MATHEMATICAL THEORY OF POWER SYSTEM RELIABILITY EVALUATION

by

Felix Wu et al.

Memorandum No. UCB/ERL M79/67

5 November 1979

MATHEMATICAL THEORY OF POWER SYSTEM RELIABILITY EVALUATION

Memorandum No. UCB/ERL M79/67

November 5, 1979

Electronics Research Laboratory
College of Engineering
University of California, Berkeley
94720

PREFACE

This is an interim report on the research conducted under contract EC-77-S-01-5105 with the Division of Electic Energy Systems, Department of Energy. The report is prepared for the presentation at Engineering Foundation Conference, Systems Engineering for Power VI: Electric Power System Effectiveness Analysis, Asilomar Conference Grounds, Pacific Grove, California, on November 4-9, 1979. Professors F. F. Wu, R. E. Barlow, and A. R. Bergen are responsible for the supervision of the graduate students research.

We would like to thank L. H. Fink, R. L. Sullivan (formerly with DOE), and T. A. Trygar (DOE) for their advice and support on this research project.

TABLE OF CONTENTS

INTRODUCTION

Felix F. Wu

I. POWER SYSTEM RELIABILITY EVALUATION

The power system reliability problem is to evaluate the ability of a system to supply the load demand, taking into account the random effects of equipment outages and load fluctuations. Probability methods are used in the analysis [1-5]. The classical loss-of-load probability (LOLP) model [6] for generation reliability has been well developed. This model assumes that the transmission system is perfectly reliable and has unlimited power transfer capability. The reliability problem that incorporates transmission system becomes a difficult problem in terms of computational complexity. Work in this area has been relatively scarce and primitive. In this introduction we will present:

(i) a brief review on the current state-of-the-art of methodologies for bulk power power system reliability evaluation that incorporates transmission system.

(ii) a description of our research directions in the development of a methodology for bulk power system reliability evaluation.

(iii) a summary of our research results so far accomplished for this project.

To put different methods into proper perspectives we first present a unified conceptual model for power system reliability evaluation. Different methods are then classified within the model into four categories. Most of the current research in power system reliability evaluation falls into the first three approaches. The last one is the one we are following. It is along the same direction as does the development of modern mathematical theory of reliability.

II. METHODOLOGIES FOR POWER SYSTEM RELIABILITY EVALUATION

2.1. A Unified Conceptual Model

The basic components of a bulk power supply system consists of generation, transmission and load. The generation and transmission equipments have random outages. The load demand is fluctuating and unpredictable. The power system reliability problem is to evaluate the ability of a system to supply the load demand, taking into account the random effects of equipment outages and load fluctuations.

Generators, transmission lines, and loads will be referred to as components in the model. The state of a generator is the maximum power that can be supplied. The state of a transmission line can be either the maximum power that can be transferred or its admittance. The state of a load is the power demand. Let us for simplicity assume that the states are discrete and finite. If each component has k states and there are n components we have k^n possible combinations, each of which is called a system state or a configuration.

For a fixed configuration a test is conducted in order to determine the ability of that particular generation and transmission system configuration to supply that particular load demand. The following is a list of models used in the tests for reliability evaluation. The list is arranged in increasing order of complexity. References of methods using the model is cited in the brackets.

1). Service continuity [7]. The test is simply to determine whether there is a transmission path from generation to load point.

2). Network flow [8]. Only the real power flow and its constraints are considered.

3). Load flow [9-13]. DC load flow, ac load flow and its approximations have been used. The model includes real and reactive power flows, and voltage magnitudes.

We now describe our unified model in mathematical terms. Let x_i denote the state of component i. A configuration of a system is represented by an n-vector $\underline{x} = (x_1, x_2, \ldots, x_n)$. The collection of all possible system configurations or states $\underline{x}$ is called the state-space of the system and is denoted by X. The test to determine whether the configuration is able to supply load can be represented by a binary-valued function ϕ,

$$\phi(\underline{x}) = \begin{cases} 1 & \text{if load can be supplied} \\ 0 & \text{if load can not be supplied.} \end{cases} \tag{1}$$

The subset of X for which the corresponding $\underline{x}$ passes the test is called the "working" subset and is denoted by W, i.e., $W = \{\underline{x} | \phi(\underline{x})=1\}$. The complement of W is called the "failed" subsets and is denoted by F, i.e., $F = \{\underline{x} | \phi(\underline{x})=0\}$.

For each component there is a probability distribution associated with the occurrence of different states. In other words, the state of a component x_i is a random variable. For each configuration $\underline{x}$ we can associated with it a probability, $\text{Prob}\{\underline{x}\}$, induced from the probabilities of its components x_i.

A reliability function is a set function f defined on the subsets of X such that the function is evaluated on the intersection with either W or F. For example, let $A \subseteq X$, we may define

$$f(A) = \text{Prob}\{\underline{x} | \underline{x} \in A \cap F\} \tag{2}$$

A reliability measure associated with a reliability function f is defined as f(X). For example if f is defined as above f(X) will be the loss-of-load probability of the system.

$$f(x) = \sum_{\underline{x} \in F} \text{Prob}\{\underline{x}_i\} \tag{3}$$

The model we have described so far is a probabilistic one in which the time element is not included. We may view each component of a power system as an alternating renewal process with exponential distributions of "working" time and repair time. A continuous-time Markov chain model of the system is obtained. For reliability evaluation, the expected duration of being in the set W, and the expected frequency of entering W, in addition ot the probability of W, can be calculated. They provide additional reliability measures.

2. Classification of Evaluation Methods

Reliability evaluation of bulk power system based on service continuity is generally considered inadequate. We will therefore consider only methods based on network flows and load flows. Because the number of possible states in the system increases exponentially as the number of components increases, an exhaustive enumeration is out of the question. Various methods have been suggested in practice. Based on the essential features we classify the various methods within our unified model into four categories of alternative approaches.

Approach I. This approach selects a few sample points in X. Calculation of reliability measures are performed using only these points.

The fundamental steps involved in this approach are

1) Select sample points $\underline{x}_i$ (Contingency selection)

2) Evaluate $\phi(\underline{x}_i)$

3) Compute reliability $\sum_{\{\underline{x}_i \mid \phi(\underline{x}_i)=0\}} f(\underline{x}_i)$

This approach is currently widely used in industry. The second step of evaluating $\phi(\underline{x}_i)$ may be carried out by DC power flow, or full AC power flow.

The approach used by Reppen [9], McCoy [10] and Marks [11] all have the same basic features of this category.

Approach II. In this approach it is assumed that the reliability function f is additive, i.e.,

$$f(A \cup B) = f(A) + f(B)$$

This approach is based on the principle that if a family of subsets $\{E_i\}$ is a partition of the state-space X, i.e.,

$$\bigcup_i E_i = X$$

$$E_i \cap E_j = \phi, \quad i \neq j$$

then

$$f(W) = \sum_i f(W \cap E_i)$$

Two fundamental steps are involved in this approach:

1). Select $\{E_i\}$

2). Compute $f(W \cap E_i)$

The second step involves performing a test or a number of tests. The conditional probability method suggested by Billinton and Bhavaraju [12-13] belongs to this class. A family of subsets consisting of single contingencies and possibly some double contingencies are used. They do not exhaust the state-space X, but only form a set of sample points as in Approach I. The reliability measures used are the loss-of-load probability and the expected frequency of loss-of-load.

Approach III. It is still required to assume that the reliability function f is additive. This approach is based on the fact that if we partition S into nonoverlapping subsets $\{W_i\}$, then

$$f(W) = \sum_i f(W_i)$$

This approach is viable if the family of subsets $\{W_i\}$ can be generated recursively.

Two fundamental steps are involved in this approach

1). Perform a test, the result of which defines S_i. Update f(W).

2). Decompose $W - \bigcup_{k<i} W_k$ if necessary. For each subset go to Step 1.

The method developed by Doulliez and Jamoulle [8] belongs to this class. Here because the network flow model is used it is possible to generate W_i recursively.

Approach IV. Two fundamental steps are involved in this approach.

1). Characterize the working subset W and its properties.

2). Evaluate f(W) based on the characterization and properties of W.

This approach is the one we are following. It is along the same direction as does the development of modern mathematical theory of reliability [14].

III. RESEARCH DIRECTIONS AND RESULTS OF THE PROJECT

3.1. Research Directions

The main directions of our research efforts in the development of a methodology for bulk power system reliability evaluation are the following:

1). Based on the mathematical model we want to analytically characterize the set of working states W (or the set of failure states F) and to derive its properties.

2). Develop methods to evaluate system reliability or its upper and lower bound based on the characterization and properties of the sets W and F.

We believe that for reliability evaluation of large power systems the idea of decomposing the system into subsystems or modules is important. The main directions of our research in modular decomposition are the following:

1). Study the fundamental requirements of a module and its properties.

2). Develop decomposition schemes based on the understanding of the properties of a module.

The model for bulk power system reliability evaluation using flow networks belongs to the class of systems considered in the modern Reliability Theory. In section 2 we state some basic concepts in Reliability Theory to facilitate our later presentation. In section 3 we summarize the research results we accomplished in the area of power system reliability evaluation using flow network model. This includes the development of evaluation methods and modular decomposition. In section 4, we summarize our research in laying a solid foundation for reliability evaluation based on Markov model. In section 5 we summarize the on-going research on the analysis of the reliability evaluation model based on DC load flow and on a direct method for reliability evaluation.

It seems to be generally agreed in power industry that for transmission system reliability evaluation a DC power flow or an AC power flow should be used. The justification for our investigation of reliability model with flow networks consists of the following:

1). This is an area where theoretical study is almost nonexist.

2). The results obtained for this model may serve as a foundation for further research for the model with power flows.

3). This model itself is considered adequate for multi-area or

regional reliability evaluation, and distribution system reliability evaluation.

3.2. Basic Concepts from Reliability Theory

The fundamental concept in Reliability Theory [14] is that of coherency. A coherent system is one where improving the performance of a component will not cause the system to deteriorate, i.e., the function ϕ is nondecreasing. A coherent system may be characterized by way of a minimal path representation or a minimal cut representation. A minimal path set is a minimal set of components whose functioning insures the functioning of the system. A minimal cut set is a minimal set of components whose failure causes the system to fail. A coherent system has a unique family P of minimal path sets and a unique family C of minimal cut sets such that the system reliability function can be expressed in terms of either of them. When a coherent system is represented by a minimal cut structure and a minimal path structure, bounds for the reliability function can be readily obtained.

The concept of a module is related to decomposition of complex systems. A module is a set of components which can be replaced conceptually by a single fictitious component. The state of the fictitious component is determined according to a well-defined rule from the states of the components in the module. The state of the system may be determined from the state of the fictitious component together with the states of the rest of the components.

3.3. Reliability Evaluation Based on Flow Networks

It can easily be shown that the reliability model with flow network is a coherent system. Because of this coherency property, when a network flow is performed the result can also be used for a subset of system

states. The iterative method developed by Doulliez and Jamoulle [8] utilizes this property. In order to facilitate probability calculations, they decompose the set not yet classified at each iteration into nonoverlapping subsets. Lee in Chapter 2 develops an improved scheme for decomposition of the unclassified sets in the state space by the use of lexicographic ordering.

The algorithm by Doulliez and Jamoulle generates a path set or a cut set one at a time. Eventually the iterative process will reach all the "minimal" path sets in P and "minimal" cut sets in C one by one. On the other hand, Willie, in Chapter 3, develops a new algorithm which generates a family of path sets and a family of cut sets at each iteration. The scheme will converge to the family of minimal path sets P and the family of minimal cut sets C. The method is based on the property that each cut set must intersect all the path sets. Given a family of sets R, the dual family $d(R)$ is the family of sets such that each set in the dual family intersects all the sets in R. Willie's algorithm at each iteration constructs a dual family and tests certain sets derived from the dual family to enlarge the families of path sets and cut sets. The method has the provision to terminate when an "interesting" subfamily is found, e.g., one which is limited by size or by probability.

In Chapter 4 Hagstrom investigates the possible definitions of modules in the reliability model with flow metworks. Her starting point is a graph-theoretic characterization of the set of working states W by Gale. Two types of modules, structural module and functional module, are defined. Their characterization, properties and relation are studied. The report included here does not contain her latest results on this topic. A complete report is forthcoming.

