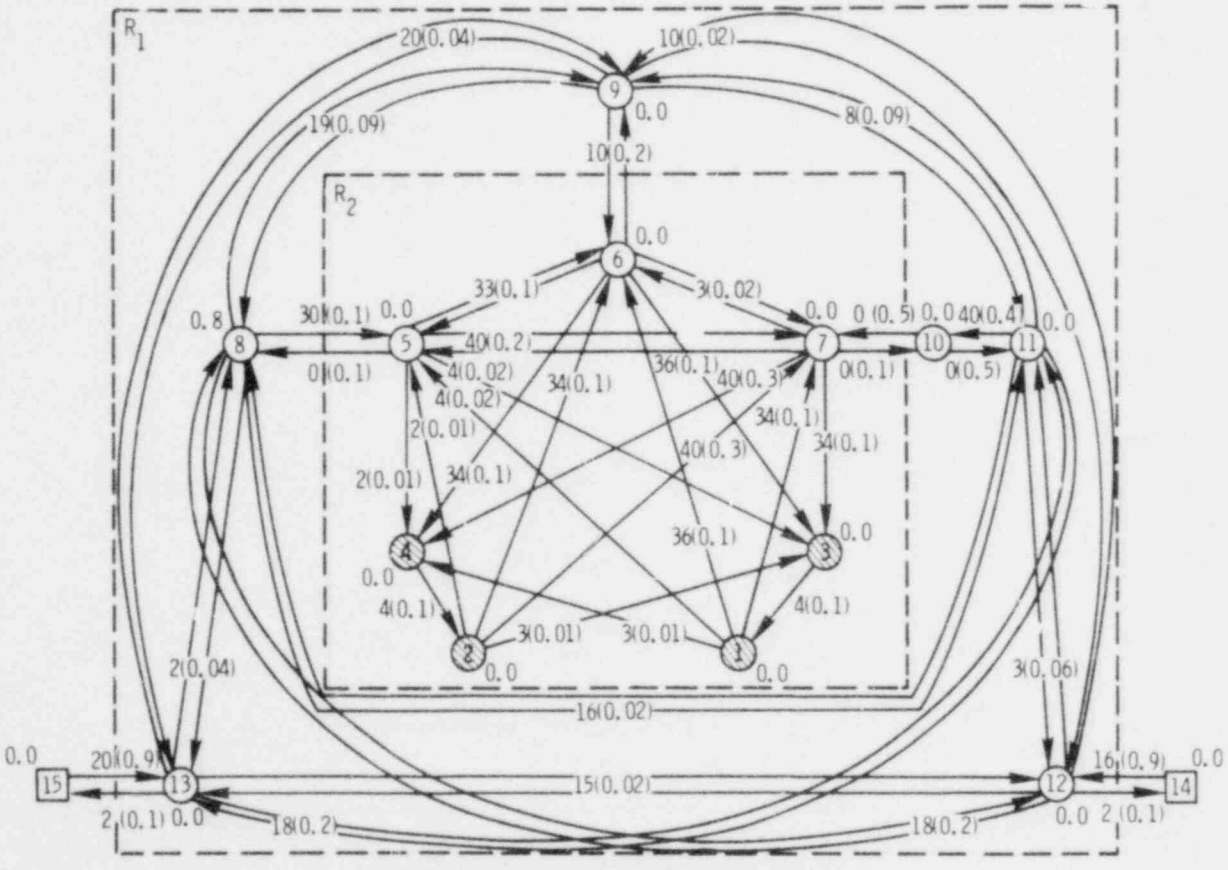

Figure 3. A Weighted Facility Digraph

3. Path Problems

It is now possible to pose the single-target minimum-interruption-probability path problem as a problem of finding certain minimum paths in the facility digraph. The time to traverse a path, of course, is simply the sum of the time weights for each arc in the path. Recall that every node has a zero delay time. The cumulative detection probability for a path is the "probability sum" of the detection probability weights for each node and arc in the path, where "probability sum" $\oplus$ is defined by

$$P_a \oplus P_b = P_a + P_b - P_a P_b.$$

In addition to the weighted digraph, the user of ADPATH specifies a list of sabotage targets and associated guard-response times, a list of theft targets and response times for each off-site node, and a list of start nodes at which the sabotage and theft paths can begin. One minimum-interruption-probability path is computed for each target listed.

