27
3-16-82
J-1954
①
SANDIA REPORT

SAND81-1261 • Unlimited Release • UC-32

Printed December 1981

MASTER

FIRE: A Subroutine for Fire- Protection Network Analysis

NOTICE

**PORTIONS OF THIS REPORT ARE ILLEGIBLE. It
has been reproduced from the best available
copy to permit the broadest possible avail-
ability.**

Bernie Hulme, Ann W. Shiver, Peter J. Slater

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof or any of their contractors or subcontractors.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

SAND--81-1261

DE82 008620

SAND81-1261
Unlimited Release

FIRE: A SUBROUTINE FOR FIRE PROTECTION NETWORK ANALYSIS

Bernie Hulme
Numerical Mathematics Division, 5642

Ann W. Shiver
Nuclear Facility Analysis Division, 4414

Peter J. Slater
Applied Mathematics Division, 5641

Sandia National Laboratories
Albuquerque, New Mexico 87185

ABSTRACT

FIRE is a FORTRAN subroutine designed to compute fire protection sets in facility fire spread networks. The input consists of (a) a network in which the vertices are fire areas and the arcs are fire barriers between physically adjacent areas, (b) a collection of critical combinations of equipment locations called target vertices, and (c) costs for protecting the individual areas from fire and for preventing fire spread through the barriers. FIRE computes a relatively inexpensive subset of areas and barriers, called a fire protection set, which, if protected from fire, will guarantee that no single-source fire can destroy any critical equipment combination. Because it will compute only some of the many possible fire protection sets, FIRE produces solutions that may not have the minimum possible cost.

TABLE OF CONTENTS

	Page
1. Introduction	3
2. The Facility Network Model	5
3. The Fire Protection Problem	8
4. An Example	9
5. The Heuristic Approach of FIRE	11
5.1 The Computational Flow Network	12
5.2 Minimum Cuts in a Flow Network	13
5.3 An Outline of FIRE and How to View its Results	13
5.4 The Problem Tree	15
6. Calling FIRE	19
7. Solving Some Examples with FIRE	23
8. A Description of the FIRE Algorithm	32
9. Subroutine Performance	36

This book was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

1. Introduction

The subroutine FIRE is designed for use by fire protection engineers who wish to find economical ways to eliminate vulnerabilities to fire in a facility. While this work was motivated by the desire to study nuclear power plants, where the objective is to avoid any possible radioactive release due to the consequences of fire, use of the method is not restricted to such plants. Any facility may be represented by a network in which the vertices are fire areas and the directed arcs are ways for fire to spread from one area to another. If certain sets of components, located at some of the vertices, are critical in the sense that destruction of all the components in one set can have unacceptable consequences, then it is important to know if all of the critical sets can be protected from fire by protecting individual arcs and vertices. That is, can the entire facility be protected by judiciously selecting certain fire areas and/or barriers for increased fire protection? A moment's thought indicates that one can always try to protect all of the components in all of the critical sets, but just as quickly one suspects this might be prohibitively expensive. Thus arises the notion of seeking to protect the facility in some relatively economical way, perhaps in some minimum-cost way.

This work grew out of an earlier study [1] by Dennis Berry, who proposed a method of deriving the adjacencies in a fire spread network based on conservatively predicting barrier failures whenever a ventilation-limited combustion calculation

indicates that the fire exposure can exceed the fire rating of a barrier. In addition, Berry used a Boolean algebraic technique to derive all the ways of protecting a facility by protecting minimal sets of critical areas. The present computational method expands on Berry's ideas by identifying ways to protect a facility by protecting any combination of arcs and vertices, not just vertices where critical components are located. Also, the present objective is to do this in such a way as to minimize the sum of the costs of protecting the arcs and vertices, rather than protecting the fewest vertices.

This study deals only with single-source fires, multiple-source fires being much more complicated and of less concern in the context of accidental fire. Also, FIRE only achieves a heuristic solution to the optimization problem. All solutions produced by FIRE are guaranteed to protect the facility against single-source fires. The costs of FIRE's solutions may not be the minimum possible. They will, however, be upper bounds on the minimum cost, and they will be the result of careful attempts to minimize cost over a variety of possibilities using network flow theory. Used alone, then, FIRE cannot necessarily find minimum-cost solutions, and the user is provided only with the assurance that the computed solution is the best among several but without the assurance that the cost is anywhere near minimum.

Fortunately the authors have developed an iterative method for obtaining optimum solutions to this problem [6]. The FORTRAN subroutine FIRE will be used to obtain feasible solutions

whose costs are upper bounds on the minimum cost. An interpretive SETS [9] user program will use Boolean and numerical techniques to obtain lower bounds. Since the iterative method will be coded in the Cyber Control Language, available only on CDC computers, it was decided that the heuristic, but portable, FORTRAN subroutine FIRE should be documented separately.

2. The Facility Network Model

The facility model is a network consisting of a set of vertices and a set of directed arcs between certain pairs of vertices. A fire protection analyst can determine the vertex set from the facility blueprints together with knowledge of the fire ratings of the walls, floors, and ceilings in the facility. One vertex is assigned to each fire area, where a fire area may be a single room or perhaps a cluster of rooms the partitions of which offer little resistance to fire. The vertices are numbered 1, 2, The arc set is determined by first identifying which fire areas are physically adjacent and then doing combustion calculations for each fire area to determine whether the fuel load, ventilation rate, room configuration, etc., could support a fire sufficient to penetrate the surrounding walls, ceilings, or floors and to propagate into a physically adjacent fire area. For each pair of physically adjacent vertices, u and v , the analyst must define a pair of directed arcs (u,v) and (v,u) , if fire can spread from u to v and from v to u . If fire can spread only in one of the directions, both arcs are still needed in the model; they will be weighted differently. If fire can

spread in neither direction, the arcs (u,v) and (v,u) should not be included in the network.

Development of predictive combustion models is a topic of current research. The fire spread network for a facility will therefore depend greatly upon the combustion model chosen by the analyst and upon the assumptions employed. It is essential that an analyst use these models and assumptions uniformly when defining the facility network for input to FIRE, and that he keep them clearly in mind when interpreting the output from FIRE.

The vertices where vital components are located are called target vertices. A minimal set of target vertices corresponding to a critical set of components is called a target set. In a nuclear power plant, for example, destruction of a target set can lead to a nuclear release. Being minimal, one target set cannot completely contain another target set. The collection of target sets of concern is typically derived from fault tree analysis [10]. Since all of the vertices in a target set must be destroyed in order to initiate the undesired event, protecting one target vertex is sufficient to protect any target set of which it is a member.

The fire protection analyst must also assign nonnegative weights to each arc and vertex in the facility network. The weights will be thought of as costs for fire protection, although any nonnegative measure which one wants to minimize can be used. The vertex weights are the costs of both preventing the spread of fire through a fire area and preventing damage to any

vital components located there. There are two kinds of arc weights, individual and joint. The individual arc weight for arc (u,v) is the cost of preventing fire from penetrating the barrier from vertex u to vertex v . The arc weight for (v,u) may be different from that of (u,v) . The joint arc weight for the pair (u,v) and (v,u) is the cost of protecting both arcs. Depending upon the specific protection methods envisioned by the analyst who prepares the input data for FIRE, the joint arc weight will be a number between the larger of the two individual arc weights and their sum. If protecting the more expensive arc also protects the less expensive one, the joint weight is the larger of the individual weights. If protecting both arcs requires separate expenses, then the joint weight may be as large as the sum of the individual weights.