In Chapter 5 Shogan defines a special class of modules and develop a modular decomposition scheme for a certain class of networks. Upon adapting the graph-theoretic concepts of "cut node" and "block," it is possible to identify a "block-module," defined as an independent, non-trivial subnetwork that has one and only one node (the "cutnode") connected to the nodes outside the subnetwork. The reliability of the network will increase by a known factor after a "block-modular decomposition" that consists of a transformation of the cutnode's supply-demand random variable and the deletion of the remainder of the block-module. Provided that the original network possesses at least one block-module, the reliability can be determined from a sequence of block-modular decompositions that reduce the original network to a single node whose reliability is easily computed.

Hagstrom develops a decomposition scheme in Chapter 6, which introduces a more general framework. A graph-theoretic decomposition is used first to decompose the network into its "triconnected components." A "tree" to show how to reassemble the graph from its triconnected components can be generated. This decomposition can be used to compute system reliability. The procedure is that the probability distribution of capacity or demand is computed for each of the triconnected component, then this process is repeated as these components are iteratively reassembled back into the graph itself.

3.4. Reliability Theory Based on Markov Model

We provide here a solid theoretical foundation fcr the Markov model of power system reliability. In the Markov model of power system reliability evaluation, the probability distributions of component working time and repair time can be assumed exponential. When the components are

assembled together to form the system the probability distribution of the time to system failure is no longer exponential. Narasimhamurthi, in Chapter 7, derives the expression of the time to failure distribution for systems modeled as continuous-time Markov chains. Greif, in Chapter 8, derives an expression for the long-term average frequency of entering and duration of staying in the set of failure states and develops a recursive formula for computing the frequency and duration for a coherent system. The recursive formula is a generalization of the one developed by Hall, Ringlee and Wood and can be applied to bulk power system reliability evaluation.

3.5. Reliability Evaluation Based on DC Power Flow

A more accurate model of a power system for reliability evaluation than the flow networks is the DC power flows. Moslehi investigates the properties of this model in Chapter 9 and suggest a direct method for reliability evaluation without solving the power flow. It turns out that reliability model using DC power flow is not a coherent system. The concept of local coherency is introduced and its application to obtain a characterization of the set of working states W is indicated. A sufficient condition for local coherency is derived. The class of network topology which guarantees coherence is identified. A method of reliability evaluation which is independent of the concept of coherency is suggested. A sequence of hyperboxes is iteratively constructed in this method and a subset of the working states is readily obtained for each hyperbox. The method is direct and does not require the solution of a power flow.

REFERENCES

1. R. Billinton, Power System Reliability Evaluation, Gordon and Breach, New York, 1970.

2. R. Billinton, R. J. Ringlee and A. J. Wood, Power System Reliability Calculations, MIT Press, 1973.

3. R. L. Sullivan, Power System Planning, McGraw-Hill, 1977.

4. A. D. Patton and A. K. Ayoub, "Reliability evaluation," in Systems Engineering for Power: Status and Prospects, ed. by L. H. Fink and K. Carlsen, ERDA 1975, pp. 275-289.

5. J. Endrenyi, Reliability Modeling in Electric Power Systems, John Wiley, 1979.

6. G. Calabrese, "Generating reserve capability determined by the probability method," AIEE Trans., vol. 66, pp. 1439-1450, 1947.

7. Z. G. Todd, "A probability method for transmission and distribution outage calculations," IEEE Trans., PAS-83, pp. 695-701, July 1964.

8. P. Doulliez and E. Jamoulle, "Transportation networks with random arc capacities," Revue Francaise d'Automatique, Informatique et Recherche Operationnelle, vol. 3, pp. 45-59, Nov. 1972.

9. N. D. Reppen, "Comprehensive bulk power reliability evaluation," EPRI Workshop Proceedings: Power System Reliability - Research Needs and Priority, pp. 2-18 to 2-29, 1978.

10. M. F. McCoy, "Reliability evaluation to aid in long-range transmission planning," Reliability Conference for the Power Industry, New York, 1977.

11. G. E. Marks, "A method of combining high-speed convergency load flow analysis with stochastic probability methods to calculate a

quantitative measure of overall power system reliability," Paper A78 053-1, presented at the IEEE PES Winter Meeting, Jan. 1978.

12. R. Billinton, "Composite system reliability evaluation," IEEE Trans., PAS-88, pp. 276-280, Apr. 1969.

13. R. Billinton and M. P. Bhavarajn, "Transmission planning using a reliability criterion, Part I - a reliability criterion," IEEE Trans., PAS-89, pp. 28-34, Jan. 1970.

14. R. E. Barlow and F. Proschan, Statistical Theory of Reliability and Life Testing, Holt, Rinehart and Winston, 1975.

RELIABILITY EVALUATION OF A COHERENT SYSTEM

Sun H. Lee*

ABSTRACT

We present a method for evaluating the exact reliability of a coherent system using the concept of lexicographic ordering. The procedure is general in the sense that it does not depend on the structure function of the system, and the memory requirement is negligible when executing the algorithm on a computer.

* Department of Electrical Engineering and Graduate Program in Operations Research, North Carolina State University, Raleigh, N.C.

I. INTRODUCTION

In system reliability analysis, we often assume that the system is represented by a prohabilistic graph, and the system is functioning if there exists a path from the input node to the output node. Thus we have considered reliability as a matter of connectivity only and reliability analysis has been primarily concerned with the enumeration of paths or cuts in the graph [1,2,etc.]. But in many physical systems such as power transmission systems, and oil or water pipeline systems, there will be numbers associated with every arc, for instance, the flow capacity of the arc; consequently, reliability of an arbitrary system may not necessarily be characterized by only connectivity.

In this paper, we give a definition of system reliability [3] and present a procedure for computing the exact reliability of a "coherent" system using the concept of lexicographic ordering.

Throughout the paper we assume that the system consists of n statistically independent components. Let a random variable X_i indicate the state of the ith component:

$$X_i = \begin{cases} 1 & \text{if component i is functioning} \\ 0 & \text{if component i is failed.} \end{cases}$$

Thus the system can be completely described by a vector-valued random variable $X = (X_1,\ldots,X_n)$.

Let the binary variable ψ indicate the performance of the system:

$$\psi = \begin{cases} 1 & \text{if the system is functioning} \\ 0 & \text{if the system is failed.} \end{cases}$$

It is assumed that the performance of a system depends on the states of the components of the system; thus, we define a function $\psi(X)$ and call

it the structure function of the system.

II. PRELIMINARIES NOTATION AND DEFINITIONS

x_i	values that X_i can assume.
$x = (x_1,\ldots,x_n)$	values that X can assume
p_i	$\Pr\{X_i = 1\}$.
$q_i (=1-p_i)$	$\Pr\{X_i = 0\}$.

System reliability: The reliability of a system is defined as (see [3]):

$$\Pr\{\psi(X) = 1\}. \tag{1}$$

Coherent system: A system of components is coherent if its structure function ψ is nondecreasing and each component is relevant.

Path vector: A path vector is a vector x such that $\psi(x) = 1$.

Minimal path vector: A minimal path vector is a path vector x such that $x' < x \Rightarrow \psi(x') = 0$, where the notation $x' < x$ implies $x_i' \leq x_i$, $i = 1,\ldots,n$, with at least one strict inequality.

Cut vector: A cut vector is a vector x such that $\psi(x) = 0$.

Minimal cut vector: A minimal cut vector is a cut vector x such that $x' > x \Rightarrow \psi(x') = 1$.

We wish to emphasize that the meaning of the above-mentioned paths or cuts is not equivalent to that in graph theory; however, they are equivalent if we consider reliability as a matter of connectivity only. The reliability (1) of any coherent system can be computed by using the minimal path and minimal cut representations [3]. Nevertheless, unless the structure function of the system is simple, it is not an easy task to determine all minimal paths or minimal cuts.

A brute force approach to evaluating (1) is to sum over all 2^n

binary n-vectors so that we have

$$\Pr\{\psi(X) = 1\} = \sum_{x} \psi(x) \prod_{i} p_i^{x_i} q_i^{1-x_i}. \qquad (2)$$

Observing that the method (2) is impractical, we can reduce our problem to one of finding the set of all path vectors $\{x|\psi(x) = 1\}$ without complete enumeration. Doulliez and Jamoulle [4] have applied decomposition principle to calculating the system reliability, in which the whole state space is decomposed into three categories: sets of functioning states, sets of failed states and sets of undetermined states. Each set of undetermined states is again decomposed into three categories, and so forth. This involves keeping track of numerous sets of undetermined states as well as the relevant upper and lower limiting states of each set; hence, it requires large memory size when large-scale systems are considered.

III. APPROACH

Our procedure hinges on the following definitions:

Lexicography: (a) A vector is lexicographically positive (or negative) if its first nonzero component is positive (or negative). Thus the vector (0,0,3,-4,1) is lexicographically positive and the vector (0,-2,1,5,3) is lexicographically negative. A vector x^1 is lexicographically greater than (or smaller) than x^2 if (x^1-x^2) is lexicographically positive (or negative).

(b) A vector x' is lexicographically greater than x" with respect to the ordered index set A if $x'\langle A\rangle - x''\langle A\rangle \overset{L}{>} 0$ (see the notation below).

NOTATION

$x \overset{L}{>} 0$	a vector x which is lexicographically positive.
$x \overset{L}{<} 0$	a vector x which is lexicographically negative.

$x' \overset{L}{>} x''$	x' is lexicographically greater than x".
$x' \overset{L}{\underset{(A)}{>}} x''$	x' is lexicographically greater than x" with respect to the ordered index set A.
$\|\cdot\|$	cardinality of the set $\cdot$.
A	"ordered" index set of some or all elements in a binary n-vector x
F	index set, without ordering, of the remaining elements in x. Note $\|A\| + \|F\| = n$.
$x\langle A\rangle$	$\|A\|$-vector x whose elements are ordered according to the ordered index set A. For example, given $A = \{3,1,5\}$, $F = \{2,4\}$ and $n = 5$, $x\langle A\rangle = (x_3, x_1, x_5)$.
$\hat{A}^i$	ordered index set A with $\|\hat{A}^i\| = i$, whose elements are the same as the first i elements in A. In the above example, $\hat{A}^2 = \{3,1\}$.
ϕ	null set

Our strategy is essentially as follows. Suppose that we have the desired set $\{x|\psi(x) = 1\}$; therefore, it is assumed to be possible to partition this set into exclusive and exhaustive subsets, say $M^1, M^2, \ldots, M^z$, by the following partitioning procedure. In fact, the algorithm described in the next section is nothing but a systematic scheme for generating all these subsets.

Partitioning Procedure:

0. Let $\hat{x}$ be any path vector, i.e., $\psi(\hat{x}) = 1$; order the index set $\{j|\hat{x}_j = 1\}$ arbitrarily to get an ordered index set A.
1. Partition the set $\{x|\psi(x) = 1\}$ into $|A| + 1$ exclusive and exhaustive subsets $H^0, H^1, \ldots, H^{|A|}$ so that for any $x' \in H^i$ and $x'' \in H^j$, $x' \overset{L}{\underset{(A)}{>}} x''$ if $i < j$. Of course, for some i H^i may possibly be empty.

After partitioning, we have:

$$\text{for any } x \in H^0,\ x\langle A\rangle = (1,1,\ldots,1,1)^{|A|},$$

$$\text{for any } x \in H^1,\ x\langle A\rangle = (1,1,\ldots,1,0),$$

$$\vdots \qquad\qquad \vdots$$

$$\text{for any } x \in H^i,\ x\langle \hat{A}^{|A|-i+1}\rangle = (1,\ldots,1,0)^{|A|-i+1},$$

$$\vdots \qquad\qquad \vdots$$

$$\text{for any } x \in H^{|A|},\ x\langle A^1\rangle = (0).$$

Note that $\psi(x^0) = 1$, where

$x^0 \in \{x | x_j = 1 \text{ for } j \in A,\ x_j \geq 0 \text{ for } j \notin A\}$, for the system is coherent by our assumption.