The sabotage path computed for sabotage target i having response time R_i is a path that minimizes cumulative detection probability from any start node up to a locus of points whose time to target i along a shortest-time path is exactly R_i and that minimizes time from the locus to i. Notice that this guard-response-time locus is a discrete set of points which may lie anywhere along an arc or coincide with a node of the digraph. When a start node lies inside a response-time locus, the adversary need not worry about detection. By minimizing time he is guaranteed of a zero interruption probability. In this case, ADPATH produces the shortest-time path from any start node to i.

The theft path computed for theft target i is a path that begins at any start node, passes through target node i, ends at any off-site node, minimizes the cumulative detection probability from the start nodes up to the guard-response-time loci about all of the off-site nodes and minimizes time from the locus to off-site. Notice that the theft target may lie inside the locus about some off-site node, meaning that the thief switches to time minimization before he reaches the target. Suppose for example that target node i has a shortest-time path to off-site node j of length t(i,j) and the guard-response time to j is $R_j > t(i,j)$. Then the best theft path through i and out j is one

which minimizes detection probability from the start nodes up to a locus of points at time $R_j - t(i,j)$ away from target i and which minimizes time from this secondary locus through i to j. Of course, target i can lie inside the loci of two or more off-site nodes, and then a secondary locus is defined about i for each of these off-site nodes. The best path minimizes detection probability up to any one of these secondary loci and from the minimizing locus point minimizes time through i to the off-site node associated with the minimizing locus point. If a start node lies inside one of these secondary loci, ADPATH yields the shortest-time path from any start node through i to off-site. On the other hand, it maybe that the theft target i lies outside of all of the boundary loci. Then the best theft path minimizes detection probability from the start nodes to i and from i to any of the boundary loci and minimizes time from there to the corresponding off-site node.

In all cases where there are two or more locus points of minimum cumulative detection probability, the one with shortest time to the destination is best, and the path through it is reported. In case of equally short times to the destination, the first one encountered is reported.

4. Assumptions and Rationales

This code seeks only single-target paths. Often a single target is sufficient for a theft, and sometimes one target is a sufficient sabotage objective. Even when the objective involves two or more targets, it is still important to treat them separately because a very real, worst-case, possibility is that they could be attacked separately and simultaneously. Another rationale is based on computational complexity. Single-target sabotage paths can be found in polynomially bounded computation time since only the starting and ending nodes are specified. However, multiple-target sabotage paths necessarily involve finding minimal paths constrained to pass through certain intermediate nodes, a problem for which no polynomially bounded algorithm is known. A similar computational concern applies to the single-target theft problem since this target is really an intermediate node, leading us to make some further theft assumptions which are discussed later in this section.

The weights, including the guard-response times, in this deterministic model are constants. Of course, the actual task times and detection probabilities cannot be known precisely since they must vary with such things as environmental conditions and adversary performance, so some engineering judgement will be involved in the weight selections. The user of ADPATH may be conservative and use estimates of minimum possible values, or he may prefer to estimate average values for the weights. The resulting paths must always be interpreted in view of the graph weighting philosophy. It is possible also to repeatedly draw weights from given distributions and call ADPATH in a Monte-Carlo fashion to compute stochastically minimum interruption probability paths. Such a procedure based on MINDPT is already available [2].

It should be noted that the probability sum used to accumulate the detection probability weights assumes that the various detection devices are independent.

As mentioned above, the user must be aware of an important assumption used in solving for the theft paths so that the weights assigned to the digraph will be consistent with this solution process. The assumption is motivated by the fact that the theft target is an intermediate node, and the removal segment of a theft path could

revisit a node on the access portion of the path. It is generally accepted that in gaining access to a theft target the thief may alter a barrier in such a way that its weights are different than they were before the barrier was penetrated. For example, passing through a portal a second time may involve less delay time and greater detection probability. In general then, the best removal path depends on the access path that was used. It follows that a best theft path consists of a combination of access and removal paths that minimizes interruption probability. Clearly this optimal combination could involve an access path which is not the best access path but one which makes possible such an effective removal path that the combination is optimal. Therefore, solving for the best theft path, given alternate weights on all penetration arcs and barrier and boundary nodes, would involve exhaustively enumerating all access-removal path combinations. Even with branch and bound strategies such an approach would be impractical because of the enormous number of possible paths in realistic digraph models. To keep the run time of the (single-target) theft path solution process polynomially bounded, ADPATH assumes that the user has specified constant weights which are the minimum weights achievable by an inside assistant to the thief. The insider may act on the barriers and alarms before and/or after the thief first encounters them in order to keep the weights constant and minimal. This assumption eliminates the dependence of the removal path upon the access path (since the weights along the access path do not change) thereby allowing a much more efficient solution process. It also provides a conservative treatment of an important critical case involving inside assistance.

