SAND78-1349

October 1978

A PATH ENUMERATION PROGRAM (ENUMPTH) FOR PHYSICAL PROTECTION EFFECTIVENESS EVALUATION

Ronald C. Hall
Safeguards Evaluation Division 1758

Sandia Laboratories
Albuquerque, New Mexico 87185

ABSTRACT

This report describes the structure and use of ENUMPTH, a program for enumerating paths which an adversary might follow in attempting defeat of physical protection systems. The paths are evaluated in terms of the probability of detecting and then interrupting the adversary as the paths are traversed. The program is intended to be practical in orientation, selecting all paths which meet some specified minimum criteria. The nature of the physical protection issue suggests that all such paths may be of equal interest to analysts who are concerned with a total facility. An example is given to demonstrate the program's applicability to practical problems.

A PATH ENUMERATION PROGRAM (ENUMPTH) FOR PHYSICAL PROTECTION EFFECTIVENESS EVALUATION

Table of Contents

A PATH ENUMERATION PROGRAM (ENUMPTH) FOR PHYSICAL PROTECTION EFFECTIVENESS EVALUATION

1. Introduction

The ENUMPTH program treats one aspect of physical protection effectiveness evaluation. It identifies relevant paths through a facility which are characterized by the lowest probability for interrupting adversary action. It recognizes that detection must occur while enough time remains for a guard force to respond. Both sabotage and theft paths are considered. The purpose of this report is to identify suitable applications for the program and to serve as a user's guide for those employing it as a design or analysis tool.

2. General Discussion

Effective design and analysis of physical protection systems necessitate assessment of the degree of protection achieved against <u>sabotage</u> and <u>theft</u> of special nuclear material (SNM). In this context, theft means the unauthorized removal of SNM from a facility. It implies penetration of the facility by an adversary, access to the SNM, and finally departure from the facility while still possessing the SNM. Sabotage means commiting a deliberate act directed against a facility which can result in exposure of the public to radiation to the extent that health and safety are endangered.

It is appropriate to refer to a sequence of actions necessary to accomplish a theft or sabotage goal as a scenario. A scenario can be associated with a <u>path</u>, which is an ordered set of points into or through a facility. Each path is characterized by a traversal time, which is the amount of time required to perform all tasks necessary to traverse the path. At any point along the path, a current penetration time and time remaining can be determined. The penetration time is the amount of time necessary to accomplish the task(s) necessary to reach the next point in the path. The time remaining is the sum of penetration times at all subsequent points in the path. A probability of detection can be associated with performance of the tasks necessary to reach the point which is adjacent to the current position (point) in a path. This permits calculation of a joint cumulative probability of detection up to any current point in a path.

Probabilities of detection and penetration times associated with paths provide a basis for quantitative assessment of the effectiveness of physical protection systems. The Department of Energy [1] has recognized measurement of the detection and delay capabilities provided by a physical protection system as a first measure of performance of the system. Path analysis techniques are suggested as a means by which a physical protection system can be evaluated to determine these measures. Cravens [2] has identified critical paths as those paths for which the cumulative detection probability does not reach an acceptable level (as specified by a regulatory or other empowered agency) before the time remaining for an adversary to complete the path is less than the time required for response force arrival. This is the basis for the analysis performed by the ENUMPTH program.

The foregoing characterization of adversary actions permits graph-theoretic analysis of physical protection systems. This is true in the case of design, prior to installation of proposed physical protection hardware. It is also true in the case of assessing the protection afforded by existing systems. In either event, the first step in performing the analysis is preparation of a facility description (e.g., specifying the location of points which must be traversed in paths through the facility). Cravens [2] treats this process from a design perspective. Once such a description is available, detection probabilities and penetration times can be expressed as in Appendix B to this report. Commonality must be considered in this assignment of values. Commonality is the property associated with a change in probability of detection or penetration time due to a previous defeat of a physical protection component in a sequence of adversary actions. This consideration is represented under the column heading "Common Out" of Appendix B. The data included in Appendix B constitute a graph-theoretic representation of a physical protection system. Nodes are points in the paths. Adjacencies exist between any two consecutive points in any possible path. Target nodes are nodes which provide access to SNM. The total representation constitutes a graph.

The program employs a breadth-first search technique [3] for selecting paths, which means that paths are enumerated by fanning out from each node. Each possible adjacency is evaluated as paths are enumerated. This is distinguished from depth-first search, which enumerates a complete path and then "backtracks" to evaluate alternate adjacencies. In any event, ENUMPTH retains all paths which meet certain user-specified criteria. The nature of the physical protection issue (as described above) suggests that there is interest in all paths which meet the criteria used to define criticality.

The program represents a deterministic system. There is no engagement model. It is assumed that if detection occurs early enough, a satisfactory response force could be provided to interrupt the adversary. The next section describes the detailed structure of the program. Those interested only in using the existing program may prefer to skip that discussion. Section 4 specifies the input formats for providing data to ENUMPTH.

3. Program Description

ENUMPTH has been implemented in extended FORTRAN on the CDC 6600 computer. It maintains all relevant path data in core until available space is exhausted, and then utilizes a paging mechanism as necessary to overflow to mass storage. Two 5984 word pages (one being processed and one for overflow) are always core resident. The program accepts problem descriptions containing up to 100 nodes. An upper limit of 1,000 mass storage pages provides sufficient working space for any conceivable application of the program.

3.1 Program Structure

A small main driver invokes subroutine modules which access global data in an unlabeled common block. Appendix A lists and describes the global variables. The general functioning of the modules is described now.

ENUMPTH:

This is the main program. It opens the mass storage area for the paging mechanism, calls on the initialization subroutine, and passes control to an iterative module to do the path analysis. Control returns after paths meeting the selection criteria for the problem have been enumerated and evaluated. A termination module is called which prints the results of the problem analysis. An attempt is made to read further search parameters from a second SCOPE INPUT file [4,5]. The program terminates if the file is empty. If further input specifications are found, problem analysis is again performed using the new search criteria and the original graph-theoretic representation for the facility. Further detail of this activity is given in the discussion of program input.

INITMOD:

This is the initialization module. It operates in two modes depending on the value of its one parameter. If INITLZE = 1, complete system initialization takes place including calculation of lengths for the path data to be stored on the pages. The graph-theoretic representation for the facility being analyzed is read. Pointers and data pertaining to available overflow space are established. The functioning of these data items is described later in connection with the program's data structure. If INITLZE $\neq$ 1, reinitialization takes place for current search parameters.

ITERMOD:

This module iteratively processes pages of path data, repetitively calling upon the INTNOD module to extend intermediate paths to adjacent nodes. This is done when the resulting probabilities do not exceed previously established minimums. Stacks are maintained on each page for each node. Each stack contains the paths that have just reached the node for which the stack is being maintained. When sabotage paths have reached the target node, the paths are considered complete. The module differentiates between these and theft paths, to continue enumeration until the latter have been extended to reach the exit node.