2. Excluding the set H^0, for each $i = 1,\ldots,|A|$, partition the set H^i, if not empty, into exclusive and exhaustive subsets. In what follows, without loss of generality, we explain how to partition H^i. For every $j \in \hat{A}^{|A|-i+1}$, let x_j^* denote the value of component j of x in H^i. Suppose that we can find an ordered index set A_α by augmenting the set $\hat{A}^{|A|-i+1}$ so that $\psi(x^{**}) = 1$, where $x_j^{**} = x_j^*$ for $j \in \hat{A}^{|A|-i+1}$, $x_j^{**} = 1$ for $j \in A_\alpha$ and $j \notin \hat{A}^{|A|-i+1}$, and $x_j^{**} = 0$ for $j \notin A_\alpha$. Note that $\hat{A}^{|A|-i+1}$ constitutes the first $|A|-i+1$ elements of the set A_α. Now partition the set H^i into $|A|-|A|+i$ exclusive and exhaustive subsets $G^0, G^1,\ldots,G^{|A_\alpha|-|A|+i-1}$ (some of them may possibly be null sets) so that for any $x' \in G^k$ and $x'' \in G^r$, $x' \overset{L}{(A_\alpha)} x''$ if $k < r$. As before, we have:

$$\text{for any } x \in G^0,\ x\langle A_\alpha\rangle = (1,\ldots,1,0,1,\ldots,1,1)^{|A_\alpha|},$$

$$\text{for any } x \in G^1,\ x\langle A_\alpha\rangle = (1,\ldots,1,0,1,\ldots,1,0)^{|A|-i+1},$$

$$\vdots \qquad\qquad \vdots$$

$$\text{for any } x \in G^{|A_\alpha|-|A|+i-1},\ x\langle \hat{A}_\alpha^{|A|-i+2}\rangle = (1,\ldots,1,0,0).$$

Note that $\psi(x^0) = 1$, where $x^0 \in \{x|x_j = x_j^{**}$ for $j \in A_\alpha$, $x_j \geq 0$ for $j \notin A_\alpha\}$.

3. Except the set G^0, we again partition $G^1,\ldots,G^{|A_\alpha|-|A|+i-1}$ into exclusive and exhaustive subsets, respectively, in the same way that we partitioned H^i. Of course, this partitioning procedure is applied to all subsets obtained from the partitioning of H^k, $k = 1,\ldots,|A|$, and the procedure continues until no more partitioning is possible. This procedure can be visualized in terms of a tree, as shown in Fig. 1, and the tree may not be uniquely determined.

Now suppose that the above partitioning procedure has yielded exclusive and exhaustive subsets, say $M^1,\ldots,M^k,\ldots,M^z$, of the set $\{x|\psi(x) = 1\}$.

Properties of M^k: Associated with M^k is an ordered index set A_k satisfying the following:

i) For any x', $x'' \in M^k$, $x'\langle A_k\rangle = x''\langle A_k\rangle$; for all $j \in A_k$, let x_j^* denote the value of component j of x in M^k.

ii) $\psi(x^0) = 1$, where $x_j^0 = x_j^*$ for $j \in A_k$, and $x_j^0 = 0$ for $j \notin A_k$; consequently, it follows from the assumption that the system is coherent $M^k = \{x|x_j = x_j^*$ for $j \in A_k$, $x_j \geq 0$ for $j \notin A_k\}$.

Remark 1: $\Pr\{\psi(X) = 1\} = \sum_{k=1}^{z} \Pr\{x \in M^k\}$, where $\Pr\{x \in M^k\}$

$$= \prod_{j \in A_k} \Pr\{X_j = x_j^*\} \sum_{j \notin A_k} \Pr\{X_j \geq 0\}.$$

The conceptual implementation of the above idea may be simple; however, the key to the procedure is the ability to partition sets in such a way that it does not require large memory size.

IV. ALGORITHM

We now state the algorithm in detail. The justification of the algorithm is given in the appendix. Our algorithm begins with any path vector, say x'; let $J = \{i | x'_i = 1\}$ and $\bar{J} = \{i\ x'_i = 0\}$.

Algorithm:

0. (Initialization) Order the set J arbitrarily to get an ordered set A and let $F = \bar{J}$. Set $h = \prod_{i \in A} p_i$, $k = 1$ and $x^*_i = 1$ for all $i \in A$. Let $B_0 = \{x | x_i = 1 \text{ for } i \in A,\ x_i \geq 0 \text{ for } i \in F\}$. Suppose that r is the last element of A. Go to Step 6b.

1. Determine a binary n-vector $\hat{x}$ such that $\hat{x}_i = x^*_i$ for all $i \in A$ and $\psi(\hat{x}) = 1$. If such an $\hat{x}$ does not exist, go to Step 5; otherwise go to Step 2.

2. Let $h = h + \prod_{i \in A} \Pr\{X_i = \hat{x}_i\} \prod_{i \in F} \Pr\{X_i \geq \hat{x}_i\}$.

3. If $F \neq \phi$, go to Step 4a.

   Otherwise, let $B_k = \{x | x_i = \hat{x}_i,\ i \in A\}$ and $k = k+1$. If the last element of $\hat{x}\langle A\rangle$ is zero, go to Step 5; otherwise go to Step 4b.

4a. Delete from F all elements $i \in F$ with $x_i = 1$ to obtain a reduced set F; add those elements deleted from F, in arbitrary order, to the right of A to obtain an augmented ordered set A. Let $B_k = \{x | x_i = \hat{x}_i \text{ for } i \in A,\ x_i \geq 0 \text{ for } i \in F\}$, and $k = k+1$.

b. Set $x^*_i = \hat{x}_i$ for all $i \in A$ except the last element of A; set the last element of $x^*\langle A\rangle$ equal to 0. Return to Step 1.

5. If $x^*_i = 0$ for all $i \in A$, terminate; otherwise go to Step 6a.

6a. Suppose x^*_r is the rightmost element of $x^*\langle A\rangle$ such that $x^*_r = 1$. Reduce the set A by deleting all elements following (coming after) element r in the ordered set A; augment the set F by including those elements deleted from F.

6b. Set $x_r^* = 0$ and return to Step 1.

Remark 2: B_i's were defined for the sake of explanation; however, they need by no means be recorded. At termination $h = \Pr\{\psi(X) = 1\}$ and $\bigcup_i B_i = \{x|\psi(x) = 1\}$.

It is worth noting that there is room for improvement of the efficiency of the algorithm. A couple of remarks are in order in this regard.

Remark 3: The computation time can naturally be reduced by a good choice of $\hat{x}$ in Step 1. If the determined $\hat{x}$ is minimal in the sense that, for any $x < \hat{x}$ with $x_i = \hat{x}_i$ for all $i \in A$, $\psi(x) = 0$, then unnecessary steps will be saved by modifying Step 3 as follows:

3. If $\hat{x}_i = 1$ for all $i \in F$, let $B_k = \{x|x_i = \hat{x}_i, i = 1,\ldots,n\}$, $k = k+1$ and go to Step 5; otherwise go to Step 4a.

Justification of this change easily follows from the coherent structure function ψ.

Remark 4: Component i is said to be vital if $x_i = 0 \Rightarrow \psi(x) = 0$. Provided that we know which components are vital, the computation time can be saved by making the following changes in Step 6a: Let x_r^* be the rightmost element of $x^*\langle A\rangle$ such that $x_r^* = 1$ and component r is not vital.

Of course, in addition to the above general rules, exploiting the special property of the structure function of the system, additional modifications can be made to reduce an computation effort.

V. EXAMPLE

Consider the probabilistic network shown in Fig. 2, where the numbers beside the arcs denote indices of arcs (components). Associated

with each arc, say are i, are two numbers: y_i, the flow capacity of arc i, and $p_i = \Pr\{X_i = 1\}$. Suppose $y_1 = 2$, $y_2 = 2$, $y_3 = 3$, $y_4 = 1$, $y_5 = 3$, $y_6 = 4$, and $y_7 = 2$. Assuming that the system is said to be functioning if it is possible to route 3 units of flow from the input node to the output node, we want to determine the reliability of this system. Suppose that flow is conserved at every node of the network.

In what follows, let $\bar{x}_i$ and $\underline{x}_i$ represent $x_i = 1$ and $x_i = 0$ respectively and let $\hat{x}$ determined in Step 1 be denoted by a vector $(x_i, i \in A; x_i, i \in F)$. Suppose we start with an initial path vector $(\bar{x}_6, \bar{x}_5, \bar{x}_3; \underline{x}_1, \underline{x}_2, \underline{x}_4, \underline{x}_7)$. The following in the sequence of path vectors $\hat{x}$ determined in Step 1:

$$(\bar{x}_6, \underline{x}_5; \bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_7, \underline{x}_4),$$

$$(\bar{x}_6, \underline{x}_5, \underline{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4, \bar{x}_7).$$

Note that Remarks 3 and 4 have been taken into account when executing the algorithm. Actually we need two more iterations to consider the cases, $(\bar{x}_6, \underline{x}_5, \bar{x}_1, \bar{x}_2, \bar{x}_3, \underline{x}_7; \ldots)$ and $(\bar{x}_6, \underline{x}_5, \bar{x}_1, \underline{x}_2; \ldots)$, to reach the final path vector $(\bar{x}_6, \underline{x}_5, \underline{x}_1; \bar{x}_2, \bar{x}_3, \bar{x}_4, \bar{x}_7)$ from the preceding one. The system has 128 distinct capacity states and the algorithm terminates after 5 iterations. From Step 2 the reliability of the system is given by

$$P_6P_5P_3 + P_6(1-P_5)P_1P_2P_3P_7 + P_6(1-P_5)(1-P_1)P_2P_3P_4P_7.$$

Finally we make a comment on finding $\hat{x}$ in Step 1. For this problem an efficient labeling algorithm [5] can produce $\hat{x}$; furthermore, based on the knowledge of the current $\hat{x}$, subsequent $\hat{x}$ can be obtained by rerouting the flows in those arcs whose states have changed to failed states.

VI. CONCLUSIONS

In implementing the algorithm on a computer, the memory requirement is negligible and the value of h before termination yields the lower bound of the system reliability. Recalling that path vectors have been extensively dealt with to calculate reliability, we can similarly devise another algorithm, in a sense the dual approach of the one given here, for computing the "unreliability" of the system by using cut vectors instead of path vectors.

REFERENCES

[1] R. S. Wilkov, "Analysis and design of reliable computer networks," IEEE Transactions on Communication, vol. COM-20, 1972 June, pp. 660-678.

[2] S. Rai, K. K. Aggarwal, "An efficient method for reliability evaluation of a general network," IEEE Transactions on Reliability, vol. R-27, 1978 August, pp. 206-211.

[3] R. E. Barlow, F. Proschan, Statistical Theory of Reliability and Life Testing, Holt, Rinehart and Winston, New York, 1975.

[4] P. Doulliez, E. Jamoulle, "Transporation networks with random arc capacities," Revue Francaise d'Automatique, Informatique et Recherche Operationnelle, vol. 3, 1972 November, pp. 45-59.

[5] L. R. Ford, D. R. Fulkerson, Flows in Networks, Princeton University Press, Princeton, N.J., 1962.