Although revisiting a barrier penetration point is not an issue in single-target sabotage path calculations, and therefore no inside assistance assumption is necessary, a user might well study sabotage paths in the presence of an inside assistant. In this case the user should adopt the same philosophy of specifying degraded weights in those regions to which an insider might gain access.

5. Calling ADPATH

Listings of ADPATH may be found on the microfiche card inside the back cover of this report. The first listing is in FLECS (FORTRAN Language with Extended Control Structures), the language in which ADPATH was developed and which is easiest to read. The FORTRAN code that resulted from use of the FLECS preprocessor is also given.

The call list for ADPATH is

```
ADPATH(N,N1,N2,N3,NO,NS,NT,NA,NSTRT,IPATHD,
       TRGT,SRT,ORT,START,DPN,T,DPA,JNDX,IPTR,
       PATH,PPTR,LPDP,LPT,KFLAG,
       RWK,IWK).
```

The arguments have the following meanings.

Input:

- N - total number of nodes (N=N1+N2+N3);
- N1 - number of target nodes (target completion nodes are numbered first, other target nodes are numbered second);
- N2 - number of barrier nodes (barrier nodes are numbered third);
- N3 - number of boundary nodes (boundary nodes other than off-site nodes are numbered fourth, off-site nodes are numbered last);
- NO - number of off-site nodes;
- NS - number of sabotage targets to be analyzed;
- NT - number of theft targets to be analyzed;
- NA - number of arcs;
- NSTRT - number of starting nodes;
- IPATHD - total number of nodes in all output paths (an estimate = (average number of nodes per path)*(NS+NT));
- TRGT - an integer array of sabotage target completion nodes followed by theft target completion nodes to be analyzed, dimensioned NS+NT;
- SRT - a real array of sabotage target response times, dimensioned NS;
- ORT - a real array of off-site node response times, dimensioned NO (needed iff NT>0);
- START - an integer array of starting nodes, dimensioned NSTRT;

DPN - a real array of detection probabilities for the nodes, dimensioned N;

T - a real array of arc delay times for the arcs from node 1 in any order, then the arcs from node 2 in any order,..., the arcs from node N in any order, dimensioned NA;

DPA - a real array of arc detection probabilities for the same arc ordering as in T, dimensioned NA;

JNDX - an integer array of the node numbers at the ends of the arcs taken in the same order as in T and DPA, dimensioned NA;

IPTR - an integer array of locations in T, DPA and JNDX of the last entry for the arcs out of each node, dimensioned N.

Notice that the last four arrays are simply an economical storage scheme for the two arc-weight matrices, which are very sparse. T and DPA are the finite, off-diagonal, entries from these matrices listed row by row. JNDX is a list of the column numbers for each of these entries, and IPTR points to the last entry from each row of the matrices. The storage required is $3NA+N$ instead of $2N^2$.

Output:

There are one output path and two numerical values for each target listed in TRGT.

PATH - an integer array of the nodes in the output paths, dimensioned IPATHD;

PPTR - an integer array pointing to the location in PATH that contains the last node of each output path, dimensioned NS+NT;

LPDP - a real array of path interruption probabilities, i.e., detection probabilities accumulated from the start node to the locus point, dimensioned NS+NT;

LPT - a real array of path times accumulated from the locus point to the end of the path, dimensioned NS+NT;

KFLAG - an integer indicating the error status,
= 0 means there were no errors,
= 1 means that IPATHD was not large enough and the problem must be rerun with IPATHD set to the value in PPTR (NS+NT);

= 2,...,10 means certain input variables had inadmissable values (see listing) which must be corrected before rerunning the problem.

Before accepting any output from ADPATH, the user must verify that KFLAG = 0.