A zero weight indicates that a vertex, an arc, or an arc pair is already protected against fire and no additional expense is necessary for fire protection. If fire can spread from u to v , but not from v to u , then arc (u,v) has a positive weight equal to the cost of preventing this spread, arc (v,u) has a zero weight, and their joint weight is equal to the weight of (u,v) . There are times when it is more convenient to place zero weights on a vertex or an arc pair (individual and joint) than to delete the vertices or the arcs. This is perfectly acceptable and has the same effect as deletion (except on storage space).

3. The Fire Protection Problem

A fire protection set in a network is any collection of arcs and vertices which, when deleted (protected), produces a network not having fire paths from a single source vertex to all the vertices of any target set. The cost of a fire protection set is the sum of the weights of its elements, using the joint arc weight when a pair of arcs (u,v) and (v,u) are both in the set. The minimum-cost fire protection problem can be stated as follows. Given (a) a network $\mathcal{N}$ consisting of a digraph $D(V,A)$ with vertex set $V = \{1,2,\dots,NV\}$ and arc set A together with vertex weights, individual arc weights and joint weights for each arc pair (u,v) and (v,u) , and (b) a collection $\mathcal{T}$ of target sets $T_1, T_2, \dots, T_k$, no one of which completely contains another, find a minimum-cost fire protection set for $(\mathcal{N}, \mathcal{T})$.

This is the problem FIRE attempts to solve. Although FIRE always achieves the primary objective of finding a fire protection set, it may not achieve the secondary objective of minimizing cost. This is because FIRE only computes some of the possible fire protection sets and reports one of them that has the smallest cost among those it computed. If several computed protection sets have this same cost, only the first one encountered is reported. Users who want more than one feasible solution can obtain all of the fire protection sets which FIRE considered by using a special output option explained later.

Every fire protection set that FIRE considers is guaranteed to be minimal, meaning it does not contain some other fire protection set. This property is important because it implies

that whenever a facility is already protected against fire, and hence the optimum solution is an empty fire protection set having a cost of zero, FIRE is guaranteed to produce this optimum result. When the minimum-cost for fire protection is strictly positive, however, FIRE can produce solutions whose costs are greater than the minimum.

4. An Example

Figure 1 shows a fire spread network having 8 vertices and 20 arcs. The joint arc weights are enclosed in parentheses, e.g., both (5,6) and (6,5) can be

Figure 1.

A Fire Spread Network $\mathcal{N}_1$

protected for a total cost of 4. Let the collection $\mathcal{T}_1$ of target sets be

$$\mathcal{T}_1 = \{T_1, T_2\}$$

$$T_1 = \{1, 3, 5\} \quad T_2 = \{3, 5, 6\}.$$

There are many fire protection sets for $(\mathcal{N}_1, \mathcal{T}_1)$. The most obvious one is $FP_1 = \{1, 3, 5, 6\}$, the set of all target vertices, having a cost of 200. This is far more expensive than necessary,

because $FP_2 = \{3,5\}$ is also a fire protection set costing only 100. To see this, just imagine that fire cannot flow into or out of 3 and 5, i.e., that 3 and 5 have been protected at an expense of 100. No matter where a single-source fire starts, it cannot destroy T_1 because 3 and 5 are protected. Neither can it destroy T_2 for the same reason. Actually 100. is too much to spend also. $FP_3 = \{3\}$ costs only 50., and it is a fire protection set too. If 3 were protected, neither T_1 nor T_2 could be destroyed by a single-source fire. It is true that a fire could start at 6 and spread to 5 as well as through 4 and 2 to 1, but that is not enough. Also, a fire could start in 3, but it could not leave 3 if 3 were protected.

Because the vertex weights are relatively large in this example, FP_3 is still too expensive. Consider the fire protection set $FP_4 = \{(1,3), (3,1); (3,4), (4,3); (5,6), (6,5); (6,8), (8,6)\}$. Summing the joint weights of these arc pairs shows that FP_4 has a cost of $9.+8.+4.+8. = 29$. Clearly, if all of these arcs were deleted from the network, there would be no fire paths from any single vertex to T_1 or to T_2 because the network would have been cut into two pieces with 1 and 6 in one piece and 3 and 5 in the other. A single-source fire could destroy one, but not both, pieces.

The latter observation is the basis for the method in FIRE: try to find a relatively inexpensive set of arcs and vertices whose deletion splits the network into enough pieces that every target set has vertices in at least two different pieces. This should make clear the restriction $|T_i| \geq 2$, for all i . If any

T_1 consists of only one target vertex, that vertex must be included in any fire protection set. Therefore, the analyst must supply FIRE only with the target sets having two or more elements and remember to unite the answer from FIRE with every singleton target that he identified in order to form a complete solution.

Consider now some other ways to split the network so that T_1 and T_2 have elements in different pieces. $FP_5 = \{(1,3); (3,4), (4,3); (3,5), (5,3)\}$, consisting of two arc pairs and an individual arc, has a cost of $5.+8.+9. = 22$. FP_5 is a fire protection set because its protection essentially isolates vertex 3. A fire outside 3 could not reach 3, and a fire inside 3 could not propagate beyond vertex 1. In neither case would T_1 or T_2 be destroyed. A better splitting, however, is achieved by $FP_6 = \{(3,5), (5,3); (5,6), (6,5); (6,8), (8,6)\}$ having a cost of $9.+4.+8. = 21$.

The optimum solution, one that not only protects all the target sets but does so at the minimum possible cost, is $FP_7 = \{(3,5), (5,3); (4,6), (6,4)\}$ having a cost of $9.+8. = 17$. Each of these arcs is absolutely essential. For example, if the first three arcs were protected, but $(6,4)$ were not, a fire at 5 could spread to 6 and 3, destroying T_2 . Similarly for each of the other arcs.

5. The Heuristic Approach of FIRE

In order to prepare the input for FIRE a user must know a little about what FIRE does. Briefly, FIRE computes the

solutions to a sequence of "minimum cut problems" specified by the user. These minimum cuts are united and modified in appropriate ways to form minimal fire protection sets, one of the least expensive of which is saved for output. (All the minimal fire protection sets that were computed are available to the user through a special output option.)

5.1 The Computational Flow Network

Before we define a minimum cut in a network, let us define the network in which FIRE seeks minimum cuts. FIRE converts a given fire spread network $\mathcal{N}$ into an associated computational flow network $\mathcal{N}'$ by (a) replacing every vertex k by the two vertices k' and k'' and the single arc (k', k'') , (b) replacing all arcs (i, j) by (i'', j') , (c) putting the vertex weight of k on the arc (k', k'') , and (d) putting the joint arc weight of (i, j) and (j, i) on (i'', j') and (j'', i') . The network $\mathcal{N}_1$ in Figure 1 transforms into the network $\mathcal{N}_1'$ of Figure 2. Only the arcs of $\mathcal{N}_1'$ are weighted, in conformity with standard network flow theory, yet it is easy to relate sets of arcs in $\mathcal{N}_1'$ to sets of arcs and vertices in $\mathcal{N}_1$, and vice-versa.