CURPAG:

This module is invoked when processing has been completed on the page currently in memory. It retrieves the next page from mass storage so that it can be processed. It updates the page just processed, writing it to mass storage. It also updates the overflow page, and retrieves a new overflow page if necessary. The paging mechanism handles the pages in a circular manner, with the first page succeeding the last. The applicable overflow page is the next one in the chain which contains available overflow space.

OVFPAG:

The OVFPAG module is invoked in conjunction with CURPAG. It selects the next appropriate page in the chain, and retrieves it for overflow space. If no page exists which contains available space, a new page is added and initialized for that purpose. A 17 word array is maintained in core (PAGOVF) to provide 1,000 bits which are used to indicate pages with available overflow area.

INTNOD:

This is the largest subroutine of the program; and, it does the most work. It is repetitively called by ITERMOD to operate on paths in a stack on the page being processed. The purpose is to intersect the last node in the paths with any adjacent subsequent nodes. If the last node is the terminal node, the paths are evaluated for retention (minimums may have decreased since the paths were placed in the stack). Otherwise they are compared with minimums applicable to whichever node the stack concerns (which also may have decreased). Recursion is next considered.[1] If an adjacency would cause a path to loop back on itself, the path is not extended. Remaining adjacencies result in a subroutine call for CALCVAL to accumulate detection probabilities associated with the adjacency. If current minimums are not exceeded, the extended paths are placed in an appropriate stack for the node just reached. They are stacked on the current page if space is available. Otherwise they are placed in the stack on the overflow page. The space for the original path (which now may or may not have been extended) is returned as available area for new path extensions.

1. It should be recognized that theft paths which retrace entry nodes are not considered recursive.

COMNOUT:

Satisfactory treatment of theft paths requires recognition of common outs. Their representation is essential in accurately modeling barriers which need not be penetrated a second time upon leaving a facility if they have been defeated while entering. The COMNOUT module identifies such adjacencies. Transit times and detection probabilities are modified in response to this identification. The subroutine has one parameter: J. It specifies the node to which the adjacency extends.

CALCVAL:

Detection probabilities are summed as paths are extended, but the measure for probability of interruption is not incremented until sufficient time has elapsed to allow the guard force to make a response. The CALCVAL routine performs this function, concerned with both the total probability of detection and the probability of detection to the point in the path which is one response interval from its current end. The subroutine has four parameters: VAL, INODE, J, and JNP. VAL is used to return the value for the path's current probability of interruption. INODE specifies the sequential page location which contains the path being extended. J specifies the node to which the extension is being made. JNP is a pointer to the absolute location for storing the next extended path.

TERMMOD:

The primary purpose of this module is to print the processing results. Output data are described for sabotage and theft cases in the final sections of this report.

3.2 Data Structure

The paging mechanism implemented in the program has already been mentioned. A circular chain of pages is maintained and processed sequentially, extending paths to appropriate adjacent nodes. As such extensions are made, formerly used space is returned and made available for reuse. One stack can exist on each page for each node. If the node is the terminal point, the stack is a completed path list. Otherwise the stack is an intermediate path list. Pointers to the heads of these stacks are maintained at the end of the pages. A pointer is likewise maintained to a chain of available spaces on the page. The number of words required to store the data for a path varies for different problem definitions. Storage is dynamically allocated for data items as required by the problem being analyzed. The result is more efficient program execution because mass storage accesses are minimized. Storing the maximum number of paths on each page means that fewer pages are used, and they are read into memory less frequently. Figure 1 shows the layout for a general page of data.

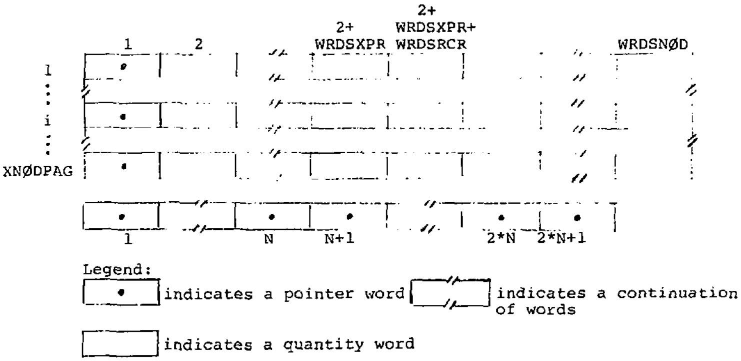


Figure 1. Data Page

The entities included on the data page require further elaboration. First, data pertinent to the general path are described. Next, the pointers included at the end of the page are discussed.

Information of several types is maintained for each path. For the general path i in Figure 1, the first word is a pointer. If the space is empty, the pointer points to the next empty space in the chain of available spaces. If the space is in use, the pointer provides the link to the next path in the same stack. A zero value represents the end of the chain in either event.

The series including the second word through the 1+WRDSXPR word is used to store path expression values. These values represent path segments in the form of integers. This method of numerical expression is discussed in separate documentation [6]. Paths are segmented, with MAXI nodes per segment. An integer value representing each sequential directed segment is stored in these words. Each segment contains the maximum number of nodes which assures that the maximum single precision integer value is not exceeded. The limiting number of nodes is dependent upon the number of nodes in the facility representation, and is calculated for each problem definition.

The series beginning with word 2+WRDSXPR and continuing through word 1+WRDSXPR+WRDSRCR is used to store bits to indicate which nodes have already been included in a path. The bits are set with elements from the BITMSK array. This allows path recursion to be avoided, enumerating only simple paths. The length of this series is determined dynamically, utilizing one 60 bit word for each set of up to 60 nodes. All bits are reset when a theft path reaches a target node, permitting the exit to be constructed independently from the manner of entry.

Path values are stored in the series of words beginning with 2+WRDSXPR+WRDSRCR. The number of words in this series is equal to three plus the integer guard response time specified by the user. The first word contains the current value of the probability of interruption associated with the path. This is the cumulative joint probability of initial detection at each node, to the point in the path which is one response interval from its current end. The second word is used for accumulation of path time since the path was last determined to meet a minimum retention criterion. The next ZRESPTM words store the increments for the probability of interruption which will occur at the discrete time units. As the path is extended, they shift through the array to eventually be added to the composite measure. The final word in this series is the last word used to store data for the path. It contains the joint probability of failing to detect adversary activity up to and including the preceding node in the path. It is initialized to unity. Its value is updated at each node by multiplying it by the complement of the applicable probability of detection. Its product with the uncomplemented value is the increment for total probability of detection. This value is added to the appropriate word in the preceding series of ZRESPTM words.

There are 2*N+1 pointers maintained at the end of the page. The first N pointers point to chains of intermediate (incomplete) paths. One of the pointers from word N+1 through 2*N will be used to point to the head of the chain for completed paths to ZENDNOD. The final pointer (2*N+1) is used to point to the head of the chain for available spaces on the page.