The output path through TRGT(I) is listed as a node sequence in PATH(PPTR(I-1)+1), PATH(PPTR(I-1)+2),..., PATH(PPTR(I)). This path has the minimum possible interruption probability, LPDP(I). The locus node, which is that node in the path lying on or just outside of the response-time locus, is repeated in the node sequence. This repetition aids the user in finding the locus point, which lies either at the locus node or between the locus node and its successor. The locus point is the place where the adversary switches from detection probability to time minimization, and this point is exactly a guard-response time away from the end of the path. However, when the start node for the TRGT(I) path lies inside the locus, the start node is repeated, LPDP(I) = 0., and LPT(I) is the actual time from the start node to the end of the path, a value less than the guard-response time.

Occasionally a user may be concerned about the possibility of ties for an optimal path. To see if a particular output path is unique the user may slightly increase all of the detection probabilities along the reported path, say by multiplying them by $1+\epsilon$, and call ADPATH again. ϵ must be so small (say $\epsilon \leq 10^{-8}$) that the reported path still has a smaller interruption probability than the next smallest value. If the same path is reported again, then it is a unique optimum. Otherwise, one of the other optimal paths will be reported, its detection probabilities can be perturbed, and the process can be repeated. Obviously this applies only to paths having a nonzero interruption probability. Alternate start nodes inside a response locus may be found by inspection.

Work Arrays:

RWK - a real work array of dimension N(NO+NS+NT+1);
IWK - an integer work array of dimension N(NO+NS+NT+2) + 2NS + 3NT.

The reader should notice how rapidly the size of these two arrays grows with the number of paths requested. As NS+NT increases, the space needed to store intermediate pathfinding results grows as 2N(NS+NT). The performance results in Section 7 show that the code is very efficient with respect to run time. Also, the sparse matrix storage of the arc weights is the best possible. Therefore, the most likely constraint in dealing with very large problems will be due to storage limitations, and the user's best control over this is to call ADPATH more than once and to limit the number of paths, NS+NT, requested in each call.

6. Sample Problems

Several problems will be posed for the weighted digraph in Figure 3. Only the start nodes, the targets, and the response times will change. In practice, of course, as different adversaries are postulated for a given facility the weights could change from problem to problem.

The following data are common to all of the sample problems. Since there are 4 target nodes, 7 barrier nodes and 4 boundary nodes for a total of 15 nodes, 2 of which are off-site nodes, and there are 54 arcs, we have

N = 15 N1 = 4 N2 = 7 N3 = 4 NO = 2 NA = 54.

The greatest number of paths requested in any run will be 4. So if we estimate about 8 nodes per path, then

IPATHD = 4·8 = 32

should be a sufficient dimension for PATH in all the runs. The only node having a positive jump in detection probability is node 8, so that

DPN = (.0,.0,.0,.0,.0,.0,.0,.8,.0,.0,.0,.0,.0,.0,.0).

The sparse matrix form for the arc times, T, and the arc detection probabilities, DPA, is

```
T = ( 3., 4.,36.,34. | 3., 2.,34.,40. | 4. | 4.|
      4.,2.,33.,40.,0. | 36.,34.,33.,3.,10.|
      34.,40.,40.,3.,0. | 30.,19.,16.,18.,2.|
      10.,19.,8.,10.,20.|0.,0.|16.,8.,40.,
      3.,18.|18.,10.,3., 15., 2. | 2.,20.,18.,
      15.,2.|16.|20.),

DPA = (.01,.02, .1,.1 |.01,.01,.1,.3 |.1 | .1|
       .02,.01,.1,.2,.1|.1,.1 ,.1,.02,.2|
       .1,.3,.2,.02,.1 | .1,.09,.02,.2,.04|
       .2,.09,.09,.02,.04|.5,.5 | .02,.09,.4,
       .06,.2 | .2,.02,.06,.02,.1 | .04,.04,.2,
       .02,.1 | .9 | .9),
```

```
JNDX = ( 4,5,6,7 | 3,5,6,7 | 1 | 2 |
         3,4,6,7,8 | 3,4,5,7,9 |
         3,4,5,6,10 | 5,9,11,12,13 |
         6,8,11,12,13 | 7,11 | 8,9,10,
         12,13 | 8,9,11,13,14 | 8,9,11,
         12,15 | 12 | 13),

IPTR = ( 4,8,9,10,15,20,25,30,35,37,42,47,52,53,54).
```