Figure 2.

Computational Network $\mathcal{N}_1'$ Associated with $\mathcal{N}_1$

5.2 Minimum Cuts in a Flow Network

Let N be any arc-weighted network with two distinguished subsets of vertices, X and Y , assumed to be disjoint and nonempty. The vertices in X are the sources and those in Y are the sinks. The remaining vertices are called intermediate vertices. If the weight of arc (i,j) were thought of as a maximum rate at which a commodity could flow through (i,j) , then one could imagine a flow through the network from X to Y such that the balance of flow at each intermediate vertex is zero and the flow in each arc is no greater than its arc weight.

Given subsets of vertices, S and T , we denote by (S,T) the set of arcs from a vertex in S to a vertex in T . An X - Y cut in N is a set of arcs of the form $(S, \bar{S})$ where S contains X and $\bar{S}$ (the set of vertices not in S) contains Y . The capacity of a cut is the sum of the weights (capacities) of the arcs in the cut. A cut in N is a minimum cut if no other cut in N has a smaller capacity. The max-flow min-cut theorem of Ford and Fulkerson [4] states that the maximum flow through a network is equal to the capacity of a minimum cut.

5.3 An Outline of FIRE and How to View its Results

FIRE uses an efficient minimum cut algorithm due to Dinic [3] to find minimum cuts in the flow network $\mathcal{N}'$. The cut problems solved by FIRE are specified by the user. For the j -th problem to be solved the user gives a set of sources X_j and a set of sinks Y_j in the fire spread network $\mathcal{N}$. FIRE converts $\mathcal{N}$ into the computational flow network $\mathcal{N}'$. Then, for each given problem, FIRE translates X_j and Y_j into sources

$x_j' = \{k' | k \in x_j\}$ and sinks $y_j' = \{k'' | k \in y_j\}$ in $\mathcal{N}'$ and computes a minimum $x_j' - y_j'$ cut in $\mathcal{N}'$. When its sources contain none of the T_i , a least-capacity cut among these translates back to a relatively economical, but not necessarily minimum-cost, fire protection set in $\mathcal{N}$.

An arc in a cut, although oriented with the flow in $\mathcal{N}'$, is either (a) (k', k'') representing the protection of vertex k in $\mathcal{N}$, or (b) (i'', j') representing the protection of the pair of arcs (i, j) and (j, i) in $\mathcal{N}$. Notice that because $\mathcal{N}'$ was constructed so that (i'', j') and (j'', i') have the same capacity, namely the joint weight of (i, j) and (j, i) , $\mathcal{N}'$ is symmetric with respect to maximum flows and minimum cuts. That is, a minimum $x_j' - y_j'$ cut becomes a minimum $y_j' - x_j'$ cut if every arc (i'', j') is replaced by its partner (j'', i') . This means that any of the minimum cuts in $\mathcal{N}'$ will correspond to sets of vertices and arc pairs in $\mathcal{N}$ that will prevent fire propagation in both directions in $\mathcal{N}$, an essential attribute of a fire protection set. The peculiar use of the joint arc weights in $\mathcal{N}'$ has the very purpose of allowing a minimum cut calculation, which is inherently "one-way," to find a pretty good "two-way" cut for the original physical problem.

An $x_j' - y_j'$ cut which translates into a fire protection set may have certain arcs in $\mathcal{N}'$ that translate into arc pairs in $\mathcal{N}$, and sometimes only one arc in a pair really needs protection. To the extent that the joint weight exceeds the essential individual arc weight, the corresponding fire protection cost is larger than necessary. This is one way in which FIRE can

fail to achieve minimum cost. Another way is when a source set X_j contains some of the target sets T_i , forcing the user to specify a subproblem $j+1$ in which certain vertices of X_j are taken as sources X_{j+1} and others as sinks Y_{j+1} and a $j+1$ -st cut on the "source side" of $\mathcal{N}' - \{X_j' - Y_j' \text{ cut}\}$ is required in order to split up those target sets contained in X_j . FIRE forms the union of the j -th and $(j+1)$ -st minimum cuts, adding their costs. Assuming that X_{j+1} contains none of its target sets, this union cuts $\mathcal{N}'$ into three pieces and yields a fire protection set in $\mathcal{N}$. However, there is no assurance that this union is the minimum-cost splitting of $\mathcal{N}'$ into three pieces or that the corresponding fire protection set has minimum cost.

5.4 The Problem Tree

Having introduced the user to FIRE, we conclude this section with a recommended way to pose minimum-cut problems for FIRE to solve. The user has great flexibility here. He may add to the list cut problems which occur to him but which are not indicated by our recommendation. He may also delete problems. Generally the more cuts FIRE computes the more opportunities there are to find lower cost solutions.

In order to pose a list of problems, some of which are subproblems of others, the user should develop a problem tree whose vertices represent minimum-cut problems. Source and sink sets of target vertices must be associated with each vertex except the root, which corresponds to the main problem. The main problem is decomposed into subproblems each represented by a vertex which is a descendant of the root vertex. These

vertices are said to be on the "first level" of the problem tree. Some of the first-level problems may have subproblems represented by descendant vertices on the second level. These vertices can also have descendants, and therefore a problem tree can have several levels. The endpoints in the tree are problems that have no subproblems. The vertices are numbered in preorder starting with zero for the root. This means that in a drawing of the tree the leftmost descendant is 1, its leftmost descendant is 2, etc. At an endpoint, backup one level and number the leftmost unnumbered descendant, then its leftmost descendant, etc.

The best way we know to select source and sink sets requires the notion of a minimal representing set. Let C be a collection of nonempty, proper subsets of some set U , i.e., $C = \{C_1, C_2, \dots, C_m\}$, $\emptyset \neq C_i \subset U$. A subset $R \subseteq U$ having at least one element from every C_i is a representing set of C . A representing set which contains no other representing set is a minimal representing set.

There is a purely algebraic way to find all the minimal representing sets of any collection. It is a Boolean algebraic method and hence can be carried out with SETS [9]. Let's examine the collection $\mathcal{T}_1 = \{T_1, T_2\}$ of target sets $T_1 = \{1, 3, 5\}$ and $T_2 = \{3, 5, 6\}$. For a moment view the target vertex numbers in T_1 and T_2 as the names of Boolean variables. Form the product

$$F = (1+3+5) \cdot (3+5+6),$$

where $+$ and $\cdot$ are Boolean addition (OR) and multiplication (AND), respectively. Expanding the product and simplifying by means of idempotence, $a \cdot a = a$, and absorption $a + a \cdot b = a$, yields

$$F = 3+5+1 \cdot 6 . \quad (1)$$

Each term of F is a minimal representing set of $\mathcal{T}_1$, and there are no others.