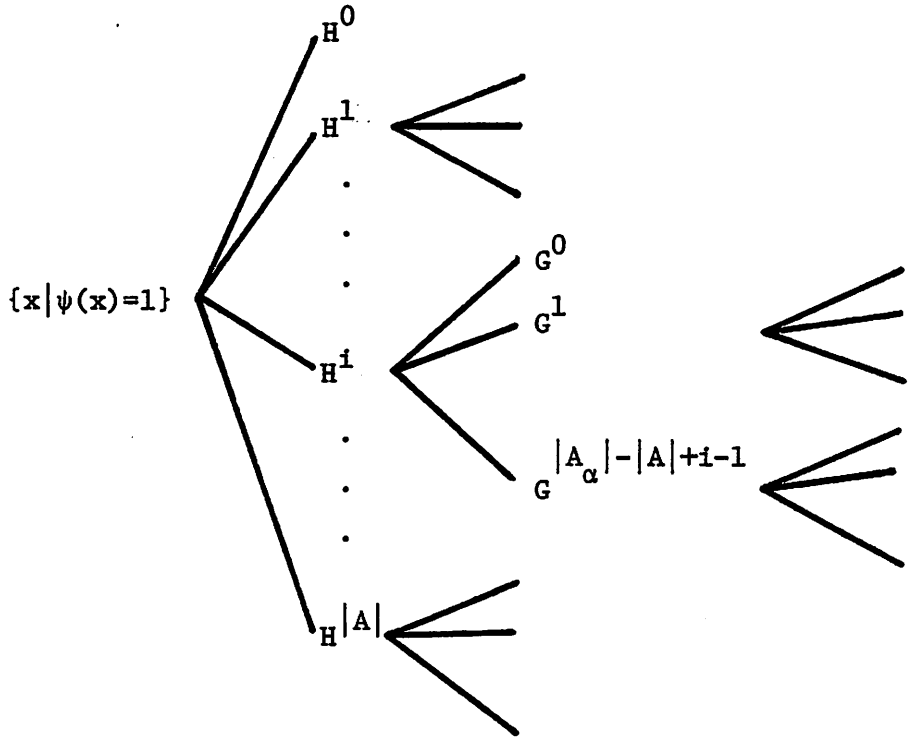


Fig. 1 Partitioning of the set of path vectors

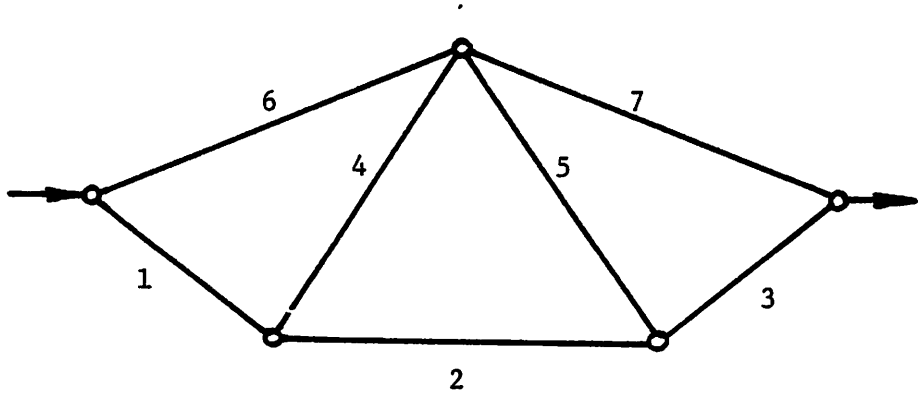


Fig. 2 A flow network for the example

APPENDIX

Justification of the Algorithm

We will prove the validity of the algorithm by examining how B_j's generated in the algorithm are related to those sets shown in Fig. 1.

NOTATION

A_k set A used in Step 1 in the algorithm just prior to entering Steps 3 or 4a to generate B_k.

a_1 $\min\{r \mid |A_r| < |A_1|\}$

a_2 $\min\{r \mid |A_r| < |A_{a_1}|\}$

a_3 $\min\{r \mid |A_r| < |A_{a_2}|\}$

$\vdots$

b_1 $\min\{r \mid |A_r| < |A_{a_i+1}|\}$

b_2 $\min\{r \mid |A_r| < |A_{b_1}|\}$

$\vdots$

It is easily seen that B_0, $\bigcup_{j=1}^{a_1-1} b_j$, $\bigcup_{j=a_1}^{a_2-1} b_j$,... correspond to H^0, H^1, H^2,..., respectively. Without of generality, suppose that $\bigcup_{j=a_i}^{a_{i+1}-1} B_j$ corresponds to H^i. Then B_{a_i}, $\bigcup_{k=a_i+1}^{b_1-1} \bigcup_{j=b_1}^{b_2-1} B_j$, ... correspond to G^0, G^1, G^2,..., respectively. In conclusion, we should be able to verify that B_j, $j = 1,2,...$, correspond to those sets at terminating nodes in the tree, shown in Fig. 1, from which no partitioning is possible.

COMBINATORIAL METHODS FOR RELIABILITY ANALYSIS OF FLOW NETWORKS

Randall Willie

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California 94720

ABSTRACT

We consider a capacitated flow network with multiple supplies and demands. Edges are assumed to be subject to failure and the system "works" if a flow which meets all demands can be constructed using unfailed edges. In Part I, methods are given for determining the family (or an "interesting" subfamily) of all minimal sets of edges whose mutual failure implies system failure, as well as the family or subfamily of all minimal sets of working edges that will insure the system is working. Part II is a guide to use of the Flow Network Analysis Program, a computer code based on the techniques of Part I.

Key words: Reliability, Network Flows, Cut Sets, Path Sets

Research sponsored by the Department of Energy Contract EC-77-S-01-5105.

ACKNOWLEDGEMENT

I would like to thank Professor Felix Wu of the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, for his support, encouragement, and helpful suggestions during preparation of this report and the associated computer program. I also owe him a debt of gratitude for helping me to understand reliability problems associated with electric power grids.

TABLE OF CONTENTS

PART I. METHODOLOGY

A flow network with edge capacities and multiple supplies and demands is a useful model for a wide variety of actual systems, such as transportation and communication network, water resource allocation systems, and electric power grids. This model has been employed most frequently for optimization problems involving such systems, but problems of system reliability can also occasionally be approached by considering flow networks with edges and/or vertices subject to failure.

This report considers the flow network model from a reliability standpoint. We shall be concerned with deriving system reliability minimal cut and path set families, or "interesting" subfamailies, when the system can be modelled as a flow network. However, the methodology is very general and can often be usefully extended to other system configurations.

Section I.1 reviews briefly the concept of a coherent system and introduces most of the notation used in subsequent sections. Coherent systems have been extensively studied [BP], and reliability analysis for such systems is usually easier than for non-coherent systems. The advantages of the methods discussed here are dependent on the coherency property, and extension of these methods to non-coherent systems does not seem worthwhile. Because of the traditional meaning of the terms "cut" and "path" in a network context we will call a reliability cut set a "failure" set and a reliability path set a "success" set.

Section I.2 introduces the flow network model, and Section I.3 present the basic algorithm for deriving families of minimal failure and success sets. Section I.4 is concerned with obtaining "interesting"

subfamilies associated with large flow networks. Some examples to illustrate the methodology are given in Section I.5. Two variants of the basic algorithm that are suitable for computer implementation are discussed in Section I.6. Section I.7 presents computational results for a flow network of moderate size. Finally, Section I.8 suggests an algorithm for construction of dual families, to be used in conjunction with the first method of Section I.6.

I.1 Coherent Systems

Let $E = \{1,\ldots,n\}$ the set of component indices for a system Z, and for $e \in E$, let x_e be the binary state variable for component e; $x_e = 1$, say, if component e is failed, and $x_e = 0$ if e is working. The system state (1 if the system is failed, 0 if the system is working) is then a binary function of the state vector $(x_1,\ldots,x_n)$, and it is fair to assume that the system is working when all components are working and failed when all components are failed; i.e., $x_Z(\underline{x}) = 0$ when $\underline{x} = (0,\ldots,0)$ and $x_Z(\underline{x}) = 1$ when $\underline{x} = (1,\ldots,1)$.

A failure (cut) set is a set of components $F \subseteq E$ such that failure of all components in F implies system failure. The system is said to be coherent if any set containing a failure set is itself a failure set. Coherency is an intuitively reasonable property which means that if the system is failed, failure of additional components will not restore the system to the working state. For a coherent system, we may define a failure set F as minimal if no proper subset of F is also a failure set. It is well known that a given system Z has a unique family $\mathcal{F}_Z = [F_1,\ldots,F_u]$ of minimal failure sets, and the logical function for system failure can be represented as a Boolean sum of products:

$$x_Z(\underline{x}) = \sum_{F \in \mathcal{F}_Z} \prod_{e \in F} x_e \equiv \max_{F \in \mathcal{F}_Z} (\min_{e \in F}(x_e)).$$

There is a parallel manner in which we can characterize the system Z in terms of working, rather than failed, components. A set $S \subseteq E$ is called a success (path) set if the system is working whenever all components in S are working, and S is minimal if no proper subset of S is a success set. A coherent system has a unique family $\mathcal{S}_Z = [S_1, \ldots, S_v]$ of minimal success sets, and the logical function for the system working is

$$y_Z(\underline{y}) = \sum_{S \in \mathcal{S}_Z} \prod_{e \in S} y_e \equiv \max_{S \in \mathcal{S}_Z} (\min_{e \in S}(y_e)),$$

where $y_e \equiv (1-x_e)$ and $z_Z(\underline{y})$ is 1 if and only if the system is working. A family of sets having the property that no set contains another of the family is called a clutter. For a family $\mathcal{R}$ which is not necessarily a clutter, it is convenient to let $m(\mathcal{R})$ (the "minimization of $\mathcal{R}$") denote the clutter obtained by removing from $\mathcal{R}$ sets which contain another set of the family. Families $\mathcal{F}_Z$ and $\mathcal{S}_Z$ are clutters, so for any $\mathcal{F} \supseteq \mathcal{F}_Z$ and $\mathcal{S} \supseteq \mathcal{S}_Z$, clearly $m(\mathcal{F}) = \mathcal{F}_Z$ and $m(\mathcal{S}) = \mathcal{S}_Z$.

There is a simple way to characterize the relationship between families $\mathcal{F}_Z$ and $\mathcal{S}_Z$. Given a family of subsets of a finite set, say E, the blocker or dual family, denoted by $d(\mathcal{R})$, is defined by

$$d(\mathcal{R}) = \left\{ Q \subseteq E \;\middle|\; \begin{array}{l} Q \cap R \neq \emptyset \text{ for each } R \in \mathcal{R} \\ \text{and no subset of } Q \\ \text{has this property} \end{array} \right\}$$

The family $d(\mathcal{R})$ is a clutter, and if $\mathcal{R}$ is also a clutter, $d(d(\mathcal{R})) = \mathcal{R}$. The pair $(\mathcal{R}, d(\mathcal{R}))$ is called a blocking pair. In reference [EF] it is shown that two clutter $\mathcal{R}$ and $\mathcal{Q}$ consisting of subsets of E form a blocking pair if the following condition is satisfied: For any $U \subseteq E$, either U contains a member of $\mathcal{R}$ or E - U contains a member of $\mathcal{Q}$, but not both. The families $\mathcal{F}_Z$ and $\mathcal{S}_Z$ clearly meet this condition, since for any vector of component states the system is either working or failed.

By assuming that each component as well as the system itself is either working or failed, we have adopted the simplest logical scheme for analyzing reliability of a complex system. The coherency concept may be generalized to cases where the system and/or its components are permitted to have multiple performance states [JD, BW]. There does not appear to be any great difficulty in generalizing the ideas of this chapter to allow for multiple performance states, but the two-state assumption leads to clearer exposition and easier computation.

A great deal of literature has addressed the problem of obtaining minimal failure and success set families for coherent systems. However, much of this literature has been directed toward specific system models, such as fault trees and reliability graphs. These models are very natural for complex systems that can be repeatedly decomposed into subsystems having only a few components in common. In such cases, the basic idea is to suggest a procedure for relating the system minimal failure or success set family to families for major subsystems. Recursive application of the procedure to various subsystems then eventually permits construction of the family for the system itself, hopefully without too much computational effort.

Unfortunately, for some systems, such as the flow netowrks we are going to study, there seems to be no appropriate scheme for decomposition into subsystems so that the results of separate analyses of these subsystems will be useful for constructing the minimal failure or success set family for the entire system. Such systems cannot be modelled as reliability graphs or fault trees. The approach here is to propose a more general technique for deriving minimal failure and success set families, or, when these families are too large to obtain in their entirety, subfamilies that are "interesting" to the reliability analyst in a special sense. Though we are specifically concerned with analyzing capacitated flow transportation networks with edges subject to failure, much of the methodology here can be applied to other types of systems.

Our technique depends on selective testing of sets of failed components to determine if the system is working or failed. Several other methods of assessing complex system reliability by selective testing of component state vectors have also been suggested. The "backtracking" algorithm of Ball and Van Slyke [BV] may be used in conjunction with a number of system models, including flow networks, to obtain an expression for system failure as a Boolean sum of logically disjoint products; each product is represented by a "modified" failure set. Jamoulle and Doulliez [JD[employ a testing procedure to partition the space of component state vectors, thereby facilitating computation of the exact failure probability of a flow network.

Minimal failure and success set families can be found in many situations where system size limits the possibility of finding a family of logically disjoint sets associated with the system failure function.

Moreover, for large systems it may not be possible to obtain the exact system failure probability. However, bounds on this probability may be calculated in a number of ways utilizing these families [BP, HU], and for the high component reliability values usually encountered in applications, these bounds are quite good. In addition, these families are useful for evaluating the relative contribution of each system component to reliable system operation [BU]. Other uses for these families have also been proposed, such as minimal cost diagnosis of the state of the system.