Run 1

The objective of this run is to find a sabotage path to node 1 and a theft path through node 2 for an outsider, who begins at an off-site node. The guard-response time for the sabotage target is 40. Since a theft target is given, off-site response times of 20 and 6, respectively, are specified for nodes 14 and 15. The additional input then is

```
NS = 1  NT = 1  TRGT = (1,2)  SRT = (40.)  ORT = (20.,6.)
NSTRT = 2  START = (14,15).
```

The output consists of KFLAG = 0 and

```
PATH = (15,13,13,8,5,3,1,14,12,9,6,4,2,2,5,8,13,15)
PPTR = (7,18)  LPDP = (.9,.936496)  LPT = (40.,6.).
```

This means that the first 7 elements in PATH contain the node sequence for a minimum-interruption-probability sabotage path, (15,13,13,8,5,3,1), where the locus node 13 is repeated. In this case the locus point lies at node 13. The locus point detection probability (or interruption probability) is .9, and the locus point time is 40.

To see that this is the optimal path, notice that the guard-response-time locus of radius 40 about node 1 consists mainly of three points: one at node 13, one at node 6 and one 38/40 of the way from 11 to 10. We denote this locus by the set $L_1(40.) = \{13,6,38/40(11,10), \ldots\}$. The minimum-detection-probability path from START to $L_1(40.)$ is clearly (15,13), and the minimum-time path from 13 to 1 is (13,8,5,3,1).

The optimal theft path is given by the last eleven entries of PATH, (14,12,9,6,4,2,2,5,8,13,15). Backing up 6 units of time from node 15, we find that the locus point lies at node 2. To see that this is the optimal theft path, identify the guard-response-time loci for the off-site nodes,

$$L_{14}(20.) = \{5,2/10(6,9),25/40(2,7),...\},$$
$$L_{15}(6.) = \{2,...\}.$$

Node 2 lies outside the first locus and right on the second one. The minimum-detection-probability path from START through node 2 to $L_{14}(20.) \cup L_{15}(6.)$ is clearly the minimum path from START to node 2, (14,12,9,6,4,2) having a detection probability of $.9 \oplus .02 \oplus .2 \oplus .1 \oplus .1 = .936496$. Since the minimizing locus point belongs to $L_{15}(6.)$, the remainder of the theft path is a minimum-time path from the locus point to 15,(2,5,8,13,15) having a time length of 6.

Run 2

This run is to find two sabotage paths to node 1 using two different response times, and similarly to node 2, for an insider who can begin at 8, 9 or 11. The additional input here is

NS = 4 NT = 0 TRGT = (1,1,2,2) SRT = (28.,40.,21.,50.)
ORT = ¢ NSTRT = 3 START = (8,9,11).

The output consists of KFLAG = 0 and

PATH = (9,6,6,3,1,8,8,5,3,1,9,6,6,4,2,8,8,5,4,2)
PPTR = (5,10,15,20)
LPDP = (.2267,.0,.24,.0) LPT = (28.,38.,21.,36.).

If the response time to node 1 is 28, then the best sabotage path to 1 is (9,6,6,3,1), where the locus point lies 1/3 of the way from 6 to 3. The associated locus point detection probability and time are $.2 \oplus .1/3 \approx .2267$ and 24+4 = 28. However, if the response time is 40, then the best path is (8,8,5,3,1) having interruption probability .0 and time 38. Node 8 lies inside while nodes 9 and 11 lie outside of the locus $L_1(40.)$, and therefore this path is a minimum-time path from START to 1.

Similarly, when the response time to node 2 is 21, the best sabotage path is (9,6,6,4,2) having interruption probability .2 $\oplus$ .1/2 = .24 and locus point time 21. But, if the response time is 50, then the best path is (8,8,5,4,2) with interruption probability .0 and time 36.

Run 3

This run computes a theft path through node 1 and a theft path through node 2 for an insider beginning at 5, 6 or 7. The additional input is

NS = 0 NT = 2 TRGT = (1,2) SRT = ¢ ORT = (26.,12.)
NSTRT = 3 START = (5,6,7).