We recommend that the first-level vertices in a problem tree correspond to cut problems whose sources are minimal representing sets of $\mathcal{T} = \{T_1, T_2, \dots, T_k\}$, the collection of target sets to be protected. For each such source set X_j , the corresponding sink set should be a minimal representing set of the collection $\mathcal{T}-X_j = \{T_1-X_j, T_2-X_j, \dots, T_k-X_j\}$. Generally $\mathcal{T}-X_j$ has several minimal representing sets, and hence the problem tree can have problems on the same level with the same source set and different sink sets. Some of the T_i-X_j may be empty, indicating the need for a subproblem, but such empty sets are simply ignored in finding minimal representing sets of $\mathcal{T}-X_j$.

To illustrate the above ideas let's first draw the problem tree for $(\mathcal{N}_1, \mathcal{T}_1)$. From (1) we see that the first problem should have source set $X_1 = \{3\}$. The sink set Y_1 should be a minimal representing set of $\{T_1-X_1, T_2-X_1\} = \{\{1,5\}, \{5,6\}\}$, i.e., a term of

$$(1+5) \cdot (5+6) = 5+1 \cdot 6 . \quad (2)$$

This means we can select $Y_1 = \{5\}$ for the first problem. Since X_1 does not contain any T_i , or equivalently all $T_i-X_1 \neq \emptyset$, problem 1 has no subproblem and the problem 1 vertex has no descendants. For problem 2 we use the last term in (2) to define $X_2 = \{3\}$, $Y_2 = \{1,6\}$. Having coupled the first term of (1) with every term in (2), we use the second term in (1) to define

$x_3 = \{5\}$. Since $\{T_1 - x_3, T_2 - x_3\} = \{\{1,3\}, \{3,6\}\}$, we have the terms of

$$(1+3) \cdot (3+6) = 3+1 \cdot 6 \quad (3)$$

as possible sink sets. So we let $Y_3 = \{1,6\}$ and forget about $\{3\}$ because $\{3\}$ and $\{5\}$ already appear as source and sink in problem 1. Since $\{1,6\}$ has already been paired with $\{3\}$ and $\{5\}$ in previous problems, a problem tree for $(\mathcal{N}_1, \mathcal{T}_1)$ has been constructed.

Figure 3.

A Problem Tree for $(\mathcal{N}_1, \mathcal{T}_1)$

For each first-level descendant of vertex 0, FIRE computes a minimal fire protection set. A best one is returned to the user. The results of these three computations have already been discussed in Section 4. Problem 1 yields fire protection set FP_7 with cost 17.; problem 2 yields FP_5 with cost 22.; and problem 3 yields FP_6 having cost 21. In this case FIRE reports an optimum solution, FP_7 .

Suppose some first-level source set, say X_j , contains some of the T_i 's, say T_2, T_4 , and T_5 . This would be detected during the selection of the sinks Y_j because $T_2 - X_j = T_4 - X_j = T_5 - X_j = \emptyset$.

The tree must be continued below vertex j because splitting X_j from Y_j will not split T_2 , T_4 , or T_5 . The "source side" of $\mathcal{N}' - (X_j' - Y_j' \text{ cut})$ must be further split so that T_2 , T_4 , and T_5 will be protected. The user must create at least one subproblem for problem j . This is done in the same way that subproblems of problem 0 were created. Here vertex j plays the same role as vertex 0. The collection of T_i 's is now $\{T_2, T_4, T_5\}$, the target sets in X_j that need protecting. The descendants of vertex j have source sets that are minimal representing sets of $\{T_2, T_4, T_5\}$. If one of these source sets contains T_2 , T_4 , or T_5 , then the corresponding vertex must have descendants, and so on. The growth of the problem tree stops when every endpoint has a source set not containing any of its associated target sets. Section 7 contains an example of a problem tree with more than one level.

Pairing each source set X_j with every minimal representing set of $\mathcal{T} - X_j = \{T_1 - X_j, T_2 - X_j, \dots, T_k - X_j\}$ as sinks can lead to a very large problem tree. In this event the user may want to exercise his prerogative to trim the problem tree. One way to do this is to pair each source set X_j with only one minimal representing set of $\mathcal{T} - X_j$, perhaps one of minimum cardinality. The user thereby saves on input preparation time and FIRE execution time at the expense of possibly overlooking a better fire protection set.

6. Calling FIRE

The call list for FIRE is

FIRE (NV, NA, VWT, AWT, AWTJ, JNDX, IAWPTR, NTVERT, NTSETS,

LTSETS, LTPTR, NPRB, ISUB, LSRC, LSRPTR, LSNK, LSNPTR,
 NNETS, LAFDIM, LVFDIM, LNADIM, IACDIM, IDCDIM, ISWDIM,
 IRWDIM, IWDIM, IWSLN, IWNET, NAF, NVF, LARCSF, LVERTF,
 WTSUM, IERR, RWRK, IWRK).

The input variables are NV, ..., IWNET; output variables are NAF, ..., IERR; and RWRK and IWRK are work arrays. Only the scalar input variables NV, NA, NTVERT, NTSETS, NPRB, NNETS, IWSLN, and IWNET may be called by value. All other variables must be called by name.

The input consists mainly of the weighted digraph D, the target sets and the problem tree. The output consists of one fire protection set passed through the call list plus optional output on tape.

Input

- NV - number of vertices in D;
- NA - number of arcs in D [if (i,j) is in D, then (j,i) must be also];
- VWT - a real array of vertex weights, dimensioned NV;
- AWT - a real array of individual arc weights for the arcs from vertex 1 in any order, then the arcs from vertex 2 in any order, ..., the arcs from vertex NV in any order, dimensioned NA;
- AWTJ - a real array of joint arc weights for the same arc ordering as in AWT [pairs (i,j) and (j,i) have the same joint weight], dimensioned NA;
- JNDX - an integer array of the vertex numbers at the ends of the arcs taken in the same order as in AWT and AWTJ, dimensioned NA;
- IAWPTR - an integer array of pointers to the last entries in AWT, AWTJ and JNDX for the arcs out of each vertex, dimensioned NV;
- NTVERT - number of target vertices;

NTSETS - number of target sets;
 LTSETS - an integer array listing the vertices of each target set, dimensioned LTPTR (NTSETS);
 LTPTR - an integer array of pointers to the last entry in LTSETS for each target set, dimensioned NTSETS;
 NPRB - number of cut problems (highest numbered vertex in the problem tree);
 ISUB - an integer array of parent vertex numbers for each subproblem of the problem tree, dimensioned NPRB;
 LSRC - an integer array listing the sources for each cut problem, dimensioned LSRPTR (NPRB);
 LSRPTR - an integer array of pointers to the last entry in LSRC for each problem, dimensioned NPRB;
 LSNK - an integer array listing the sinks for each cut problem, dimensioned LSNPRT (NPRB);
 LSNPTR - an integer array of pointers to the last entry in LSNK for each problem, dimensioned NPRB;
 NNETS - number of flow networks stored at once (number of levels in the problem tree);

The next six variables are dimensions of arrays whose lengths cannot be known in advance. Reasonably estimated values are suggested in parentheses. If any of these input values proves to be too small, FIRE will return with IERR = 8 and newly estimated values stored in those variables that were too small. See IERR = 8 below.