I.2 The Flow Network Model

Let (V,E) be a network with vertex set V and edge set E. Suppose $V_s \subset V$ is the set of supply vertices, with supplies $\{a_v, v \in V_s\}$ and $V_d \subset V$ is the set of demand vertices with demands $\{b_v, v \in V_d\}$. Each edge is either directed or undirected and is assumed to have integer upper bound capacity c(e) and lower bound capacity zero. The network of Figure I.2.1 is an example. The notation e/c(e) designates each edge and its capacity. Vertex 1 is the single supply and $V_d = \{3,4,5\}$. All edges are directed except 6 and 7.

For simplicity, only network edges are assumed subject to failure, though the methods to be considered can be readily extended to include vertex failures. The system, call it Z, is taken to be working if and only if there exists a flow which satisfies all demands and which uses only unfailed edges. (Alternative definitions of the system working could also be used. For instance, given weights w_v, $v \in V_d$, the system might be assumed working if and only if the minimum weighted sum of deficiencies at demand vertices is less than some fixed value.)

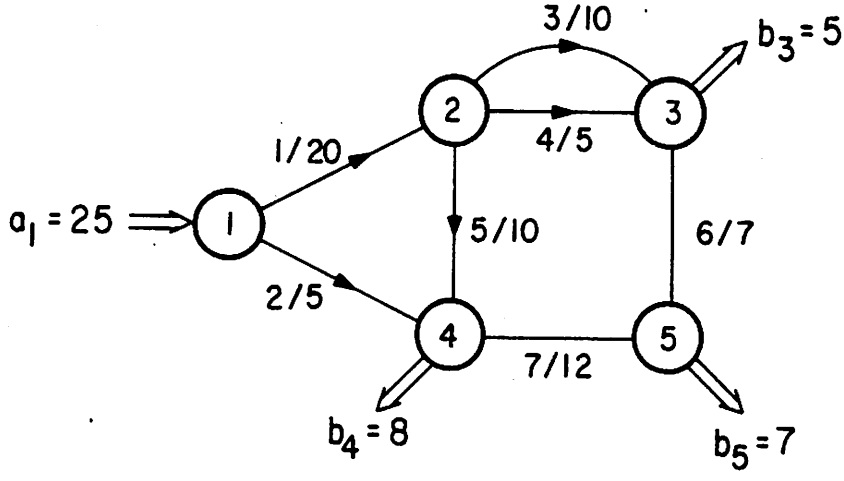


Fig. I.2.1

A set of edges $S \subseteq E$ is then a success set for this reliability model if and only if there is a feasible flow which is zero on edges in $E - S$. Since lower bound capacities are zero on all edges, the system is coherent and thus has unique minimal failure and success set families $\mathcal{F}_Z$ and $\mathcal{S}_Z$. Given a set $U \subset E$, the obvious procedure for checking whether U is a failure set is to attempt a feasible flow using only edges in $E - U$. Perhaps the best known algorithm for constructing a feasible flow is that of Ford and Fulkerson [FF], but more efficient algorithms are available [EK[. If edges in $E - U$ admit a feasible flow, U is not a failure set, but the set $S \subset E - U$ of edges on which the flow is non-zero is clearly a success set.

I.3 Obtaining Minimal Failure and Success Set Families for the Flow Network Model

The Failure and Success Set Family (FSF) Algorithm, which is stated in this section, represents the basic technique we propose for deriving minimal failure and success sets associated with the flow network model. Subsequent sections in Part I discuss some refinements intended to make the technique practical for large networks and computer implementation.

The motivation for this algorithm is quite simple. Suppose a procedure is available that endeavors to construct the complete minimal failure set family $\mathcal{F}_Z$ by testing various subsets $U \subset E$, seeking a feasible flow on edges in $E - U$. At a particular point, collections $\mathcal{F}$ and $\mathcal{S}$ of (not necessarily minimal) failure and success sets have been obtained from previous tests, and let U be the present candidate. We can avoid applying the feasible flow algorithm in two situations. First, U cannot be a failure set if there is at lease one success set $S \in \mathcal{S}$ that is disjoint from U, since each failure set must intersect all

success sets. Secondly, U cannot be a minimal failure set if there is a failure set $F \in \mathcal{F}$ such that F is a proper subset of U. In fact, as the theorem below indicates, it is only necessary to apply the feasible flow algorithm when U is contained in $d(\mathcal{S}) - \mathcal{F}$, that is, when U is a member of $d(\mathcal{S})$ that has not previously been found to be a failure set.

Theorem I.3.1

Let $\mathcal{S}$ be a collection of (not necessarily minimal) success sets. Each $U \in d(\mathcal{S})$ is either a minimal failure set or E - U is a success set not in $\mathcal{S}$.

Proof:

If U is a failure set, it must be minimal. For otherwise, U has a proper subset V which is a minimal failure set, and $V \cap S \neq \emptyset$ for each $S \in \mathcal{S}$. Thus, $U \notin d(\mathcal{S})$, a contradiction. If U is not a minimal failure set, $E - U \notin \mathcal{S}$, since $U \cap S \neq \emptyset$ for all $S \in \mathcal{S}$, but U is disjoint from E - U. ¤

Corollary I.3.1

Let $\mathcal{S}$ be a collection of minimal success sets. Each $U \in d(\mathcal{S})$ is either a minimal failure set or E - U contains a minimal success set not in $\mathcal{S}$.

Based on these considerations, an iterative scheme to construct the minimal failure set family can be outlined. At iteration k, suppose a family $\mathcal{F}^k$ of known minimal failure sets and a family $\mathcal{S}^k$ of known (not necessarily minimal) success sets are available from previous iterations. The k^{th} candidates U in the family $\mathcal{U}^k = d(\mathcal{S}^k) - \mathcal{F}^k$ by attempting a feasible flow on edges in E - U. This leads to a new group of minimal failure sets $\mathcal{F}^{new}$, consisting of sets in $\mathcal{U}^k$ for which a flow does not

exist on edges of E - U, and a new group of success sets $\mathcal{S}^{new}$ from feasible flows constructed. Iteration k + 1 then begins with the collection of known minimal failure sets $\mathcal{F}^{k+1} = \mathcal{F}^k \cup \mathcal{F}^{new}$ and the collection of known success sets $\mathcal{S}^{k+1} = \mathcal{S}^k \cup \mathcal{S}^{new}$. This procedure initially requires a family $\mathcal{S}^o$ containing at least one success set. Of course, one initial success set can always be obtained if the entire network admits a feasible flow.

A formal statement of the method is as follows:

Algorithm FSF

1. Obtain $\mathcal{S}^o$, a family containing at least one success set. $k \leftarrow 0$, $\mathcal{F}^o \leftarrow \emptyset$ and go to 4.
2. $k \leftarrow k + 1$, $\mathcal{S}^{new} \leftarrow \emptyset$, $\mathcal{F}^{new} \leftarrow \emptyset$. Perform steps 2a, 2b, and 2c for each $U \in \mathcal{U}^k$:
   a. Select next $U \in \mathcal{U}^k$. If all sets U have been considered go to 3.
   b. If $\mathcal{S}^{new} \neq \emptyset$ and $U \cap S = \emptyset$ for any $S \in \mathcal{S}^{new}$, go to 2a.
   c. Test U by attempting a feasible flow using edges in E - U. If no feasible flow is possible, include U in $\mathcal{F}^{new}$. If a feasible flow is possible, obtain a success set $S \subseteq E - U$ and include S in $\mathcal{S}^{new}$. Go to 2a.
3. $\mathcal{F}^k \leftarrow \mathcal{F}^{k-1} \cup \mathcal{F}^{new}$, $\mathcal{S}^k \leftarrow \mathcal{S}^{k-1} \cup \mathcal{S}^{new}$. If $\mathcal{S}^{new} = \emptyset$, stop.
4. $\mathcal{U}^{k+1} \leftarrow d(\mathcal{S}^k) - \mathcal{F}^k$. If $\mathcal{U}^{k+1} = \emptyset$, stop. Otherwise go to 2.

(The notation "symbol ← formula" means that indicated by the formula on the right have been performed, the resulting object, whether it be a family, set, or quantity, is to be represented by the symbol on the left.)

Theorem I.3.1 above guarantees that each F added to a family $\mathcal{F}^{new}$

is a minimal failure set. Moreover, no set placed in $\mathcal{F}^{new}$ at iteration k can be in $\mathcal{F}^{k-1}$ because minimal failure sets known from previous iterations are excluded from $\mathcal{U}^k$. Also, the procedure is certainly finite, since each family of success sets $\mathcal{S}^k$, except possibly the last, contains at least one success set not in $\mathcal{S}^{k-1}$. In fact, the argument of Theorem I.3.1 applied inductively shows that sets of $\mathcal{S}^k$ are distinct and that a set U tested in Step 2c at some iteration will not be tested again in any succeeding iteration. Our remaining concern, therefore, is that the procedure does not end prematurely, that is, before all minimal failure sets have been generated. Theorem I.3.2 guarantees that Algorithm FSF provides not only all minimal failure sets, but also all minimal success sets.

Theorem I.3.2

If Algorithm FSF stops at iteration ℓ, the $\mathcal{F}^\ell$ is precisely the family $\mathcal{F}_Z$ of all minimal failure sets and $\mathcal{S}^\ell$ contains the family $\mathcal{S}_Z$ of all minimal success sets.

Proof:

In Step 3 for $k = \ell$, if $\mathcal{S}^{new} = \phi$ then it must be the case that $\mathcal{F}^{new} = \mathcal{U}\phi$. Were the stopping condition in Step 3 ignored, we would obtain $\mathcal{U}^{\ell+1} = \phi$ in Step 4, since

$$\mathcal{U}^{\ell+1} \equiv d(\mathcal{S}^{\ell-1}) - \mathcal{F}^{\ell-1} - \mathcal{F}^{new} = \mathcal{U}^\ell - \mathcal{U}^\ell = \phi$$

Hence, it suffices to consider the situation where the procedure stops with $\mathcal{U}^{\ell+1} = \phi$.

Thus, suppose $\mathcal{U}^{\ell+1} = \phi$ but there is a minimal failure set $F \notin \mathcal{F}^\ell$. Since F has a non empty intersection with every success set, there is

some smallest subset $F' \subseteq F$ that has a nonempty intersection with every set in $\mathcal{S}^\ell$, that is, $F' \in d(\mathcal{S}^\ell)$. Now, if $F' = F$ then $F' \notin \mathcal{F}^\ell$ by assumption, whereas if F' is a proper subset of F, then $F' \notin \mathcal{F}^\ell$ because F' is not a failure set. In either case, $\mathcal{U}^{\ell+1} \neq \phi$, a contradiction. Therefore, $\mathcal{F}^\ell$ contains all minimal failure sets.

To show that $\mathcal{S}^\ell$ contains all minimal success sets, we note that since $\mathcal{U}^{\ell+1} = \phi$, $d(\mathcal{S}^\ell) \subseteq \mathcal{F}^\ell$. Each $F \in \mathcal{F}^\ell$ has a nonempty intersection with all $S \in \mathcal{S}^\ell$, and $\mathcal{S}^\ell$ is a clutter. These facts are sufficient to insure that $d(\mathcal{S}^\ell) = \mathcal{F}^\ell$. Now the clutter $m(\mathcal{S}^\ell)$ satisfies $d(m(\mathcal{S}^\ell)) = d(\mathcal{S}^\ell)$; and from the remarks of Section I.1 $m(\mathcal{S}^\ell) = d(\mathcal{F}^\ell)$, so $m(\mathcal{S}^\ell)$ is precisely the family $\mathcal{S}_Z$ of all minimal success sets. ¤

Derivation of dual families and construction of flows are clearly the major tasks associated with Algorithm FSF. To find the dual family associated with an arbitrary family of sets can sometimes be quite difficult, but it is convenient to discuss this problem in later sections. We will close the present section with some remarks on construction of flows.

The total number of flow constructions that will be attempted by the procedure will equal the sum of the numbers of sets in the final families $\mathcal{F}^\ell$, and $\mathcal{S}^\ell$. The sum will be a minimum if each set of $\mathcal{S}^\ell$ is a minimal success set, so $\mathcal{S}^\ell = \mathcal{S}_Z$. However, given a flow and the set of edges having nonzero flow, a few additional flow constructions will usually be required in order to verify that S is a minimal success set, or to find at least one minimal success set that is a proper subset of S. Of course, if S were found not to be minimal success sets, ideally we would like to find all prime success sets contained in S and include these sets in $\mathcal{S}^{new}$. This suggests a possible recursive application of the algorithm, that is, applying the algorithm again to the subnetwork

defined by edges in S to obtain a final family which includes all minimal success sets that are subsets of S.