4. Program Input

The program will accept two card files, the second of which may be empty. The first file must contain the initial search parameters, and the adjacencies and detection probabilities for the facility. The purpose of the second file is to optionally specify additional search parameters for further analysis of the original facility definition.

The first file contains cards of up to five different types (formats). The first card must provide initial search parameters for the analysis. Then, depending on some of the parameters included on this card, specification of previously determined minimum cumulative detection probabilities to each node may be required. If so, either one or two cards for each node must immediately follow using a second card format. Finally, a variable number of cards containing node adjacency specifications are included. Three types of cards are used for this purpose. Further description of each of the five card types follows.

Figure 2 shows the format of the card containing search parameters. The card is divided into eight fields of ten columns each.

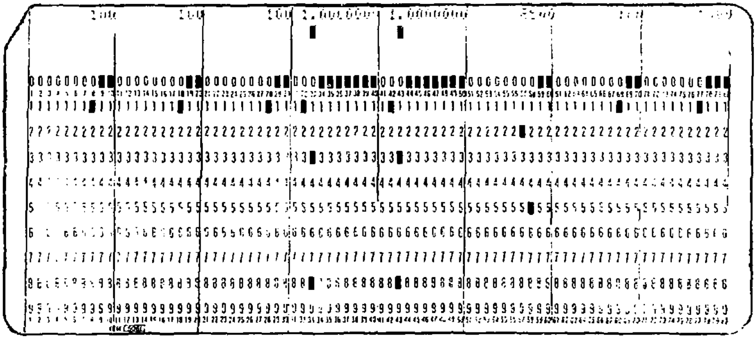


Figure 2. Card Format 1

The values shown in the figure indicate maximum quantities which the parameters should attain. Although some of the maximums are not strict, values in excess of these quantities are not recommended because degradation of program performance will result.

The first field specifies the number of nodes included in the graph-theoretic representation for the facility. The second field specifies the beginning node from which paths are to be enumerated. It can be used to represent a node connecting all points which are external to the facility's outer boundary. The third field specifies the target node. For sabotage cases, this is the end of the paths. In the case of theft problems it represents the node from which the adversary begins his exit. The fourth and fifth fields specify selection ranges for critical paths above minimum probability values. All paths falling in the ranges are retained. The ranges specified are applicable to the minimum probability of interruption and the minimum cumulative probability of detection, respectively. If either value is nonzero, the program requires additional input specifying all node minimums. The sixth field contains the response time for reacting to a detected threat. The seventh field is used only for theft analysis. It specifies the exit node. The eighth field optionally specifies a scaling factor. When it is used, it groups smaller time units into larger ones. For example, if times are specified in seconds and the user is concerned with minutes, a scaling factor of 60 can be used. The reason for its inclusion is to make execution of the program more efficient by reducing the amount of space on data pages which is needed to store path information. The scaling is applied to both the response time and to the transit times included in the input data. All fields except the fourth and fifth are integers, justified right. The selection ranges are real values allowing seven positions to the right of the decimal point.

The second card format is shown in Figure 3. It is used to specify node minimums when selection ranges are nonzero. When selection ranges are zero, specification of node minimums is not required.

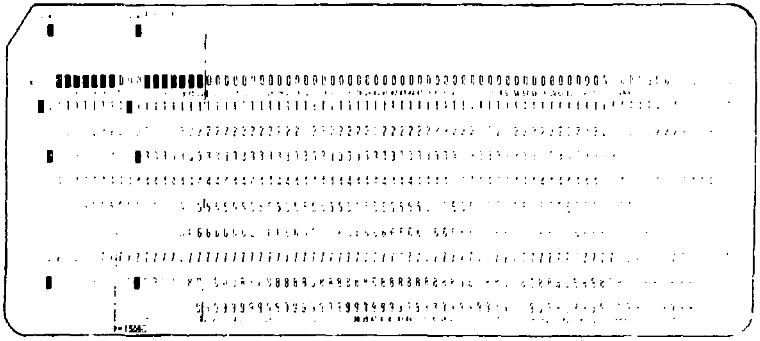


Figure 3. Card Format 2

Exactly one card is required for each node when the analysis concerns sabotage paths and nonzero selection ranges have been input. They are to be placed in sequence beginning with the first node. It is assumed that node minimums have been determined from a prior execution of the program which specified zeros as the selection ranges. The minimum probability of interruption is specified in the first field, and the minimum probability of detection is specified in the last. Both values are real numbers with seven digits to the right of the decimal point. The maximum permissible probability value is unity.

Theft analysis requires two cards for each node. One set of specifications contains minimums attained while moving inward towards the target node. These are equivalent to the minimums applicable to the sabotage case. The set of minimums applicable to outward movement after having reached the target is required next in input files for theft cases.

Figure 4 shows the third format for input cards. Data contained on it are used for reading and placement of node adjacency information. Node adjacencies are specified according to the card formats shown in Figures 5 and 6.