LAFDIM - second dimension of LARSCF (try twice the number of arcs in a two-way cut of D);
 LVFDIM - dimension of LVERTF (try twice the number of vertices in a cut of D);
 LNADIM - dimension of three subarrays in RWRK or IWRK [try $2(N_A + N_V + N_{TVERT})$];
 IACDIM - dimension of four subarrays in IWRK (try $LAFDIM/2 + LVFDIM$);

IDCDIM - dimension of five subarrays in RWRK or IWRK (try
 LAFDIM + LVFDIM);

ISWDIM - dimension of two subarrays in RWRK or IWRK (try
 LAFDIM + number of individual arcs weighted zero
 + $(2NA/NV)(LVFDIM + \text{number of vertices weighted zero})$;

IRWDIM - dimension of RWRK
 = $LNADIM + (NA + NV + NTVERT) + NV + 2NNETS - 1 +$
 IDCDIM + ISWDIM;

IWDIM - dimension of IWRK
 = $2 LNADIM + (NA + NV + NTVERT) + 18NV + 4NNETS + 16$
 + $4IACDIM + 5IDCDIM + ISWDIM$;

IWSLN - write-solution indicator
 = 0, do not write any solutions,
 > 0, write on tape IWSLN all the solutions computed
 by FIRE;

IWNET - write-subnetwork indicator
 = 0, do not compute and write any subnetworks,
 > 0, compute one subnetwork associated with each
 arc and each vertex of the fire protection
 set in LARCSF and LVERTF and write these sub-
 networks on tape IWNET.

Output

NAF - number of arcs in the output fire protection set
 (stored in LARCSF);

NVF - number of vertices in the output fire protection
 set (stored in LVERTF);

LARCSF - list of arcs in the output fire protection set,
 stored as ordered pairs of vertex numbers,
 dimensioned (2, LAFDIM);

LVERTF - list of vertices in the output fire protection set,
 dimensioned LVFDIM;

WTSUM - weight sum (cost) of the output fire protection
 set;

IERR - error indicator
 = 0, no errors; all output is meaningful;
 = 1, 2, ..., 7, input error; above output is undefined
 and there is no tape output;
 = 8, one of the input dimensions, LAFDIM, ..., IWDIM
 is too small; above output is undefined and there
 is no tape output; some of these dimensions

have been reset to new estimates; redimension LARCSF, LVERTF, RWRK, IWRK if their dimensions have been reset and rerun the problem;

LAFDIM, ..., IWDIM = minimum necessary values, if IERR = 0,
 = input values, if IERR = 1, 2, ..., 7,
 = new estimates, if IERR = 8.

Work Arrays

RWRK - a real work array, dimensioned IRWDIM;

IWRK - an integer work array, dimensioned IWDIM.

7. Solving Some Examples with FIRE

First Example

First let us solve the problem for network $\mathcal{N}_1$ of Figure 1 with target sets $\mathcal{T}_1$ and the problem tree of Figure 3. There are 8 vertices and 20 arcs, so

$$NV = 8 \quad NA = 20.$$

The weights are given by

$$VWT = (50., 50., 50., 50., 50., 50., 50., 50.)$$

$$AWT = (0., 5. | 6., 3. | 4., 8., 7. | 3., 8., 7. | 2., 4., 5. | 6., 4., 8. | 5., 6. | 8., 3.)$$

$$AWTJ = (6., 9. | 6., 3. | 9., 8., 9. | 3., 8., 8. | 9., 4., 10. | 8., 4., 8. | 10., 9. | 8., 9.)$$

$$JNDX = (2, 3 | 1, 4 | 1, 4, 5 | 2, 3, 6 | 3, 6, 7 | 4, 5, 8 | 5, 8 | 6, 7)$$

$$IAWPTR = (2, 4, 7, 10, 13, 16, 18, 20).$$

The target sets $\mathcal{T}_1 = \{\{1, 3, 5\}, \{3, 5, 6\}\}$ are specified by

$$NTVERT = 4 \quad NTSETS = 2$$

$$LTSETS = (1, 3, 5 | 3, 5, 6) \quad LTPTR = (3, 6).$$

The problem tree in Figure 3 is defined by

$$NPRB = 3 \quad ISUB = (0, 0, 0)$$

$$LSRC = (3 | 3 | 5) \quad LSRPTR = (1, 2, 3)$$

LSNK = (5|1,6|1,6)

LSNPTR = (1,3,5)

NNETS = 1.

The dimension estimates are based on the user's inspection of the network. A two-way cut of the network would involve no more than about 8 arcs (a horizontal cut in Figure 1). Only about 2 vertices would be needed to cut the network. Therefore, following the formulas recommended in Section 6, we have

LAFDIM = $2 \cdot 8 = 16$

LVFDIM = $2 \cdot 2 = 4$

LNADIM = $2(20+8+4) = 64$

IACDIM = $16/2 + 4 = 12$

IDCDIM = $16 + 4 = 20$

ISWDIM = $16 + 1 + 5(4+0) = 37$

IRWDIM = $64 + 32 + 8 + 2 \cdot 1 - 1 + 20 + 37 = 162$

IWDIM = $2 \cdot 64 + 32 + 18 \cdot 8 + 4 \cdot 1 + 16 + 4 \cdot 12 + 5 \cdot 20 + 37 = 509$.

Let us also inquire about all three solution candidates by setting

IWSLN = 7

and about the subnetworks protected by the solution returned through the call list by setting

IWNET = 7.

This will cause both kinds of optional output to appear on Tape 7.

Remembering to dimension all the arrays with the above dimension data and to declare Tape 7 in the PROGRAM statement as well as to set all of the input variables as given above, we may call FIRE and obtain the following output. Through the call list we obtain

IERR = 0,

which means there are no errors and it makes sense to read the other output, and

$$\text{NAF} = 4 \quad \text{NVF} = 0$$

$$\text{LARCSF} = \begin{bmatrix} 3 & 4 & 5 & 6 & \dots \\ 5 & 6 & 3 & 4 & \dots \end{bmatrix} \quad \text{LVERTF} = \emptyset$$

$$\text{WTSUM} = 17.$$

A best solution that FIRE found consists of 4 arcs and no vertices. The arcs are the first 4 columns in the array LARCSF, $\{(3,5), (4,6), (5,3), (6,4)\}$, and the total cost of this fire protection set is 17.

On Tape 7 are written all three solution candidates:

$$(1) \quad \text{COST} = 17.00$$

ARCS

3	5
4	6
5	3
6	4

VERTICES

$$(2) \quad \text{COST} = 22.00$$

ARCS

3	5
1	3
5	3
3	4
4	3

VERTICES

$$(3) \quad \text{COST} = 21.00$$

ARCS

3	5
5	3
5	6
6	5
8	6
6	8

VERTICES

These are followed by four subnetworks protected by the four arcs of the least-cost solution computed:

(1) SOURCE = 1

TARGETS = 1 3 5

SUBNETWORK = 1 3 1 5 3

(2) SOURCE = 5

TARGETS = 1 3 5

SUBNETWORK = 1 3 5 3 5

(3) SOURCE = 1

TARGETS = 1 3 5

SUBNETWORK = 1 3 1 5 6 4 3

(4) SOURCE = 5

TARGETS = 1 3 5

SUBNETWORK = 1 2 4 6 5 3 4 5

Each subnetwork is a tree rooted at the source and having the targets as endpoints. The vertex list defines the subnetwork by listing the first target, its predecessor, the predecessor's predecessor, etc., until the source is listed; the second target, its predecessor, ..., until a vertex is listed that has already been listed; the third target, ...; the last target,

As mentioned earlier, FIRE happens to yield an optimum solution to this problem. But consider what would have happened if the first vertex had been omitted from the problem tree. FIRE would have computed only the last two candidates and would have reported the fire protection set of cost 21. even though

$\{(3,5), (5,3); (3,4), (4,3); (2,4), (4,2)\}$ is a better solution having cost 20.