On the other hand, insuring that only minimal success sets are included in $\mathcal{S}^{new}$ for each iteration is often not worth the trouble. This is so because available techniques for constructing feasible flows tend to be conservative in the number of edges having nonzero flow, especially in sparse networks. Hence, a success set obtained from such a flow is often minimal or "nearly" minimal in the sense that only a few edges are nonessential to the flow.

Finally, testing sets U in Step 2c of Algorithm FSF is often greatly simplified when the network has a single supply vertex, especially if there are a large number of demand vertices. Any network feasible flow may be associated with an augmented network in the sense of Ford and Fulkerson [FF]. The augmented network has neither supply or demand vertices,and each edge of the original network appears in the augmented network with a forward and reverse capacity. Given a set U to test in Step 2c and the augmented network for the last feasible flow constructed, we need only remove edges in the set U from this network and designate initial and terminal vertices of these edges as supply and demand vertices, with amounts equal to the edge flows. Because the original network has a single supply vertex, a feasible flow on the augmented network under these circumstances is possible if and only if there is a feasible flow on the original network which is zero on edges in the set U. If a feasible flow is possible, we obtain a new augmented network to replace the previous one; otherwise, the previous one is retained. In either case, an augmented network is available to test the next set U. The general scheme of

Algorithm FSF can be applied to a wide variety of models. It can be applied, for example, to the related flow network model where the system is defined to be working when the minimum weighted sum of demand deficiencies is less than a fixed constant. In this case, a set $U \in \mathcal{U}^k$ is tested by first solving a minimum cost flow problem on the subnetwork consisting of edges in E - U. For applications outside the flow network context, the procedure might be useful for coherent systems where no direct technique is available for constructing either the minimal failure or success set family, and testing of individual sets is expensive.

I.4 Obtaining "Interesting" Subfamilies of Failure and Success Sets

Large systems are likely to have so many minimal failure and success sets that no computer methodology would be successful in deriving complete families, even when the system reliability structure allows description by a fault tree or reliability graph. This seems not to have deterred reliability analysts, many of whom are currently formulating fault trees for systems where the number of components subject to failure exceeds 1000. The usual practice in such applications is to seek an "interesting" subfamily of minimal failure sets, a subfamily consisting of sets which in the opinion of the analyst are most likely to be associated with actual failure of the system. Such a subfamily, for example, might consist of all minimal failure sets not having more than a fixed number of components. When component failure probabilities are available, the subfamily could be chosen to consist of all minimal failure sets for which the product of component failure probabilities is greater than some fixed constant.

This approach is suitable in many cases for two reasons. First, components in actual systems tend to be quite reliable, so assuming independent failure probabilities, the chance of more than a few components being failed at the same time is often negligibly small. Secondly, many actual systems do not possess a great deal of built-in redundancy, usually due to component costs, so at least a few minimal failure sets with a small number of components might be expected.

An importance or culling criterion designates each subset of the component set E as either important or unimportant, with the restriction that if the criterion declares a set $E' \subset E$ to be important, it must also declare each subset of E' to be important. For example, a size importance criterion chooses important subsets of E to be those not exceeding some fixed size; a probability product importance criterion designates a subset of E as important if and only if the product of component probabilities exceeds some fixed value. Clearly, many other suitable criteria can be formulated.

Given an importance criterion, let $\mathcal{F}_Z'$ be the subfamily of all important minimal failure sets of the complete family $\mathcal{F}_Z$. A simple modification to Algorithm FSF permits efficient construction of the family $\mathcal{F}_Z'$ rather than $\mathcal{F}_Z$. The family $\mathcal{U}^{k+1}$ of Step 4 is derived as $d_i(\mathcal{S}^k) - \mathcal{F}^k$, rather than $d(\mathcal{S}^k) - \mathcal{F}^k$, where $d_i(\mathcal{S}^k)$ represents the subfamily of important sets of the complete dual family $d(\mathcal{S}^k)$. Any technique for finding a dual family can be adapted to efficiently obtain the subfamily of important sets. Fortunately, the task of deriving this subfamily is usually many times easier than deriving the complete family. Theorems I.4.1 and I.4.2 below correspond to Theorems I.3.1 and I.3.2 of the previous section.

Proofs parallel the earlier proofs through use of the fact that subsets of an important set are important.

Theorem I.4.1

Let $\mathcal{S}$ be a collection of (not necessarily minimal) success sets. Given an importance criterion, each set $U \in d_i(\mathcal{S})$ an important minimal failure set, or E - U is a success set not in $\mathcal{S}$.

Theorem I.5.2

Given an importance criterion, consider Algorithm FSF with $d_i(\mathcal{S}^k)$ replacing $d(\mathcal{S}^k)$ in Step 4. Then if the modified algorithm stops at iteration ℓ, $\mathcal{F}^\ell$ is precisely the family of important minimal failure sets.

The family of success sets $\mathcal{S}^\ell$ produced by the modified algorithm will, in general, no longer contain all minimal success sets. However, rather than being an idle byproduct of deriving the important minimal failure sets, we argue that $\mathcal{S}^\ell$ is, in fact, a family of "interesting" success sets, even when $\mathcal{F}^\ell = \phi$

Theorem I.5.3

Given an importance criterion, consider Algorithm FSF with $d_i(\mathcal{S}^k)$ replacing $d(\mathcal{S}^k)$ in Step 4. Then if the modified algorithm stops at iteration ℓ, $\mathcal{S}^\ell$ contains the complete family of minimal success sets for a system Z_w having the properties:

1. System Z_w is less reliable than system Z.
2. The families of important minimal failure sets for systems Z_w and Z are identical.

Proof:

Since $m(\mathcal{S}^\ell)$ is a clutter, it can be viewed as a minimal success set

family $\mathcal{S}_{Z_w}$ for a system Z_w. This system is evidently less reliable than Z, since each $S \subset \mathcal{S}_{Z_w}$ is either identical with or a superset of a minimal success set for Z.

Now from the fact that $\mathcal{U}^{\ell+1} = \phi$ and each set of $\mathcal{F}^{\ell}$ has a nonempty intersection with all sets of $\mathcal{S}^{\ell}$, $d_i(\mathcal{S}_{Z_w}) = d_i(\mathcal{S}^{\ell}) = \mathcal{F}^{\ell}$. Since $d_i(\mathcal{S}_Z) = \mathcal{F}^{\ell}$ also, $\mathcal{F}^{\ell}$ represents all important minimal failure sets of both systems Z_w and Z.

Of course, the family of important minimal failure sets for system Z may be viewed as a complete family $\mathcal{F}_{Z_b}$ of minimal failure sets for a system Z_b which is more reliable than Z. The families $\mathcal{S}_{Z_w}$ and $\mathcal{S}_{Z_b}$ may be used in various ways to obtain upper and lower bounds on the reliability of system Z, as well as to approximate the contribution of individual components to reliable system operation.

I.5. An Example Using Algorithm FSF

To illustrate the use of algorithm FSF, let us again consider the network of Figure I.5.1. Any feasible flow will yield an initial success set for the family $\mathcal{S}^0$. A suitable flow is represented in Figure I.5.1, where each edge e with nonzero flow f(e) is labelled e/f(e). Thus, let $\mathcal{S}^0 = [\{1,3,5,6,7\}]$ and $\mathcal{F}^0 = \phi$, so $\mathcal{U}^1 = d(\mathcal{S}^0) - \phi = [\{1\},\{3\},\{5\},\{6\},\{7\}]$.

Iteration 1

$\mathcal{F}^{new} = [\{1\},\{5\}]$ $\mathcal{S}^{new} = [\{1,2,4,5,7\},\{1,3,4,5,6\}]$. Sets in $\mathcal{S}^{new}$ are derived from the flows of Figure I.5.2, obtained by testing sets {3} and {7}.

$\mathcal{F}^1 \leftarrow \mathcal{F}^0 \cup \mathcal{F}^{new}$ $\mathcal{S}^1 \leftarrow \mathcal{S}^0 \cup \mathcal{S}^{new}$

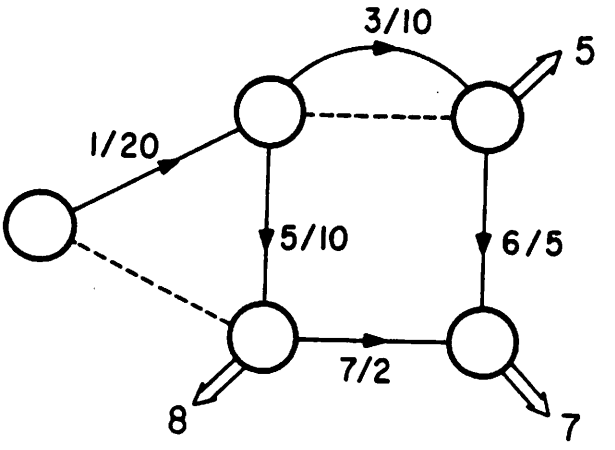
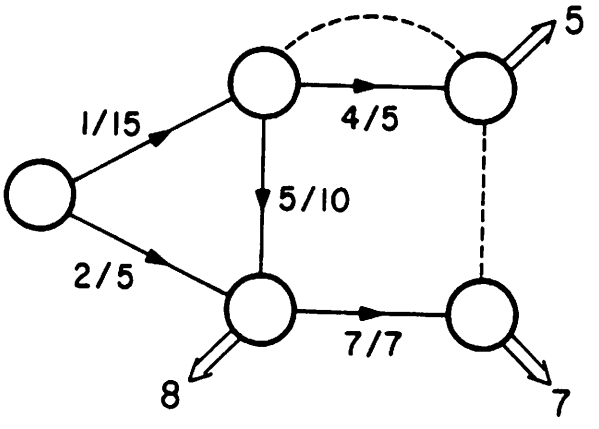


Fig. I.5.1



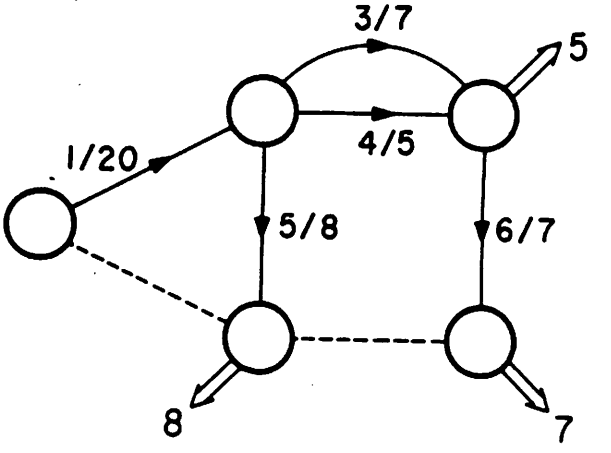


Fig. I.5.2

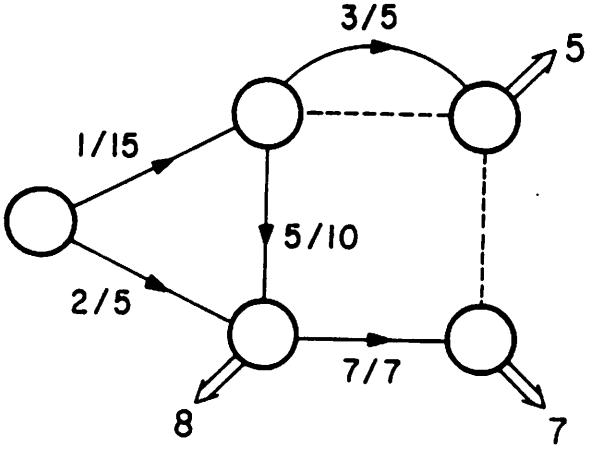


Fig. I.5.3

$U^2 = d(\mathcal{S}^1) - \mathcal{F}^1 = [\{2,3\},\{3,4\},\{3,7\},\{2,6\},\{4,6\},\{6,7\},\{4,7\}]$

Iteration 2

$\mathcal{F}^{new} = [\{2,3\},\{3,4\},\{3,7\},\{2,6\},\{6,7\},\{4,7\}]$

$\mathcal{F}^{new} = [\{1,2,3,5,7\}]$

The single set in $\mathcal{S}^{new}$ is derived from the flow of Figure I.5.3, obtained by testing set {4,6}.