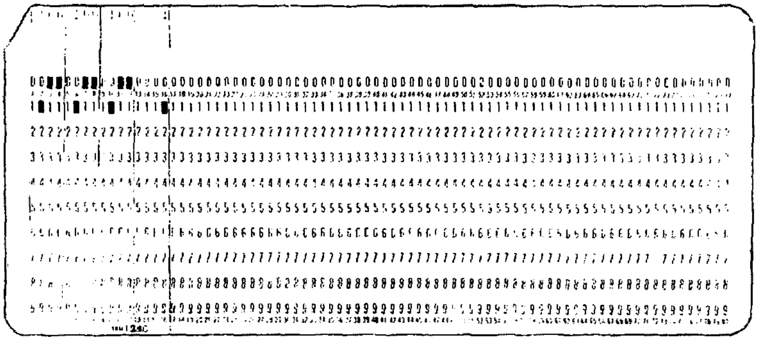


Figure 4. Card Format 3

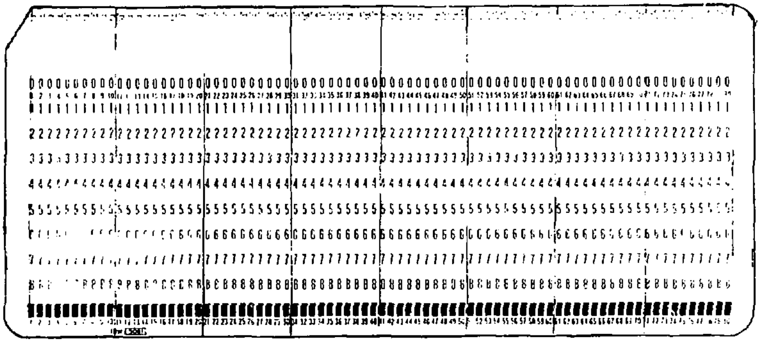


Figure 5. Card Format 4

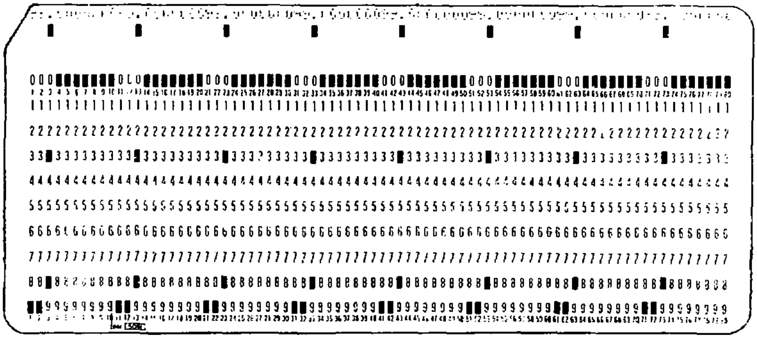


Figure 6. Card Format 5

Adjacency data are commonly contained in the structure of an adjacency matrix. This is the manner in which transit times and detection probabilities are stored for this program. It permits most rapid accessing of the data. However, when matrixes are sparse, inputting data in this form can be quite cumbersome. For this reason input is controlled according to parameters supplied on cards formatted as in Figure 4. The first value specifies the matrix row which is about to receive data. If this value is greater than N, then the row is to contain data representing a common out. (Recall that commonality is the property associated with a change in probability of detection or penetration time due to a previous defeat of a physical protection component in a sequence of adversary actions.) The node from which the adjacency extends is determined by substracting N from the value. The second parameter specifies the last column for which a value is being provided. The third parameter directs the succeeding input to be placed beginning in that column. Default values for the two column numbers are N and one, respectively, where N is the number of nodes included in the facility representation. The default case is therefore inputting an entire matrix row. The fourth value on the card must be either zero or one. A zero in this position implies that a card specifying detection probabilities will follow. A one specifies that transit times are being input.

Formats for the fourth and fifth types of input cards each contain eight fields of ten characters. In the former case the fields are right justified integer values. In the latter case the fields are real numbers containing seven places to the right of the decimal point. The integer format is used for inputting transit times. Detection probabilities require the real format. A variable number of cards are required for different facility representations. If more than eight values are required to complete a specification provided as in Figure 4, values are to be placed on sequential cards until the specification is completed. Gaps between values in matrix rows can be input. They require that non-adjacencies be represented by appropriate values. For time specification this value is 9999999999. For probability values, 99.0 is input.

The second INPUT card file can contain cards of only one type. Their format is similar to the one shown in Figure 2, except that no selection ranges or scaling factor can be included. Furthermore, if a scaling factor was included before, the response time specified must be similarly scaled. As many cards as desired can be placed in this optional file. This capability is particularly useful for repetitive analysis of varying target nodes in theft scenarios. It is also useful for analyzing the effect of varying response times.

5. An Example Problem

Graph-theoretic sabotage models have been previously constructed [7,8]. Efficient, polynomially bounded algorithms for performing the associated path analyses have been proposed [9,10]. Theft models for graph-theoretic physical protection analysis have also been considered [11]. The basic theft model addressed paths containing undirected, constant adjacencies. This was identified as the simplest model, permitting solution by an extremely efficient algorithm.

Much other preceding research has been concerned with correctly achieving maximum obtainable efficiency in general algorithms [12]. This is desired because of the theoretical limit of a factorial number of paths which are possible. Consider the case when all arc lengths are zero. In such a situation every path is a minimum path. One would certainly not want to enumerate every one if their length was all that was of interest.

Appendix E illustrates some of the simplifying assumptions which are apparent in applications of polynomially efficient search strategies to problems similar to those treated by this program. The motivation for the present work is not the same. The objective of achieving a minimal bound was relaxed in the interest of providing complete solutions to analyses which require them. Some analytical problem solving requires expression of directed travel. Travel time may be a function of direction traveled, as well as path history. Identification of all least paths may be pertinent to the problem solution. Although a minimum polynomial bound (where execution time is proportional to some constant power of the number of nodes or adjacencies) cannot be placed on such solutions, the example given below demonstrates that this approach is feasible for many physical protection analysis problems. It also provides a description of the output produced by the program.

5.1 The Facility Representation

A physical protection system design for a conceptual three-level mixed-oxide fuel fabrication facility [13] was used for constructing this example. This was done to establish comparability with the results of other work currently being done, and to assure that the problem was representative of the scope which would be encountered in practical applications of the program. An appropriate facility description existed providing a straightforward and independent source of input data. It was found that the facility could be modeled using 46 nodes connected by 166 adjacencies. Appendix B shows this representation. For purposes of this example, adjacencies have been assigned the listed detection probabilities and transit times. Program input was prepared on punched cards according to the formats described in Section 4 of this report. The nodes were assigned sequential numbers.

5.2 The Sabotage Case

Node 35 represents the end of all sabotage paths. This target node is reachable from all sabotage objectives in the facility and completes all sabotage scenarios. Its specification as the target node results in generation of all sabotage paths which meet the retention criteria. Appendix C is a listing of the output produced for this analysis.

The number of nodes in the facility representation is printed first in the output. The initial and terminal nodes for the generated paths are next given. The user-specified response time and the selection ranges above minimum interruption and detection probabilities appear on the second line. These data define the analysis which the user has requested.

Next, the minimum interruption and detection probabilities encountered to each node are listed. These processing results may be reinput for subsequent runs to select paths which fall within some specified nonzero range above the minimums. Immediately following these data, data pertaining to execution of the program are printed. First, the processing time is shown in CPU seconds. This is followed by the number of branches extended during the run. Each branch represents an adjacency which was appended to an intermediate path during the enumeration process. Together, the time required and the number of branches extended provide a measure of the computing resource requirements for the analysis.

Finally, the paths themselves are listed in the output. They appear in random order, but every path meets at least one of two retention criteria. Each is characterized by either the minimum probability of interruption, or the minimum probability of detection, or both. The paths are comprised of the adjacencies connecting nodes in the order listed.

5.3 The Theft Case

Nodes 36 through 46 represent theft objectives (targets). A separate node was used to identify each of the theft targets. This was done to insure that all exit paths proceeded outward from the target which had been reached. In any event, the paths must extend to the appropriate theft exit before they are complete. Appendix D shows the results of an analysis to select critical theft paths which pass through target node 38. Node 1 is the off-site point from which the paths begin. It is also the terminal point at which the paths end.

Several differences between this output and that for the sabotage case are apparent. The terminal node is no longer the same as the target node. Accordingly, the target node is listed separately. Also, since nodes can be included in inward path segments (towards the target) and outward path segments (away from the target), they have two sets of minimum probability values applicable to them. Therefore, four columns of minimum values are printed. Additionally, the same node may appear twice in a theft path. This is not indicative of recursion because the direction traveled in the two cases is not the same.

REFERENCES

1. U.S. Department of Energy, A Systematic Approach to the Conceptual Design of Physical Protection Systems for Nuclear Facilities, HCP/D0789-01, May 1978.