The results are

PATH = (5,3,3,1,5,8,13,15,5,5,4,2,5,8,13,15)
PPTR = (8,16) LPDP = (.02,.0) LPT = (12.,12.).

Both targets lie inside both the boundary loci. The shortest time from 1 to 14 is 24 < 26, and the shortest time from 1 to 15 is 8 < 12. Therefore, two secondary loci about node 1 are determined, one of radius 26 - 24 = 2, relative to off-site node 14 and one of radius 12 - 8 = 4, relative to node 15. The latter locus consisting of node 3 lies "nearer" to the START nodes in a probability sense, so the time segment is a minimum-time path from 3 through 1 to 15, i.e., (3,1,5,8,13,15) having delay time 12. The initial segment is (5,3) which minimizes detection probability from START to 3. Thus, the best theft path through 1 is (5,3,3,1,5,8,13,15) with associated values .02 and 12.

Now the secondary locus for target 2 relative to off-site node 14 has radius 26 - 22 = 4 and hence consists of node 4. The other secondary locus for target 2, relative to node 15, has radius 12 - 6 = 6 and includes a point at node 5, which is a start node. Thus, the best theft path through 2 is the minimum-time path (5,5,4,2,5,8,13,15) with associated values .0 and 12.

7. Methods and Performance

ADPATH has three parts: finding path segments, identifying and minimizing over appropriate sets of locus points, and piecing together the segments of the optimal paths.

All of the path segments are found by calling SPATHF, the path-finding portion of the subroutine SFORD. This subroutine uses a sparse matrix version of Ford's algorithm with a sequence list [4] to find shortest paths in a nonnegatively arc-weighted digraph. When shortest-time paths are needed, the weights T are supplied to SPATHF. To obtain least-detection-probability paths, the weights in DPA are used after they have been transformed by DPA (i,j)$\leftarrow$ $-\ln[(1-DPN(i))(1-DPA(i,j))]$. This transformation combines the node and arc detection probabilities into additive arc weights whose sums S are negative logarithms of non-detection probabilities for entire paths. Upon return from SPATHF these values S are converted into path detection probabilities, PDP, using the inverse transformation $PDP = 1 - e^{-S}$, where $S = -\ln(1-PDP)$.

The basic pathfinder SPATHF finds shortest paths <u>from</u> any set of source nodes to all nodes in the given digraph. It happens that ADPATH always requires minimum-detection-probability paths <u>from</u> certain sources and minimum-time paths <u>to</u> certain destinations. The latter paths are obtained by giving SPATHF the destinations as if they were sources along with the transpose of T. Since T is in sparse matrix form and can be large, a special transposition procedure was devised to transpose T "in-situ," avoiding a second array for T^t. T is transposed once, all time pathfinding is completed, and then T is retransposed. This process can reorder the out-neighbors of some nodes, and so both T and DPA are transposed and retransposed so that they are always based on the same arc-ordering. This possible reordering of the arcs out of each node is the only change ADPATH makes to the input.

ADPATH first finds <u>shortest-time paths</u> from all nodes

(a) to each sabotage target,

and, if there are theft targets (NT > 0),

(b) to each off-site node and

(c) to each theft target lying inside the response-time locus of some off-site node.

Next ADPATH finds minimum-detection-probability paths to all nodes

(d) from the START nodes and
(e) from each theft target lying outside all the boundary loci mentioned in (c).

In ADPATH's second part certain discrete loci are identified and that locus point having the least-detection-probability path from the START nodes (and possibly passing through a theft target) is sought. In finding a locus of points having a given time R_j to a node j, each node i of the digraph is considered. Let t(i,j) denote the shortest time from i to j. If $t(i,j) = R_j$, then node i is a locus point. If $t(i,j) > R_j$, then node i lies outside of the locus, and its out-neighbors k should be considered. If $t(k,j) < R_j$, then i is a locus node and there is a locus point along the arc (i,k). Let $L_j(R_j)$ denote the set of locus points whose time to j is exactly R_j.

For each sabotage target j = TRGT(I), $1 \leq I \leq$ NS, ADPATH seeks the points of the sabotage-response-time locus L_j (SRT(I)). These locus points are found one at a time and an interruption probability value is associated with each one. This interruption probability, also called a locus point detection probability, is the $\oplus$ of (a) the minimum detection probability accumulated from the START nodes to the locus node associated with the locus point and (b) the fractional arc-detection-probability from the locus node to the locus point. The least locus point detection probability for all the points in L_j(SRT(I)) is stored in LPDP(I) and represents the minimum interruption probability associated with the optimal sabotage path to TRGT(I).