$\mathcal{F}2 \leftarrow \mathcal{F}1 \cup \mathcal{F}new \quad \mathcal{S}2 \leftarrow \mathcal{S}1 \cup \mathcal{S}new$

$\mathcal{U}^3 = d(\mathcal{S}^2) - \mathcal{F}^2 = \phi$

Stop.

The complete minimal failure and success set families are

$\mathcal{F}_Z = [\{1\},\{5\},\{2,3\},\{3,4\},\{3,7\},\{2,6\},\{6,7\},\{4,7\}]$

$\mathcal{F}_Z = [\{1,3,5,6,7\},\{1,2,4,5,7\},\{1,3,4,5,6\},\{1,2,3,5,7\}]$.

For a second example, suppose only minimal failure sets of size 1 are considered important, and $d_i(\mathcal{S}^k)$ represents all size 1 sets of the dual family. Again let $\mathcal{S}^0 = [\{1,3,5,6,7\}]$ and $\mathcal{F}^0 =$, so $\mathcal{U}^1 = d_i(\mathcal{S}^0) - \phi$ $= [\{1\},\{3\},\{5\},\{6\},\{7\}]$.

Iteration 1

$\mathcal{F}^{new} = [\{1\},\{5\}] \quad \mathcal{S}^{new} = [\{1,2,4,5,7\},\{1,3,4,5,6\}]$

Sets in $\mathcal{S}^{new}$ again correspond to the flows in Figure I.5.2 above

$\mathcal{F}^1 \leftarrow \mathcal{F}^0 \cup \mathcal{F}^{new} \quad \mathcal{S}^1 \leftarrow \mathcal{S}^0 \cup \mathcal{S}^{new}$

$\mathcal{U}^2 = d_i(\mathcal{S}^1) - \mathcal{F}^1 = \phi$

Stop.

The failure and success set families obtained this time are then

$\mathcal{F}^1 = [\{1\},\{5\}]$

$\mathcal{F}^1 = [\{1,3,5,6,7\},\{1,2,4,5,7\},\{1,3,4,5,6\}]$

I.6 Modified Versions of Algorithm FSF

The two versions of Algorithm FSF stated in this section are intended to be suitable for computer implementation, and both differ from the original method only in the aspect of deriving dual families. The first version, Algorithm FSF.1, has performed more efficiently than FSF on computer examples analyzed thus far, but this version still relies on an explicit technique for deriving dual families. The second modified version, FSF.2, is self-contained, and dual families are derived implicitly. Computational experience indicates that when the failure and success set families to be derived are large, say more than 150 sets, Algorithm FSF.1 is more efficient than FSF.2 in terms of computation time, whereas, FSF.2 requires less space in main memory.

In order to determine the complete family $\mathcal{F}_Z$ of minimal failure sets, Algorithm FSF clearly depends on the computational feasibility of obtaining $d(\mathcal{S}_Z)$ as well as dual families for various other collections of success sets. This is often possible when $\mathcal{S}_Z$ contains as many as perhaps 200 sets and $\mathcal{F}_Z$ as many as 1000 sets, but the ability to construct $d(\mathcal{S}_Z)$ can certainly be a limiting factor in the analysis of large flow networks. Construction of a dual family falls into the category of problems designated by computer scientists as np-hard, which essentially means that

for any proposed dual algorithm, there is almost assuredly a class of examples where the effort required increases exponentially as the number of elements in the set E. However, it is pointed out in reference [RO] that problems of this type unavailably arise in the reliability analysis of many complex system models.

Not much literature is available concerning algorithms to construct $d(\mathcal{R})$ for an arbitrary family $\mathcal{R}$. Therefore, such an algorithm is discussed in Section I.8. This algorithm has been found to work well in a number of applications, most of them in connection with analysis of fault trees [WI]. The technique is also suited to efficient construction of all sets of the dual family which satisfy an importance criterion.

It is frequently much easier to find a subfamily of $d(\mathcal{R})$ than to construct the complete dual family. (Deriving such a subfamily need not be np-hard.) Rather than deriving the complete family $d(\mathcal{S}^k)$ in Step 4 of each iteration of Algorithm FSF, therefore, let us derive the subfamily consisting of all sets of $d(\mathcal{S}^k)$ not exceeding a particular size. This size, however, changes with each iteration and is not to be confused with a size importance criterion. Algorithm FSF.1, stated below, determines all minimal failure sets of a given size at the same iteration. In Step 4, the notation $d^m(\mathcal{S}^k)$ represents the subfamily of all dual sets consisting of no more than m elements.

Algorithm FSF.1

1. Obtain $\mathcal{S}^0$, a family containing at least one success set. $k < 0$, $\mathcal{F}^0 \leftarrow \phi$ and go to 5.
2. $k \leftarrow k + 1$, $\mathcal{S}^{new} \leftarrow \phi$, $\mathcal{S}^{new} \leftarrow \phi$. Perform Steps 2a, 2b, and 2c for each $U \in \mathcal{U}^k$.

a. Select next U $\mathcal{U}^k$. If all sets U have been considered go to 3.

b. If $\mathcal{S}^{new} \neq \phi$ and $U \cap S = \phi$ for any $S \in \mathcal{S}^{new}$, go to 2a.

c. Test U by attempting a feasible flow on edges E - U. If no feasible flow is possible and U has exactly m elements, include U in $\mathcal{F}^{new}$. If a feasible flow is possible, obtain a success set $S \subseteq E - U$ and include S in $\mathcal{S}^{new}$. Go to 2a.

3. $\mathcal{F}^k \leftarrow \mathcal{F}^{k-1} \cup \mathcal{F}^{new}$ $\mathcal{S}^k \leftarrow \mathcal{S}^{k-1} \cup \mathcal{S}^{new}$.

4. $m \leftarrow m + 1$.

$\mathcal{U}^{k+1} \leftarrow d^m(\mathcal{S}^k) - \mathcal{S}^k$.

If $\mathcal{U}^{k+1} \neq \phi$ go to 2.

5. If $\mathcal{U}^{k+1} = \phi$, stop. Otherwise, fix m to the smallest number of elements of any set in $^{k+1}$ and go to 2.

This algorithm can be justified by an extension of the arguments for algorithm FSF, but using induction on k. If m(k) is the value of m at iteration k ($k \geq 1$), the induction assumption is that $d^{m(k)}(\mathcal{S}^k) - \mathcal{F}^k$ in Step 4 consists only of sets having m(k) elements; in other words, all sets $d^{m(k)}(\mathcal{S}^k)$ having less than m(k) elements are known minimal failure sets.

Experience with Algorithm FSF.1 indicates that Step 5 is rarely executed between the initial and final iterations. By deriving a complete dual family on the final iteration, Step 5 "verifies" that all minimal failure sets have been found.

Like the original method, Algorithm FSF.1 can also be used to find the subfamily of all minimal failure sets satisfying an importance criterion, simply by deriving subfamilies $d_i^m(\mathcal{S}^k)$ and $d_i(\mathcal{S}^k)$ of all important sets, respectively, of $d^m(\mathcal{S}^k)$ and $d(\mathcal{S}^k)$ in Step 4 and 5. Of

course, it is not necessary to perform Step 5 if m has the same value as the size limitation for a size importance criterion; if this is the case at the beginning of Step 5, the procedure should be terminated.

Since efficient computer routines to construct the dual of an arbitrary family are not widely available, there is an advantage to stating a version of Algorithm FSF that does rely on explicit construction of dual families. Some additional notation is required.

Given families $\mathcal{G}$ and $\mathcal{H}$ of subsets of a finite set E, let the family $m(\mathcal{G}/\mathcal{H})$ (the "minimization of $\mathcal{G}$ by $\mathcal{H}$") consist of all sets of $\mathcal{G}$ that do not contain a set of $\mathcal{H}$. The simplest way to determine $m(\mathcal{G}/\mathcal{H})$ is to compare each set of $\mathcal{G}$ with sets in $\mathcal{H}$ having an equal or smaller number of elements. The number of set comparison is then bounded above by the product of the number of sets in $\mathcal{G}$ and the number in $\mathcal{H}$; so for large $\mathcal{G}$ and $\mathcal{H}$, construction of $m(\mathcal{G}/\mathcal{H})$ will involve some measure of computational effort. A more sophisticated technique might reduce the effort somewhat, but probably by not more than half in most situations.

Also, given subsets Q and R of E, the sharp product family, denoted by $Q \# R$, is defined as $[Q \cup \{e\} | e \in R]$. (The terminology and notation are adopted from reference [RE].) Thus, for example, $\{1,2,3\} \# \{3,4,5\} = [\{1,2,3\},\{1,2,3,4\},\{1,2,3,5\}]$. Sets of $Q \# R$ need not be distinct. We extend this notation by replacing Q with a family $\mathcal{Q}$ of subsets of E, writing $\mathcal{Q} \# R$ to denote the union of families $\bigcup_{Q \in \mathcal{Q}} Q \# R$.

For a family $\mathcal{R}$ and a set $R_{new} \notin \mathcal{R}$, it is easy to see that $d(\mathcal{R} \cup [R_{new}])$ consists of distinct sets of $d(\mathcal{R}) \# R_{new}$ that do not contain another set of this latter family. In fact, if $d(\mathcal{R})$ is partitioned into two families $\mathcal{Q}_d$ and $\mathcal{Q}_{nd}$ such that sets in $\mathcal{Q}_d$ are disjoint from R_{new} and sets

in $\mathcal{Q}_{nd}$ have a nonempty intersection with R_{new}, then

$$d(\mathcal{R} \cup [R_{new}]) = \mathcal{Q}_{nd} \cup m(\mathcal{Q}_d \# R_{new}/\mathcal{Q}_{nd}).$$

This relation suggests a computationally useful procedure for obtaining $d(\mathcal{R} \cup [R_{new}])$ when the family $d(\mathcal{R})$ is available.

Based on these remarks, modification of Algorithm FSF is now fairly straightforward and involves changing only Step 4. In Algorithm FSF.2 below, Steps 4a and 4b iteratively construct the family $d(\mathcal{S}^{k-1} \cup \mathcal{S}^{new}) - \mathcal{F}^k$ from the family $d(\mathcal{S}^{k-1}) - \mathcal{F}^{k-1}$ by considering successive sets $S \subset \mathcal{S}^{new}$.

Algorithm FSF.2

1. Obtain $\mathcal{S}^0$, a family containing at least one success set $k \leftarrow 0$, $\mathcal{F}^0 \leftarrow \phi$, and go to 4.
2. $k \leftarrow K + 1$, $\mathcal{S}^{new} \leftarrow \phi$, $\mathcal{F}^{new} \leftarrow \phi$. Perform Steps 2a, 2b, and 2c for each $U \in \mathcal{U}^k$.
   a. Select next $U \in \mathcal{U}^k$. If all sets U have been considered, go to 3.
   b. If $\mathcal{S}^{new} \neq \phi$ and $U \cap S = \phi$ for any $S \in \mathcal{S}^{new}$, go to 2a.
   c. Test U by attempting a feasible flow using edges in E - U. If no feasible flow is possible, include U in $\mathcal{F}^{new}$. If a feasible flow is possible, obtain a success set $S \subseteq E - U$ and include S in $\mathcal{S}^{new}$. Go to 2a.
3. $\mathcal{F}^k \leftarrow \mathcal{F}^k \cup \mathcal{F}^{new}$, $\mathcal{S}^k \leftarrow \mathcal{S}^{k-1} \cup \mathcal{S}^{new}$. If $\mathcal{S}^{new} = \phi$, stop.
4. $\mathcal{H} \leftarrow \mathcal{U}^k - \mathcal{F}^{new}$.

   Perform Steps (a) and (b) for each $S \in \mathcal{S}^{new}$:

   a. Select next $S \in \mathcal{S}^{new}$. If all $S \in \mathcal{S}^{new}$ have been considered, $\mathcal{U}^{k+1} \leftarrow \mathcal{H}$ and go to 2.

b. $\mathcal{H}_d \leftarrow [H \in \mathcal{H}\ H \cap S = \phi]$

$\mathcal{H}_{nd} \leftarrow [H \in \mathcal{H}\ H \cap S \neq \phi]$

If $\mathcal{H}_d = \phi$, go to (a). Otherwise $\mathcal{H} \leftarrow \mathcal{H}_{nd} \cup m(\mathcal{H}_d \# S / \mathcal{H}_{nd} \cup \mathcal{F}^k)$

If $\mathcal{H} = \phi$, stop. Otherwise, go to (a).