2. Cravens, M. N., and Winblad, A.E., Physical Protection System Design Method, SAND77-0119, Sandia Laboratories, Albuquerque, New Mexico, February, 1978.

3. Deo, Narsingh, Graph Theory with Applications to Engineering and Computer Science, Prentice-Hall, Englewood Cliffs, 1974, Ch. 11.

4. Smith, N. A., Scope Users' Manual for the Control Data 6600, SC-M-70-185, Sandia Laboratories, Albuquerque, New Mexico, November 1975, 3-3.

5. Control Data Corporation, FORTRAN Extended Reference Manual 6000 Version 3, 7600 Version 1, Control Data Corporation. Sunnyvale, California, 1973, 5-3.

6. Hall, R. C., A Numerical Representation for Paths in a Graph, SAND78-1350, Sandia Laboratories, Albuquerque, New Mexico, to be published.

7. Hulme, B. L., Pathfinding in Graph-Theoretic Sabotage Models I. Simultaneous Attack by Several Teams, SAND76-0314, Sandia Laboratories, Albuquerque, New Mexico, 1976.

8. Hulme, B. L., The Region Adjacency Graph in Sabotage Studies, SAND76-0574, Sandia Laboratories, Albuquerque, New Mexico, 1976.

9. Hulme, B. L., and Holdridge, D. B., SPTH3: A Subroutine for Finding Shortest Sabotage Paths, SAND77-1060, Sandia Laboratories, Albuquerque, New Mexico, 1977.

10. Hulme, B. L., and Holdridge, D. B., KSPTH: A Subroutine for the K Shortest Paths in a Sabotage Graph, SAND77-1165, Sandia Laboratories, Albuquerque, New Mexico, 1977.

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

12. Dreyfus, S. E., "An Appraisal of Some Shortest-Path Algorithms," Operations Research 17, 1969, 395-412.

13. Winblad, A. E., et al, Preliminary Facility Safeguards System Designs for a Mixed-Oxide Fuel Fabrication Facility, SAND77-1155, Sandia Laboratories, Albuquerque, New Mexico, 1977.

Appendix A - List of Global Variables

Variable	Dimension	Description
BITMSK	(60)	Array of 60 words with each of 60 bits set, from right to left
BRNNUM	scalar	Number of paths extended (branches) during current iteration of pages
BRNTOT	scalar	Total path extensions (branches); accumulation of BRNNUM
CMPSWTH	scalar	Used as switch to signal that completion of paths is appropriate (value = 1), and that all paths have reached terminal point (value = 2)
D	(N)	Double precision diagraph n-tuple (see Reference 6)
FRSTRCR	scalar	Displacement into path data for first word containing bit vector to identify recursion
FRSTVAL	scalar	Displacement into path data for first word to store path values
LP	scalar	Used to store target node while extending theft paths to exit
MINNOD	(N,2)	Used to store current path minimum to node i: minimum probability of interruption in MINNOD (i,1) and minimum probability of detection in MINNOD (i,2)
MSIX	(1001)	Array for system use in accessing mass storage pages, permits up to 1000 pages to be used
N	scalar	Number of nodes included in graphic representation of problem being analyzed
PAGBUF1	(5984)	Core area for current data page
PAGBUF2	(5984)	Core area for overflow data page
PAGIX	(2)	Indexes of PAGBUF1 and PAGBUF2 data pages, respectively

Variable	Dimension	Description
PAGLMT	(2)	Greatest used page number, and greatest valid page number, respectively
PAGOVF	(WRDSOVF)	Bit vector identifying pages with available overflow area
SLCTRNG	scalar	Scaling factor which allows user grouping of time units for response time and path time specifications
STKIX	scalar	Index into stack pointers maintained at end of pages
TIMEBGN	scalar	CPU time value (SECONDS) following initialization
TIMETRM	scalar	CPU time value (SECONDS) when results are printed
WRDSNOD	scalar	Length in words of data page area used to store data for each path
WRDSOVF	scalar	Number of words required for PAGOVF bit vector
WRDSPAG	scalar	Number of words used on each data page
WRDSRCR	scalar	Number of words used to store recursion data for each path
WRDSVAL	scalar	Number of words used to store path value data for each path
WRDSXPR	scalar	Number of words used to store path segment expressions for each path
XNODPAG	scalar	Maximum number of paths which can be stored on one data page
ZBGNNOD	scalar	Initial node from which paths are to branch
ZENDNOD	scalar	Terminal node for enumeration: target node in the case of sabotage and inward theft paths, theft exit node in the case of outward portion of theft paths

Variable	Dimension	Description
ZLSTPAG	scalar	Temporary storage for index of last page read into PAGBUF1
ZMINDEF	scalar	User specified range above minimum probability of interruption for selection of paths
ZMINDET	scalar	User specified range above minimum probability of detection for selection of paths
ZPROBDT	(2*N,N)	ZPROBDT (i,j) contains probability of detection values associated with transit from node i to node j, common out data residing in locations for which i >N.
ZRESPTM	scalar	User specifieu time for guard response
ZTHFTXT	scalar	User specified exit node to be reached for theft paths
ZVALMAT	(2*N,N)	ZVALMAT (i,j) contains transit times from node i to node j, common out data residing in locations for which i >N.
ZMIN1	(2*N)	Minimum probabilities of interruption at each node, may be determined from prior run; two node values required for theft paths
ZMIN2	(2*N)	Minimum probabilities of detection at each node, may be determined from prior run; two node values required for theft paths

Appendix B - A Facility Representation

The following symbolization is used in this Appendix:

i :	initial node for an adjacency
j :	terminal node for an adjacency
T :	time units required for traversing an adjacency
P_d:	probability of detecting adversary activity along an adjacency
$\underset{\smile}{k}$:	off-site node, k an integer
$\angle k$:	sabotage target, k an integer
$\bigtriangledown k$:	theft target, k an integer

Adjacency		Initial Penetration		Common Out	
i	j	P_d	T	P_d	T
1	1	0.0000	0	0.0000	0
1	2	.8000	300		
1	3	.9900	300		
1	4	0.0000	18		
1	34	.9900	300		
2	5	.9999	150		
2	1	1.0000	300	.9500	0
3	5	.9500	150		
3	1	.9900	300	0.0000	0
4	5	.9900	18		
4	34	.9990	300		
4	1	.9000	18	0.0000	0
5	6	0.0000	0		
5	7	.9500	150		
5	8	.9500	150		
5	9	.9500	150		
5	10	.9500	150		
5	13	.9500	150		
5	14	.9500	150		
5	15	0.0000	150		
5	16	0.0000	150		
5	17	.9500	480		
5	19	.9900	9999		
5	23	.9900	480		