Each theft target j = TRGT(I), $NS+1 \leq I \leq NS+NT$, was classified during step (c) above. Suppose j lies inside the response-time locus of an off-site node i, $N-NO+1 \leq i \leq N$, having the response time R_i = ORT (i-N+NO). Then a secondary locus of radius $\Delta_{i,j} = R_i - t(j,i)$ is defined about target j. This locus, $L_j(\Delta_{i,j})$, is the set of all points whose shortest time to i via j is equal to R_i. Let $\mathcal{L}_j$ denote the union of the loci $L_j(\Delta_{i,j})$ for all off-site nodes i such that $\Delta_{i,j} > 0$, i.e., all the boundary loci inside of which target j lies. The locus point in $\mathcal{L}_j$ having the minimum detection probability accumulated from the

START nodes determines the optimal theft path through TRGT(I) and the interruption probability to be stored in LPDP(I). For such paths the adversary switches to time minimization before he reaches the target.

The other type of theft target j lies on or outside of all boundary loci. In this case let $\mathcal{L}$ denote the union of all the boundary loci $L_i(R_i)$, where R_i = ORT(i-N+NO), N-NO+1$\leq$i$\leq$N. The locus point in $\mathcal{L}$ having the minimum detection probability accumulated from the START to target j and from target j to $\mathcal{L}$ determines the optimal theft path through j = TRGT(I) and the interruption probability to be stored in LPDP(I). On this path the adversary switches to time minimization after passing through the target.

The third part of ADPATH is simply to use the predecessor arrays from the path segment calculations (a)-(e) to list in PATH and PPTR the node sequences for the optimal paths. The tracing of predecessors to obtain a sabotage path begins with the locus node on a path and proceeds both forward along a shortest-time segment and backward along a minimum-detection-probability segment. Theft paths have three segments, but they are obtained in a similar fashion.

The storage required by ADPATH is dominated by its arrays,

array storage = 3NA+(2N+7)(NS+NT)+(2N+1)NO
+5N+NSTRT+IPATHD.

The dominant term here is almost always (2N+7)(NS+NT), not 3NA, because N tends to be large and the arc set is usually extremely sparse, i.e., $NA<<N^2$. So the number of paths requested, NS+NT, greatly affects the storage requirements.

The pathfinding part of ADPATH calls SPATHF a number of times equal to NS+1, if NT=0, and equal to NS+NT+NO+1, if NT>0. No useful theoretical run time bounds for Ford's method with a sequence list are known to us. The experiments in [4], however, suggest that the run time varies roughly as NA for sparse digraphs, where $NA<<N^2$. Based on this observation, the pathfinding part of ADPATH seems to have a run time of O(NA(NS+NT)). The locus point minimization procedure is O(N(NS+NT)), and the path listing process is negligible. Thus, the run time of ADPATH seems to be O((NA+N)(NS+NT)).

Table I summarizes ADPATH's performance on problems of various sizes. The sample problems of Section 6 are not among these. The size and location of the set of START nodes makes essentially no difference. Notice that the dramatic increase in array storage for the larger problems is due mainly to the larger number of paths requested, not to the increased digraph size. For example, in Problem 11 the 48,023 words needed for NS+NT=60 reduces to 11,030 words when NS+NT=1 and increases by 627 for each additional target. Also, the very fast execution times of SPATHF observed in [4] combine here to find 60 paths in Problem 11, a realistic reactor site model having 310 nodes and 1592 arcs, in about one second of CDC 7600 run time.