Second Example

The next problem consists of the network $\mathcal{N}_2$ in Figure 4 and the target sets

$$\mathcal{T}_2 = \{T_1, T_2, T_3, T_4\}$$

$$T_1 = \{2, 9\} \quad T_2 = \{3, 5\} \quad T_3 = \{2, 8\} \quad T_4 = \{5, 9\}.$$

Figure 4

A Fire Spread Network $\mathcal{N}_2$

To obtain the problem tree for $\mathcal{T}_2$ we form the Boolean product (using 2,3,5,8,9 as Boolean variables)

$$F = (2+9) \cdot (3+5) \cdot (2+8) \cdot (5+9)$$

which after expansion simplifies to

$$F = 2 \cdot 5 + 2 \cdot 3 \cdot 9 + 5 \cdot 8 \cdot 9 + 3 \cdot 8 \cdot 9, \quad (4)$$

whose terms are the minimal representing sets of $\mathcal{T}_2$.

The first problem has sources $X_1 = \{2, 5\}$. The sinks are taken to be a minimal representing set of $\{T_1 - X_1, T_2 - X_1, T_3 - X_1, T_4 - X_1\} = \{\{9\}, \{3\}, \{8\}, \{9\}\}$. The Boolean product $9 \cdot 3 \cdot 8 \cdot 9 = 3 \cdot 8 \cdot 9$ has only one term, so the sinks for the first problem are $Y_1 = \{3, 8, 9\}$. Since X_1 contains none of the T_i , there is no need for further splitting of the network and problem 1 has no subproblems.

Problem 2 has sources given by the second term of (4), $X_2 = \{2, 3, 9\}$. There is only one minimal representing set for $\{T_1 - X_2, T_2 - X_2, T_3 - X_2, T_4 - X_2\} = \{\emptyset, \{5\}, \{8\}, \{5\}\}$, and so the sinks are $Y_2 = \{5, 8\}$. The fact that $T_1 - X_2 = \emptyset$ means that X_2 contains T_1 . A fire protection set separating X_2 from Y_2 will not protect T_1 . Problem 2 has subproblems whose collection of target sets is $\{T_1\} = \{\{2, 9\}\}$. The associated Boolean product has only one factor, $2+9$, indicating that, in addition to the cut of problem 2, the source side of the network, which contains X_2 and hence T_1 , must also be split so that target vertices 2 and 9 are separated. Problem 2 has only one subproblem, problem 3, having sources $X_3 = \{2\}$ and sinks $Y_3 = \{9\}$.

The third term of (4) yields the sources $X_4 = \{5, 8, 9\}$ for problem 4. $\{T_1 - X_4, T_2 - X_4, T_3 - X_4, T_4 - X_4\} = \{\{2\}, \{3\}, \{2\}, \emptyset\}$ has only one minimal representing set, $Y_4 = \{2, 3\}$. Since $X_4 \supset T_4$, problem 4 has subproblems. As before there is only one subproblem having sources $X_5 = \{5\}$ and sinks $Y_5 = \{9\}$. The fourth term of (4) is Y_1 and leads to the same source-sink combination as in problem 1 so the recommended problem tree is complete.

Figure 5.

A Problem Tree for $\mathcal{T}_2$

To solve the problem $(\mathcal{N}_2, \mathcal{T}_2)$ with FIRE, the following input is defined, describing the network $\mathcal{N}_2$:

$$NV = 9 \quad NA = 32$$

$$VWT = (2., 5., 1., 2., 2., 2., 2., 3., 3.)$$

$$AWT = (1., 0., 1. | 0., 1., 1. | 1., 0., 1., 1., 0. | 0., 1., 0. |$$

$$0., 0., 1., 1. | 0., 0., 0., 1. | 0., 1., 1., 0. | 1., 0., 1. |$$

$$0., 1., 0.)$$

$$AWTJ = (1., 1., 1. | 1., 1., 1. | 1., 1., 1., 1., 1. | 1., 1., 1. |$$

$$1., 1., 1., 1. | 1., 1., 1., 1. | 1., 1., 1., 1. |$$

$$1., 1., 1. | 1., 1., 1.)$$

$$JNDX = (2,3,4 | 1,3,7 | 1,2,5,6,7 | 1,5,8 | 3,4,6,8 | 3,5,7,9 | 2,3,6,9 |$$

$$4,5,9 | 6,7,8)$$

$$IAWPTR = (3,6,11,14,18,22,26,29,32),$$

the collection of target sets $\mathcal{T}_2$:

$$NTVERT = 5 \quad NTSETS = 4$$

$$LTSETS = (2,9 | 3,5 | 2,8 | 5,9) \quad LTPTR = (2,4,6,8),$$

and the problem tree of Figure 5:

```

      NPRB = 5           ISUB = (0,0,2,0,4)
      LSRC = (2,5|2,3,9|2|5,8,9|5)   LSRPTR = (2,5,6,9,10)
      LSNK = (3,8,9|5,8|9|2,3|9)     LSNPTR = (3,5,6,8,9)
      NNETS = 2.

```

In Figure 4 one can visualize "generous" two-way cuts of $\mathcal{N}_2$ involving 8 arcs and 2 vertices. In order to dimension the work and output arrays to allow up to 8 arcs and 2 vertices in a fire protection set, one computes the dimensions as indicated in Section 6:

```

      LAFDIM = 2*8 = 16           LVFDIM = 2*2 = 4
      LNADIM = 2(32+9+5) = 92     IACDIM = 16/2 + 4 = 12
      IDCDIM = 16 + 4 = 20        ISWDIM = 16 + 16 + 7(4+0) = 60
      IRWDIM = 92 + 46 + 9 + 2*2 - 1 + 20 + 60 = 230
      IWDIM = 2*92 + 46 + 18*9 + 4*2 + 16 + 4*12 + 5*20 + 60 = 624.

```

Again we request all of the optional output on Tape 7 by setting

```

      IWSLN = 7      IWNET = 7 .