Adjacency		Initial Penetration		Common Out	
i	j	P_d	T	P_d	T
5	25	.9900	480		
5	26	.9900	480		
5	28	.9900	960		
5	29	.9500	960		
5	30	.9500	9999		
5	31	.9500	960		
5	32	.9900	9999		
5	2	0.0000	150	0.0000	0
5	3	.9500	150	0.0000	0
5	4	.9000	18	0.0000	0
6	11	.8000	300		
6	12	.8000	300		
6	13	.9500	480		
6	17	.9500	150		
6	23	.9900	480		
6	24	.9900	480		
6	26	.9900	480		
6	33	.9900	480		
6	5	0.0000	0	0.0000	0
7	17	.9500	150		
7	29	.9500	150		
7	5	.9500	150	0.0000	0
8	17	.9500	150		
8	25	.9900	150		

Adjacency		Initial Penetration		Common Out	
i	j	P_d	T	P_d	T
8	30	.9500	150		
8	32	.9900	300		
8	33	.9900	150		
8	5	.9500	150	0.0000	0
9	17	.9500	150		
9	25	.9900	150		
9	26	.9900	150		
9	31	.9500	150		
9	5	.9500	150	0.0000	0
10	17	.9500	150		
10	5	.9500	150	0.0000	0
11	17	.9999	150		
11	33	1.0000	150		
11	6	1.0000	300	.9500	0
12	17	.9500	150		
12	6	.9995	300	.9500	0
13	17	.9500	150		
13	18	.8000	0		
13	19	.9900	150		
13	5	.9500	150	0.0000	0
13	6	.9500	480	0.0000	0
14	23	.9900	150		
14	24	.8000	0		
14	5	.9500	150	0.0000	0
15	28	.9900	150		
15	5	.9500	150	0.0000	0

Adjacency		Initial Penetration		Common Out	
i	j	P_d	T	P_d	T
16	34	.9900	300		
16	5	.9500	0	0.0000	0
17	18	0.0000	150		
17	19	0.0000	150		
17	20	0.0000	150		
17	21	0.0000	150		
17	22	0.0000	150		
17	23	0.0000	150		
17	24	0.0000	150		
17	25	0.0000	150		
17	26	0.0000	150		
17	27	0.0000	150		
17	28	0.0000	150		
17	29	0.0000	150		
17	30	0.0000	150		
17	31	0.0000	150		
17	32	0.0000	300		
17	33	0.0000	150		
17	5	.9500	480	0.0000	0
17	6	.9500	150	0.0000	0
17	7	.9500	150	0.0000	0
17	8	.9500	150	0.0000	0
17	9	.9500	150	0.0000	0
17	10	.9500	150	0.0000	0
17	11	0.0000	150	0.0000	0

Adjacency		Initial Penetration		Common Out	
i	j	P_d	T	P_d	T
17	12	0.0000	150	0.0000	0
17	13	0.0000	150	0.0000	0
18	35	0.0000	225		
18	36	0.0000	135		
18	13	0.0000	0	0.0000	0
18	17	.9500	150	0.0000	0
19	35	0.0000	225		
19	37	0.0000	135		
19	5	.9500	9999	0.0000	0
19	13	.9500	150	0.0000	0
19	17	.9500	150	0.0000	0
20	35	0.0000	90		
20	38	0.0000	0		
20	17	.9500	150	0.0000	0
21	35	0.0000	120		
21	39	0.0000	30		
21	17	.9500	150	0.0000	0
22	35	0.0000	120		
22	40	0.0000	30		
22	17	.9500	150	0.0000	0
23	35	0.0000	120		
23	41	0.0000	60		
23	5	.9500	480	0.0000	0
23	6	.9500	480	0.0000	0

Adjacency		Initial Penetration		Common Out	
i	j	P_d	T	P_d	T
23	17	0.0000	0	0.0000	0
24	35	0.0000	90		
24	42	0.0000	120		
24	6	.9500	480	0.0000	0
24	14	0.0000	0	0.0000	0
24	17	0.0000	0	0.0000	0
25	35	0.0000	120		
25	43	0.0000	30		
25	5	.9500	480	0.0000	0
25	8	.9500	150	0.0000	0
25	9	.9500	150	0.0000	0
25	17	0.0000	0	0.0000	0
26	35	0.0000	120		
26	44	0.0000	30		
26	5	.9500	480	0.0000	0
26	6	.9500	480	0.0000	0
26	9	.9500	150	0.0000	0
26	17	0.0000	0	0.0000	0
27	35	0.0000	120		
27	45	0.0000	30		
27	17	0.0000	0	0.0000	0
28	35	0.0000	120		
28	46	0.0000	30		
28	5	.9500	960	0.0000	0

Adjacency		Initial Penetration		Common Out	
i	j	Pd	T	Pd	T
28	15	0.0000	150	0.0000	0
28	17	0.0000	0	0.0000	0
29	35	1.0000	9999		
30	35	1.0000	9999		
31	35	1.0000	9999		
32	35	1.0000	9999		
33	35	1.0000	9999		
34	35	1.0000	9999		
36	18	0.0000	0	0.0000	0
37	19	0.0000	0	0.0000	0
38	20	0.0000	0	0.0000	0
39	21	0.0000	0	0.0000	0
40	22	0.0000	0	0.0000	0
41	23	0.0000	0	0.0000	0
42	24	0.0000	0	0.0000	0
43	25	0.0000	0	0.0000	0
44	26	0.0000	0	0.0000	0
45	27	0.0000	0	0.0000	0
46	28	0.0000	0	0.0000	0