Table I. ADPATH Performance

Problem	Nodes N	Arcs NA	Sabotage Targets NS	Theft Targets NT	Off-Site Nodes NO	Start Nodes NSTRT	PATH Dimens. IPATHD	Array Storage (decimal)	Run Time Seconds CDC7600 Outsiders	Insiders
1	16	43	3	3	2	3	42	554	0.006	0.006
2	16	43	3	3	2	3	46	558	0.005	0.004
3	20	64	4	1	2	3	83	695	0.007	0.006
4	16	43	7	2	2	2	53	681	0.006	0.005
5	14	39	1	1	1	3	28	317	0.003	0.004
6	28	92	4	4	2	4	112	1,150	0.013	0.012
7	34	97	1	1	8	8	16	1,187	0.008	0.008
8	38	117	1	1	8	8	12	1,343	0.012	0.013
9	92	315	5	1	14	14	47	5,202	0.057	0.051
10	164	1,172	20	20	4	4	476	19,532	0.397	0.394
11	310	1,592	30	30	5	18	954	48,023	1.021	1.010

References

[1] L. D. Chapman et al., Safeguards Automated Facility Evaluation (SAFE) Methodology, SAND78-0378, Sandia Laboratories, Albuquerque, August 1978.

[2] D. Engi and J. S. Shanken, PATHfinding Simulator (PATHS) Users Guide, SAND78-2177, Sandia Laboratories, Albuquerque, to appear.

[3] B. L. Hulme, MINDPT: A Code for Minimizing Detection Probability up to a Given Time Away from a Sabotage Target, SAND77-2039, Sandia Laboratories, Albuquerque, December 1977.

[4] B. L. Hulme and J. A. Wisniewski, A Comparison of Shortest Path Algorithms Applied to Sparse Graphs, SAND78-1411, Sandia Laboratories, Albuquerque, August 1978.

DISTRIBUTION:

U.S. Nuclear Regulatory Commission (193 copies for RS)
Division of Document Control
Distribution Services Branch
7920 Norfolk Branch
Bethesda, MD 20014

U.S. Nuclear Regulatory Commission
MS 88155
Washington, DC 20555
Attn: M. Fadden

U.S. Nuclear Regulatory Commission (2)
MS 1130SS
Washington, DC 20555
Attn: R. Robinson

Los Alamos Scientific Laboratory
Attn: G. R. Keepin, R. A. Gore, E. P. Schlonka, D. G. Rose
Los Alamos, NM 87544

Allied-General Nuclear Services
Attn: G. Molen
P.O. Box 847
Barnwell, SC 29812

Lawrence Livermore Laboratory
University of California
P.O. Box 808
Attn: A. Maimoni
Livermore, CA 94550

Pritsker and Associates, Inc.
P.O. Box 2413
Attn: F. H. Grant
West Lafayette, IN 47906

Union Carbide Corporation
Nuclear Division
Bldg. 7601
Attn: D. Swindle
Oak Ridge, TN 37830

1230 W. L. Stevens, Attn: R. E. Smith, 1233
1700 W. C. Myre
1710 V. E. Blake, Attn: M. R. Madsen, J. W. Kane
1716 R. L. Wilde, Attn: B. D. Link, 1716
1730 C. H. Mauney, Attn: J. D. Williams, 1739
1750 J. E. Stiegler, Attn: M. J. Eaton, D. L. Mangan, 1759
1754 I. G. Waddoups, Attn: J. L. Todd, 1754
1758 C. E. Olson, Attn: D. D. Boozer, G. A. Kinemond, 1758
1760 J. Jacobs, Attn: M. N. Cravens, J. M. deMontmollin, 1760A
1761 T. A. Sellers, Attn: A. E. Winblad, J. L. Darby, 1761
1762 H. E. Hansen
1765 D. S. Miyoshi
4400 A. W. Snyder
4410 D. J. McCloskey
4414 G. B. Varnado
4416 L. D. Chapman
4416 K. G. Adams

4416 J. A. Allensworth
4416 H. A. Bennett
4416 D. Engi
4416 L. M. Grady
4416 C. P. Harlan
4416 R. D. Jones
4416 M. T. Olascoaga
4416 C. J. Pavlakos
4416 D. W. Sasser
4416 D. R. Strip
5000 J. K. Galt
5600 D. B. Shuster, Attn: A. A. Lieber, 5610, M. M. Newsom, 5620
R. C. Maydew, 5630
5640 G. J. Simmons, Attn: R. J. Thompson, 5641, L. F. Shampine,
5642
5641 C. A. Morgan
5642 B. L. Hulme (5)
3141 T. L. Werner (5)
3151 W. L. Garner (3)
For: DOE/TIC (Unlimited Release)
3172-3 R. P. Campbell (25)
For NRC for NTIS
8266 E. A. Aas