```

The output through the call list of FIRE is

```

      IERR = 0           NAF = 1           NVF = 2

      LARSCF =  $\begin{bmatrix} 2 & \dots \\ 7 & \dots \end{bmatrix}$            LVERTF = (5,3,...)

      WTSUM = 4.,

```

indicating a successful run which computed a one-arc, two-vertex fire protection set $\{(2,7), 5, 3\}$ having a cost of $1.+2.+1. = 4$. On Tape 7 are written the three solution candidates corresponding to the three branches of the problem tree:

(1) COST = 5.00

ARCS

1 2
2 3
2 7

VERTICES

5

(2) COST = 5.00

ARCS

1 2
5 6
8 9
2 7

VERTICES

3

(3) COST = 4.00

ARCS

2 7

VERTICES

5
3

Also on Tape 7 are three subnetworks protected by $\{(2,7), 5, 3\}$:

(1) SOURCE = 1

TARGETS = 2 9

SUBNETWORK = 2 1 9 6 5 4 1

(2) SOURCE = 2

TARGETS = 2 9

SUBNETWORK = 2 9 6 3 2

(3) SOURCE = 2
 TARGETS = 2 9
 SUBNETWORK = 2 9 6 7 2 .

Finally, if one were interested in the smallest dimensions actually needed to run this problem one could look at the output values:

LAFDIM = 3	LVFDIM = 2
LNADIM = 73	IACDIM = 5
IDCDIM = 9	ISWDIM = 28
IRWDIM = 168	IWDIM = 471 .

3. A Description of the FIRE Algorithm

FIRE transverses the problem tree solving a minimum cut problem at each vertex $i = 1, 2, \dots, \text{NPRB}$. For each first-level vertex, a union-of-cuts in $\mathcal{N}'$ (Section 5) is eventually developed and translated into a minimal fire protection set in $\mathcal{N}$, and the first one having the smallest cost is taken as the solution to be output through the call list variables NAF, NVF, LARCSF, LVERTF, and WTSUM.

The union-of-cuts associated with each first-level vertex i in the problem tree is the union of the minimum cut for problem i with the least expensive over the unions-of-cuts for the subproblems of problem i . The network for problem i is the flow network $\mathcal{N}'$. Each subproblem k of problem i in turn has as its associated union-of-cuts, the union of the minimum cut for problem k with the best of the unions-of-cuts for its subproblems. The network for problem k consists of those arcs

both vertices of which belong to the source side of the $X_i' - Y_i'$ cut in $\mathcal{N}'$. Therefore, the union-of-cuts for vertex i evolves from solving minimum cut problems in that branch of the tree rooted at i . The tree traversal must be depth-first, meaning that an entire path from 0 to an endpoint j must be traversed, all solutions for problems having the same parent vertex, p , as j has must be computed, the minimum-cost solution over these siblings united with the minimum cut for problem p and the two costs added together, solutions computed for the siblings of vertex p , the minimum-cost solution over these siblings united with their parent vertex's minimum cut and the two costs added, etc., until a union-of-cuts has been computed for every first-level vertex.

Recall that the vertices of the problem tree are numbered in preorder as described in Section 5.4. This is the order needed for the tree traversal just described.

As mentioned earlier, unions of optimum solutions to subproblems do not necessarily constitute optimum solutions to the main problem. This is one reason that the union-of-cuts associated with the first-level vertices in the problem tree are not necessarily minimum-cost. In fact, these unions may not even be minimal. That is, the associated fire protection sets in $\mathcal{N}$ may contain even lower cost fire protection sets. FIRE tests for this and replaces each nonminimal solution candidate by a minimal one. It tries, but cannot guarantee, to find a minimum-cost fire protection set within each nonminimal one. This is another opportunity at which

nonminimum-cost solutions can creep into the output of FIRE. Once all candidate solutions are minimal, a best among them is selected for output.

The minimality test for solutions associated with first-level problems consists of replacing a union-of-cuts in $\mathcal{N}$ by the collection of corresponding vertices and arc pairs in $\mathcal{N}$ and then testing each individual arc and vertex in a candidate solution to see if it is essential. To describe the test we need the following definition. An s-t net is a subnetwork of paths from a source vertex to a target set. S-t nets are critical because they are explicit ways that a single-source fire can propagate to and destroy an entire set of target vertices. Only minimal s-t nets are of interest, so henceforth s-t nets are assumed to be minimal. By definition a network is unprotected or protected depending on the existence or nonexistence of s-t nets in the network. F is a fire protection set of a network $\mathcal{N}$ with digraph D if and only if $D - F - \{\text{zero weighted arcs and vertices}\}$ has no s-t nets. FIRE tests and makes minimal the candidate fire protection set F_i for first-level vertex i by (a) ordering the arcs of F_i according to the cost savings that would result if each arc were deleted from F_i , (b) ordering the vertices of F_i in the same way, (c) forming the digraph $G_i = D - F_i - \{\text{zero weighted arcs and vertices}\}$, and (d) for each element e (arc or vertex) of F_i , first for vertices in order of greatest potential cost savings then for arcs in a similar order, testing for the existence of an s-t net in $G_i + e$ by a pathfinding routine, deleting e from F_i .

and reducing the cost of F_i if no s-t nets exist. Notice that the minimality test seeks paths in the unsymmetric fire spread digraph D modified by deletions and insertions. Thus, minimality with respect to fire protection is guaranteed.

The minimum cuts for each problem vertex are computed by a subroutine MAXFLO [8] implementing Dinic's max-flow min-cut algorithm [3]. The ordering of arcs and then vertices during the minimality test is done with a subroutine LISORT, which carries out Knuth's list insertion sort [7] using linked lists. The existence of s-t nets is determined by repeated use of the subroutine SPATHF, which implements Ford's shortest path algorithm using a sequence list [5], to find paths from each fire area vertex $1, 2, \dots, NV$ to all other vertices until some target set T_j is found to be reachable from some source vertex. FIRE takes due notice of the fact that if no T_j can be reached from a vertex i , then no T_j can be reached from any vertex reachable from i , and hence the vertices reachable from i are not used as sources in the pathfinding process.

Another instance of computational economy occurs when an arc (j,k) is deleted from fire protection set F_i because it is not essential. If F_i also contains an as yet untested arc (m,n) and in $G_i + (j,k)$ vertex n is reachable from m , then (m,n) is known to be nonessential also, and it is immediately deleted from F_i avoiding a later test.

If $IWSLN > 0$, then each minimal fire protection set associated with a first-level problem is written on Tape IWSLN. If $IWNET > 0$, then FIRE returns to the minimality test with the

output fire protection set F . We know that F is already minimal. The purpose of repeating the test is to identify a subcollection of the s - t nets protected by F . The instant that each element e in F is found to be essential to fire protection, the pathfinder has determined a source, a target set, and a collection of paths from source to targets. These constitute an s - t net containing e because G_i has no s - t nets but $G_i + e$ does. Thus, protecting e protects this s - t net, and protecting F protects the whole system, meaning F contains some element (arc or vertex) from every s - t net in $\mathcal{N}$. That s - t net discovered at the instant that each $e \in F$ is found to be essential is written on Tape IWNET. These subnetworks are intended for future use in an iteration [6] which will converge to a complete set of minimum-cost solutions.