Appendix C - Output for the Sabotage Case

```
N=  46,      PATHS FROM NODE   1 TO NODE  35 -

RESPONSE TIME =    330      INT RANGE =  0.0000000     DET RANGE =  0.0000000


MINIMUMS -    NODE          INTERRUPTION                    DETECTION
                 1           0.0000000                     1.0000000
                 2           0.0000000                      .6300000
                 3           0.0000000                      .9930000
                 4           0.0000000                     0.0000000
                 5           0.0000000                      .9900000
                 6           0.0000000                      .9900000
                 7           0.0000000                      .9995000
                 8           0.0000000                      .9995000
                 9           0.0000000                      .9995000
                10           0.0000000                      .9995000
                11            .9980000                      .9980000
                12            .9980000                      .9980000
                13           0.0000000                      .9995000
                14           0.0000000                      .9995000
                15           0.0000000                      .9980000
                16           0.0000000                      .9900000
                17           0.0000000                      .9995000
                18           0.0000000                      .9995000
                19            .9995000                      .9995000
                20            .9995000                      .9995000
                21            .9995000                      .9995000
                22            .9995000                      .9995000
                23            .9995000                      .9995000
                24           0.0000000                      .9995000
                25            .9995000                      .9995000
                26            .9995000                      .9995000
                27            .9995000                      .9995000
                28            .9980000                      .9995000
                29            .9995000                      .9995000
                30            .9995000                      .9995000
                31            .9995000                      .9995000
                32            .9995000                      .9995000
                33            .9980000                      .9995000
                34            .9980000                      .9900000
                35           0.0000000                      .9995000
                36            .9980000                      .9995000
                37            .9995000                      .9995000
                38            .9995000                      .9995000
                39            .9995000                      .9995000
                40            .9995000                      .9995000
                41            .9995000                      .9995000
                42           0.0000000                      .9995000
                43            .9995000                      .9995000
                44            .9995000                      .9995000
                45            .9995000                      .9995000
                46            .9995000                      .9995000
    2.36=TIME         141 BRANCHES


        PATH                        NODES                                              P(INTERRUPTION)     P(DETECTION)

          1                                                                                  .9995000          .9995000
                 1    4    5    6   17   19   35

          2                                                                                  .9995000          .9995000
                 1    4    5    6   17   20   35
```

```
 3                                                               .9995000    .9995000
        1   4   5   6  17  21  35
 4                                                               .9995000    .9995000
        1   4   5   6  17  22  35
 5                                                               .9995000    .9995000
        1   4   5   6  17  25  35
 6                                                               .9995000    .9995000
        1   4   5   6  17  26  35
 7                                                               .9995000    .9995000
        1   4   5   6  17  27  35
 8                                                               .9995000    .9995000
        1   4   5   6  17  28  35
 9                                                              0.0000000    .9999000
        1   4   5  14  24  35
10                                                               .9995000    .9995000
        1   4   5   6  17  23  35
11                                                               .9995000    .9995000
        1   4   5   6  17  24  35
12                                                               .9995000    .9995000
        1   4   5  17  19  35
13                                                               .9995000    .9995000
        1   4   5  17  20  35
14                                                               .9995000    .9995000
        1   4   5  17  18  35
15                                                               .9995000    .9995000
        1   4   5  17  21  35
16                                                               .9995000    .9995000
        1   4   5  17  22  35
17                                                               .9995000    .9995000
        1   4   5  17  25  35
18                                                               .9995000    .9995000
        1   4   5  17  26  35
19                                                               .9995000    .9999000
        1   4   5  17  27  35
20                                                               .9995000    .9995000
        1   4   5  17  28  35
21                                                               .9995000    .9995000
        1   4   5   6  17  18  35
22                                                               .9995000    .9999000
        1   4   5  17  23  35
23                                                               .9995000    .9995000
        1   4   5  17  24  35
```

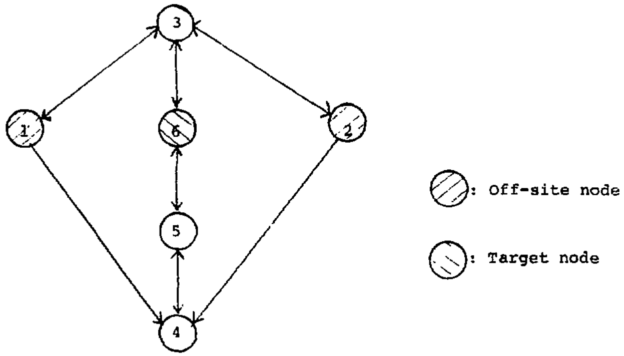
Appendix D - Output for the Theft Case

```
NO. 601      PATHS FROM NODE   1 TO NODE   1 -
             THROUGH TARGET NODE  34

RESPONSE TIME =   300      INT RANGE =  0.9000000      DET RANGE =  0.8000000


MINIMUMS -   NODE          INTERRUPTION                        DETECTION
                             IN            OUT             IN            OUT
               1          0.0000000      .9995000      0.0000000      .9995000
               2          0.0000000      .9995000      .8000000      .9995000
               3          0.0000000      .9995000      .9900000      .9999750
               4          0.0000000      .9995000      0.0000000      .9995000
               5          0.0000000      .9995000      .9900000      .9995000
               6          0.0000000      .9995000      .9900000      .9995000
               7          0.0000000      .9995000      .9995000      .9999750
               8          0.0000000      .9995000      .9995000      .9999750
               9          0.0000000      .9995000      .9995000      .9999750
              10          0.0000000      .9995000      .9995000      .9999750
              11           .9980000      .9995000      .9980000      .9995000
              12           .9980000      .9995000      .9980000      .9995000
              13          0.0000000      .9995000      .9995000      .9995000
              14          0.0000000      .9995000      .9995000      .9995000
              15          0.0000000      .9995000      .9980000      .9995000
              16          0.0000000      .9995000      .9980000      .9995000
              17          0.0000000      .9995000      .9995000      .9995000
              18          0.0000000      .9995000      .9995000      .9995000
              19           .9995000      .9995000      .9995000      .9995000
              20           .9995000      .9995000      .9995000      .9995000
              21           .9995000      .9995000      .9995000      .9995000
              22           .9995000      .9995000      .9995000      .9995000
              23           .9995000      .9995000      .9995000      .9995000
              24          0.0000000      .9995000      .9995000      .9995000
              25           .9995000      .9995000      .9995000      .9995000
              26           .9995000      .9995000      .9995000      .9995000
              27           .9995000      .9995000      .9995000      .9995000
              28           .9980000      .9995000      .9995000      .9995000
              29           .9995000      .9995000      .9995000      .9995000
              30           .9995000      .9995000      .9995000      .9995000
              31           .9995000      .9995000      .9995000      .9995000
              32           .9995000      .9995000      .9995000      .9995000
              33           .9980000      .9995000      .9995000      .9995000
              34           .9910000     1.0000000      .9910000     1.0000000
              35          0.0000000      .9995000      .9995000      .9995000
              36           .9910000      .9995000      .9995000      .9995000
              37           .9995000      .9995000      .9995000      .9995000
              38           .9995000      .9995000      .9995000      .9995000
              39           .9995000      .9995000      .9995000      .9995000
              40           .9995000      .9995000      .9995000      .9995000
              41           .9995000      .9995000      .9995000      .9995000
              42          0.0000000      .9995000      .9995000      .9995000
              43           .9955000      .9995000      .9995000      .9995000
              44           .9995000      .9995000      .9995000      .9995000
              45           .9995000      .9995000      .9995000      .9995000
              46           .9910000      .9995000      .9995000      .9995000

  0.71=TIME        418 BRANCHES


     PATH                              NODES                                        P(INTERRUPTION)     P(DETECTION)

       1                                                                                 .9995000          .9999750
                  1   4   5  17  20  34  20
                 17  18  13   5   4   1
```

```
 2                                                                         .9995000     .9999750
          1    4    5   17   20   38   20
         17   24   14    5    4    1

 3                                                                         .9995000     .9999750
          1    4    5   17   20   38   20
         17   23   14    5    4    1

 4                                                                         .9995000     .9999750
          1    4    5   17   20   38   20
         17   20   15    5    4    1

 5                                                                         .9995000     .9999750
          1    4    5   17   20   38   20
         17   12    5    4    1

 6                                                                         .9995000     .9995000
          1    4    5    6   17   20   38
         20   17    6    5    4    1

 7                                                                         .9995000     .9999750
          1    4    5    6   17   20   38
         20   17   18   13    5    4    1

 8                                                                         .9995000     .9999750
          1    4    5    6   17   20   38
         20   17   13    5    4    1

 9                                                                         .9995000     .9999750
          1    4    5    6   17   20   38
         20   17   24   14    5    4    1

10                                                                         .9995000     .9999750
          1    4    5   17   20   38   20
         17    6    5    4    1

11                                                                         .9995000     .9999750
          1    4    5    6   17   20   38
         20   17   20   15    5    4    1

12                                                                         .9995000     .9995000
          1    4    5   17   20   38   20
         17    5    4    1

13                                                                         .9995000     .9999750
          1    4    5    6   17   20   38
         20   17   23   14    5    4    1
```

Appendix E - Some Simplifying Assumptions Apparent in Applications of Polynomially Efficient Search Strategies to Similar Problems

The following graph is referred to in this Appendix.



It is desirable to employ graph-theoretic algorithms which are polynomial bounded. The execution time for such algorithms is proportional to some constant power of either the number of edges or the number of vertices in the graph. In this manner, solution of a given problem can be guaranteed within a polynomially bounded computer resource allocation. First, however, it must be determined that the solution to a problem is consistent with the polynomial bound.

Application of polynomially efficient (bounded) search strategies to path enumeration problems implies the existence of two conditions. First, the desired solution to the problem must be polynomially bounded. The shortest path length between two nodes implies a single value. However, identification of all paths which are characterized by this length is a combinatorial problem. Second, the iterative procedure used to arrive at intermediate representations of the problem must likewise have polynomial bounds. Again, the shortest distance to each intermediate node associates a single value with each node. However, the ordered sets of nodes (path history) contained in intermediate paths characterized by these values still imply combinatorial problem solutions.

The most popular algorithm which has been applied to shortest-path problems is probably the one proposed by Dijkstra.[1] Both of the foregoing conditions are present. The algorithm results in a quantification of the shortest path distance, and a set of labeled nodes which allows for reconstruction of paths. The purpose in writing the ENUMPTH program was to allow some of the assumptions made necessary

[1] Much discussion of this algorithm is found in the literature. For the basic treatise, see E. W. Dijkstra, "A Note on Two Problems in Connection with Graphs," Numer Math 1, 269-271, 1959.

by the two conditions to be relaxed. Following is a discussion of some examples of the types of analyses permitted by ENUMPTH, but which are inconsistent with the conditions stated above. Table I summarizes this discussion.

First, placing a polynomial bound on the problem solution is inappropriate for the objectives of some analyses of some problems. Unfortunately, precise estimates of the computational requirements for some problem solutions are not possible.[2] Nevertheless, as the examples included in this report demonstrate, many physical protection problems involving unrestricted solutions can be appropriately treated. The Dijkstra algorithm finds the shortest path distance between two nodes. Other examples of polynomially bounded solutions include the shortest distances between all pairs of nodes in a graph, a single minimal path, or a specified number of minimal paths. The physical protection system analyst is often concerned with identifying all ways in which a minimum path value is achieved. Additionally, he may want to know the next path value to insure that it is not within some trivial range of the minimum. Assuming arc weights equal to one in the graph example, the minimum distance from an off-site node to a target node is two. Associating this value with a single path, say (1, 3, 6), is only a partial solution. Path (2, 3, 6) is equally important in a complete set of critical paths. Any arbitrary restriction of such a solution results in incomplete information for analysis of the stated problem.

Next, some of the efficient approaches to path analysis do not recognize directed travel. A technique which has been suggested is to double arc weights and thereby derive an accumulation for theft paths. Aside from the fact that entry and exit paths may differ in a theft scenario, this technique does not allow arc weight to be a function of

---

[2] See D. E. Knuth, "Estimating the Efficiency of Backtrack Programs," <u>Math Comp</u> <u>29</u>, 121-136, 1975.

direction traveled. The lengths of paths (2, 4, 5, 6) and (6, 5, 4, 2) would, therefore, be identical. Placing a door which locks only from one direction between nodes 4 and 5 would create a situation which could not be modeled using this technique.

Finally, adequate treatment of theft paths also requires recognition of common outs. When penetration of a barrier is either partially or totally destructive, arc weights become a function of path history as well as direction traveled. No means of retaining path history has been identified which conforms to the polynomial limit imposed by the second condition stated above. As an example of a common out, assume that a concrete wall exists between nodes 1 and 3. Assume further that there is a moat between nodes 2 and 3. Disregarding bridging, penetrations of the wall are likely to be destructive. Moat crossings would not typically be destructive. Appropriate treatment of this model requires recognition that the wall must be defeated for the segment (3, 1) if and only if the segment (1, 3) is not included in the path history. The moat, however, must be defeated in the segment (3, 2) irregardless of path history.

Problem Characterization	ENUMPTH	Polynomially Efficient Strategy
1. a. Polynomially bounded solution.	Not applicable to problem.	Total solution offered.
1. b. All minimum paths or paths falling within a range of minimum.	Total solution offered.	Not applicable to problem.
2. Directed travel (direction dependent penetrations).	Total solution offered.	Partial solution offered.
3. Path history required (common out, partially or totally destructive penetrations).	Total solution offered.	Not applicable to problem.

Table 1. Comparison of Search Strategies

Distribution:

1000	G. A. Fowler
1230	W. L. Stevens
1233	R. E. Smith
1700	W. C. Myre
1710	V. E. Blake
1711	M. R. Madsen
1712	J. W. Kane
1716	R. L. Wilde
1730	C. H. Mauney
1733	T. J. Hoban
1739	J. D. Williams
1750	J. E. Stiegler
1754	J. F. Ney
1754	J. L. Todd
1754	S. T. Wallace
1758	C. E. Olson
1758	D. D. Boozer
1758	S. D. Chester
1758	G. H. Duke
1758	A. M. Fine
1758	L. A. Fjelseth
1758	K. D. Grant
1758	R. C. Hall (25)
1758	P. B. Herrington
1758	G. A. Kinemond
1758	W. W. Parker
1758	W. K. Paulus
1758	L. P. Robertson
1758	D. W. Stack
1758	L. H. Stradford
1758	R. B. Worrell
1759	M. J. Eaton
1760	J. Jacobs
1760A	M. N. Cravens
1761	T. A. Sellers
1761	A. E. Winblad
1761	J. L. Darby
1761	L. C. Nogales
1763	I. G. Waddoups
1765	D. S. Miyoshi
2620	R. J. Detry
2630	E. K. Montoya
2636	M. B. Moore
2637	D. A. Young
2640	T. L. Tischhauser
2650	A. D. Pepmueller
4410	D. J. McCloskey
4414	G. B. Varnado
4416	L. D. Chapman
5611	W. F. Roherty
5642	B. L. Hulme
8266	E. A. Aas (2)
8320	T. S. Gold
3141	T. L. Werner (5)
3151	W. L. Garner (3) For DOE/TIC (Unlimited Release)
DOE/TIC (25)	(R. P. Campbell, 2172-3)