9. Subroutine Performance

Much attention has been given to both the run-time and storage efficiency of FIRE. The input network as well as the flow networks are stored in successor form using a minimum amount of storage. The basic minimum cut, pathfinding and sorting subroutines are the best routines that were readily available to us. They are based on methods that performed well in recent tests [2,5]. Consequently the run times and storage requirements of FIRE are quite reasonable. As shown in the last two entries of Table 1, realistic problems require only a few seconds on the CDC 7600. The array storage shown in Table 1 is computed from

$$\begin{aligned} \text{array storage} = & 3NA + 2NV + NTSETS + 3NPRB + \\ & \text{dim(LTSETS)} + \text{dim(LSRC)} + \\ & \text{dim(LSNK)} + \text{IRWDIM} + \text{IWDIM} + \\ & 2LAFDIM + \text{LVFDIM} . \end{aligned}$$

Table 1 also shows the cost achieved by FIRE as well as the minimum cost, which was computed by an iterative procedure using FIRE to obtain upper bounds at each stage. These test problems were devised solely to provide some variation in problem size. They were not intended to provide a systematic study of how often FIRE achieves optimality or expected deviations from optimality.

The last two problems are two different weightings for a network of realistic size. Pairing every source set X_j with every minimal representing set of the $T_i - X_j$ would have produced a problem tree with 96 vertices. Instead we trimmed the problem tree to 18 vertices by pairing each X_j with only one of the possible sink sets. The full recommended problem tree was used for all the other problems.

TABLE 1
FIRE Performance

Problem No.	No. of Vertices NV	No. of Arcs NA	No. of Prob. Vert. NEFB	No. of Solution Candid.	Array Storage (decimal)	DDC 7600 Run Time (sec.)	Computed Protection Cost	Minimum Protection Cost
1	2	2	1	1	126	.004	2.	2.
2	3	6	1	1	223	.007	20.	20.
3	5	12	1	1	332	.008	26.	26.
4	4	10	2	2	292	.009	17.	17.
5	8	20	1	1	489	.009	23.	22.
6	9	32	5	3	807	.023	4.	4.
7	9	32	2	2	672	.015	3.	3.
8	20	62	3	2	1321	.038	21.	21.
9	20	60	6	3	1455	.054	50.	50.
10	20	60	11	5	1588	.109	89.	89.
11	20	60	15	7	1605	.115	55.	55.
12	35	512	18	12	8822	.722	6.	6.
13	35	512	18	12	8676	1.313	20.	18.

REFERENCES

- [1] Dennis L. Berry, Nuclear Power Plant Fire Protection -- Philosophy and Analysis, SAND80-0334, Sandia National Laboratories, Albuquerque, NM, May 1980.
- [2] To-Yat Cheung, Computational Comparison of Eight Methods for the Maximum Network Flow Problem, ACM Transactions on Math. Software 6 (1980), 1-16.
- [3] E. A. Dinic, Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation, Soviet Math. Dokl. 11 (1970), 1277-1280.
- [4] L. R. Ford, Jr. and D. R. Fulkerson, Flows in Networks, Princeton University Press, Princeton, 1962.
- [5] Bernie L. Hulme and John A. Wisniewski, A Comparison of Shortest Path Algorithms Applied to Sparse Graphs, SAND78-1411, Sandia National Laboratories, Albuquerque, NM, August 1978.
- [6] Bernie L. Hulme, Ann W. Shiver, and Peter J. Slater, A Network Analysis of Fire Protection, to appear.
- [7] Donald E. Knuth, The Art of Computer Programming, Volume 3, Sorting and Searching, Addison-Wesley, Reading, MA, 1973.
- [8] Peter J. Slater, A Code for Dinic's Maximum Flow Algorithm, SAND81-1927, Sandia National Laboratories, Albuquerque, NM, November 1981.
- [9] R. B. Worrell, Set Equation Transformation System (SETS), SLA-73-0028A, Sandia National Laboratories, Albuquerque, NM, January 1975.
- [10] R. B. Worrell and D. W. Stack, A SETS User's Manual for the Fault Tree Analyst, SAND77-2051, Sandia National Laboratories, Albuquerque, NM, November 1978.

DISTRIBUTION

J. W. Bennett, Actg. Director
LWR Technology Team, NE-540
Office of Nuclear Power Systems
U.S. Department of Energy
Mail Stop B-107
Washington, DC 20545

J. H. Carlson, NE-540
Office of Nuclear Power Systems
U.S. Department of Energy
Mail Stop B-107
Washington, DC 20545

Dan Giessing, NE-540
Light Water Reactor Technology Team
Office of Nuclear Power Systems
U.S. Department of Energy
Mail Stop B-107
Washington, DC 20545

J. D. Griffith, Acting Director
Office of Nuclear Power Systems, NE-540
U.S. Department of Energy
Mail Stop B-107
Washington, DC 20545

James Hunter, NE-530
Engineered Systems and Components Team
Office of Reactor Research and Technology
U.S. Department of Energy
Mail Stop B-107
Washington, DC 20545

A. C. Millunzi, Team Leader
Safety Research and Development Team, NE-540
Office of Nuclear Power Systems
U.S. Department of Energy
Mail Stop B-107
Washington, DC 20545

M. P. Norin
Office of Safety, Quality Assurance and
Safeguards, NE-510
U.S. Department of Energy
Mail Stop B-107
Washington, DC 20545

DISTRIBUTION (continued)

J. Yevick, NE-540
Safety Research and Development Team
Office of Nuclear Power Systems
U.S. Department of Energy
Mail Stop B-107
Washington, DC 20545

Profession Loss Control
P.O. Box 446
Oak Ridge, TN 37830
Attn: K. W. Dungan

1222	C. F. Diegert
1230	W. L. Stevens, Attn: R. E. Smith, 1233
1710	V. E. Blake
1720	C. H. Mauney
1750	T. A. Sellers
1760	J. Jacobs
1765	D. S. Miyoshi
1768	C. E. Olson
2113	J. A. Wisniewski
2646	M. R. Scott
4400	A. W. Snyder
4410	D. J. McCloskey, Attn: N. R. Ortiz, 4413
4412	J. W. Hickman
4413	D. R. Strip
4414	G. B. Varnado
4414	D. E. Bennett
4414	D. L. Berry
4414	J. L. Darby
4414	D. M. Ericson
4414	D. R. Gallup
4414	A. W. Shiver
4414	D. W. Stack
4414	R. B. Worrell
4416	L. D. Chapman
4416	H. A. Bennett
4440	G. R. Otey
4441	M. Berman
4442	W. A. Von Rieseemann
4442	D. Dube
4442	L. J. Klammerus
4443	D. A. Dahlgren
4443	J. L. Sprung
4444	S. L. Thompson
4445	L. O. Cropp
4756	D. Engi
5513	D. W. Larson
5513	D. D. Cline

DISTRIBUTION (continued)

5600	D. B. Shuster, Attn:	A. A. Lieber, 5610 M. M. Newsom, 5620 R. C. Maydew, 5630 D. J. Rigali, 5650
5622	D. D. Boozer	
5640	G. J. Simmons, Attn:	R. J. Thompson, 5641 L. F. Shampine, 5642
5641	C. A. Barefoot	
5641	E. F. Brickell	
5641	J. Moore	
5641	P. J. Slater	
5642	R. J. Hanson	
5642	B. L. Hulme	(50)
8320	R. L. Rinne	
8330	G. W. Anderson	
8214	M. A. Pound	
3141	L. J. Erickson	(5)
3151	W. L. Garner	(3)
3154-3	C. H. Dalin	(25)
	For NRC for NTIS	