I.7 A Computer Example Using Algorithms FSF.1 and FSF.2

The network of Figure I.7.1 is taken from Reference [JD]. In that reference, several edges are permitted to have multiple random capacities, so we have selected the maximum capacity given there for our simpler two-state model. The integer upper bound capacity and the failure probability for each edge is shown in Figure I.7.1. These probabilities are also somewhat different than in [JD], where six edges are assumed not subject to failure.

A FORTRAN computer program implementing Algorithms FSF.1 and FSF.2 was used to analyze this network on a CDC 6400 computer. The first minimal failure and success set families for the network under a variety of demand conditions. For a given run, all demands were set to a fixed percentage of the demand amounts (rated demands) given in Figure I.7.1. Another computer routine was used to obtain the exact probabilities of system failure based on failure and success set families derived, assuming independence of edge failures.

Each line of Table I.7.1 summarizes the results of a particular run. The percentage of rated demands appears at the left, followed first by the number of sets in $\mathcal{S}_Z$ and the maximum size of any of these sets, and then by the number of sets in $\mathcal{F}_Z$ and their maximum size. Exact failure probabilities $P_{\mathcal{S}}$ and $p_{\mathcal{F}}$ were obtained respectively by utilizing

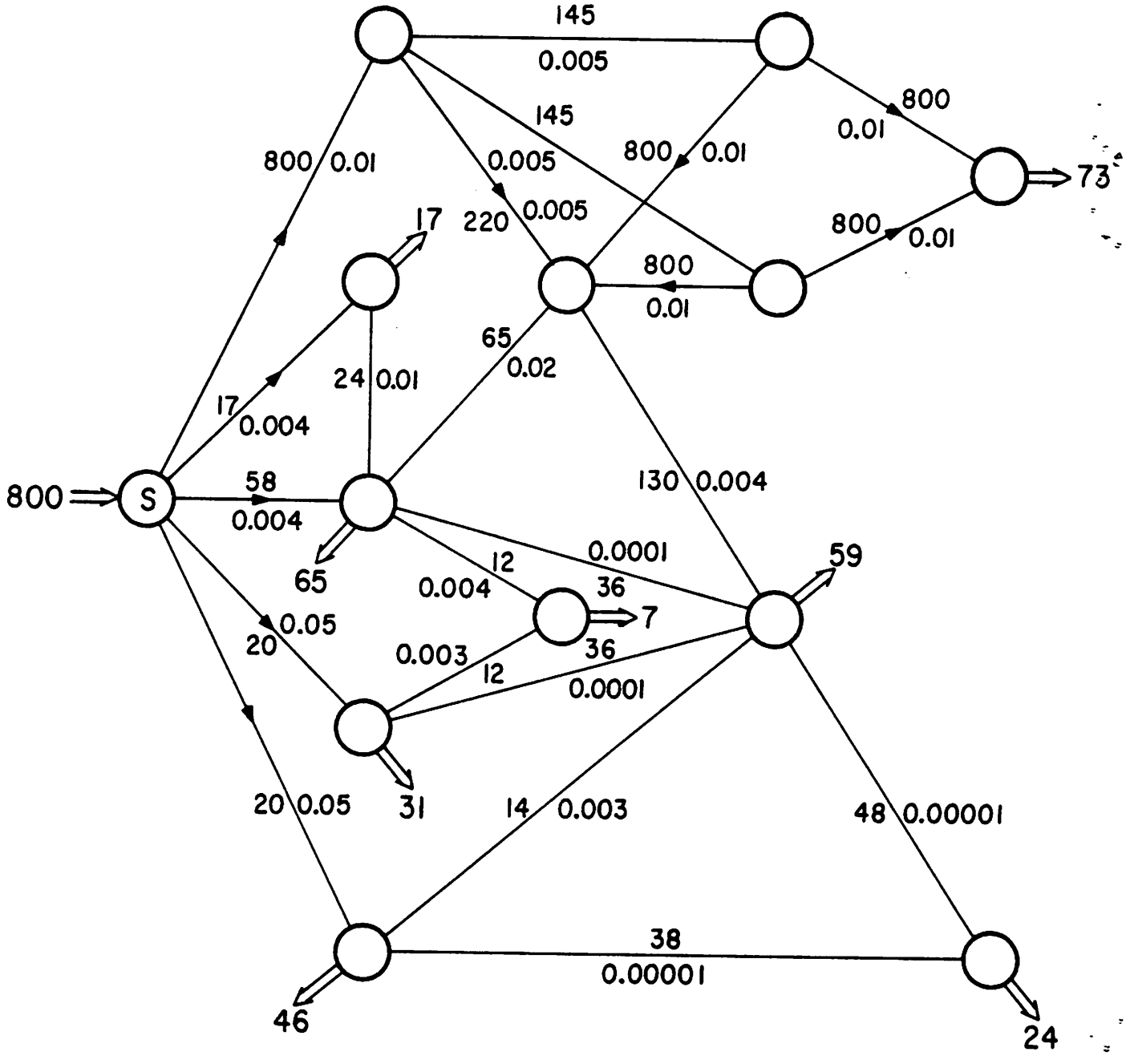

145
0.005
800
0.01
145
800
0.01
0.01
73
800
0.01
0.005
0.005
17
220
0.005
800
0.01
800
0.01
65
0.02
24
0.01
17
0.004
800
S
58
0.004
130
0.004
65
0.0001
12
0.004
36
59
7
0.05
20
0.003
36
12
0.0001
31
20
0.05
14
0.003
48
0.00001
38
0.00001
46
24

Fig. I.7.1

families $\mathcal{S}_Z$ and $\mathcal{F}_Z$. Finally, the CDC6400 CPU time required for deriving $\mathcal{S}_Z$ and $\mathcal{F}_Z$ is given. This time does include the time to calculate exact probabilities, but in all cases probability calculations from a failure or success set family required less than 1.5 CPU seconds. The network is not feasible for 115% of rated demands, and no runs were made for less than 90% of rated demands (because of a limited computer budget). All runs for Table I.7.1 used Algorithm FSF.1.

% Rated Demands	$\#\mathcal{F}_Z$	Max Length	$\#\mathcal{S}_Z$	Max Length	$P_{\mathcal{F}}$	$p_{\mathcal{S}}$	CPU Seconds
110	23	2	12	18	.128	.128	3.2
105	27	3	24	18	.119	.119	6.0
100	30	3	30	17	.0840	.0840	8.1
95	26	4	52	16	.0639	.0639	20.1
90	30	4	84	16	.0119	.0119	71.8

Table I.7.1

Derivation of Complete Families $\mathcal{F}_Z$ and $\mathcal{S}_Z$

A second series of runs was made with a probability product importance criterion, and important sets were required to have a product of component probabilities exceeding 10^{-6}. The family $\mathcal{S}$ of Table I.7.1 is in this case the family of "interesting" success sets associated with $\mathcal{F}'_Z$. Thus, according to the discussion of Section I.4, values $p_{\mathcal{S}}$ are upper bounds for the system failure probability, whereas $p_{\mathcal{F}}$, are lower bounds. Results for 95% and 90% of rated demands indicate a considerable savings in CPU time over derivation of the complete families $\mathcal{F}_Z$ and $\mathcal{S}_Z$. All runs in this series were again done with Algorithm FSF.1.

% Rated Demands	$\#\mathcal{F}'_Z$	Max Length	$\#\mathcal{S}$	Max Length	$p_{\mathcal{F}'}$	$p_{\mathcal{S}}$	CPU Seconds
110	21	2	11	19	.128	.128	2.1
105	21	3	14	18	.119	.119	2.5
100	24	3	17	18	.0840	.0840	2.9
95	20	3	31	18	.0639	.0639	4.7
90	23	3	43	18	.0119	.0120	6.6
80	17	3	42	18	.0116	.0117	6.0
60	11	3	29	16	.0107	.0107	3.9
40	8	2	24	15	.0103	.0103	4.9
20	7	2	23	13	.0103	.0103	4.3

Table I.7.2

Derivation of Probability Product Important Subfamilies $\mathcal{F}'_Z$ (Critical Value 10^{-6})

For edge probability values precisely 10 times the values shown in Figure I.7.1, a recalculation of exact system failure probabilities was done based on complete families $\mathcal{F}_Z$ and $\mathcal{S}_Z$ for the percentages of rated demands listed in Table I.7.1. These probabilities are given in Table I.7.3. (For percentages 100, 95, and 90, the slight discrepancy between values $p_{\mathcal{F}}$ and $p_{\mathcal{S}}$ is due to round-off error that accumulates during calculation of these quantities.)

Finally, a series of runs was made with a probability product importance criterion using edge probability values 10 times the Figure I.7.1 values. The critical value in this case was 10^{-3}, and all runs were done with Algorithm FSF.2. Table I.7.4 summarizes results of these runs.

Demands	$p_{\mathcal{F}}$	$p_{\mathcal{S}}$
110	.823	.823
105	.803	.803
100	.740	.739
95	.699	.698
90	.316	.314

Table I.7.3

Exact System Failure Probabilities (Using Edge Probabilities x 10) % Rated

% Rated Demands	$\# \mathcal{F}'_Z$	Max Length	$\# \mathcal{S}$	Max Length	$p_{\mathcal{F}'}$	$p_{\mathcal{S}}$	CPU Seconds
110	20	2	11	18	.823	.823	1.5
105	20	3	16	18	.803	.804	2.1
100	23	3	17	18	.740	.739	2.1
95	20	3	26	17	.703	.700	3.8
90	25	4	40	17	.316	.324	6.5
80	18	4	31	18	.216	.229	4.9
60	11	4	28	15	.131	.145	3.4
40	7	2	20	15	.124	.130	2.9
20	7	2	25	13	.124	.131	2.9

Table I.7.4

Derivation of Probability Product
Important Subfamilies $\mathcal{F}'_Z$
(Edge Probabilities X 10; Critical Value 10^{-3})

I.8 Construction of Dual Families

Given a family $\mathcal{R} = [R_1,\ldots,R_n]$ of subsets of a finite set E, let $\mathcal{R}_j \equiv [R_1,\ldots,R_j]$ for $j \leq n$. The previous section indicates we can construct the families $d(\mathcal{R}_1),\ldots,d(\mathcal{R}_n)$ by the following method:

1. $j \leftarrow 1$, $\mathcal{Q}^1 \leftarrow [\{e\}|e \in R_1]$. If $n = 1$, stop.
2. Partition sets of $\mathcal{Q}^j$ into families $\mathcal{Q}^j_d$ and $\mathcal{Q}^j_{nd}$, where sets of $\mathcal{Q}^j_d$ are disjoint from R_{j+1} and sets of $\mathcal{Q}^j_{nd}$ have a nonempty intersection with R_{j+1}.
3. $\mathcal{Q}^{j+1} \leftarrow \mathcal{Q}^j_{nd} \cup m(\mathcal{Q}^j_d \# R_{j+1}/\mathcal{Q}^j_{nd})$

   $j \leftarrow j + 1$. If $j = n$, stop. Otherwise, go to 2.

The final family $\mathcal{Q}^n$ produced by this procedure is, of course, the dual family for $\mathcal{R}$. This procedure is suitable for computer applications when either $\mathcal{R}$ or $d(\mathcal{R})$ consists of a small number of sets. However, when both $\mathcal{R}$ and $d(\mathcal{R})$ consist of more than perhaps 100 sets, significant computational effort is required for set comparisons implicit in deriving the families $m(\mathcal{Q}^j_d \# R_{j+1}/\mathcal{Q}^j_n)$.

In this section, we propose a more efficient technique for deriving $d(\mathcal{R})$ when $\mathcal{R}$ and/or $d(\mathcal{R})$ has a large number of sets. This technique endeavors to minimize the number of set comparisons necessary to detect and discard redundant sets as the dual family is being derived. The general strategy is to consider a partition of the set E and construct the dual family by combining subsets from various members of the partition; hence, the method will be referred to as the Partition Subsets (PS) Algorithm.

From here on, assume the sets of $\mathcal{R}$ have a fixed order, $\mathcal{R} = [R_1,\ldots,R_n]$. The efficiency of the algorithm depends to some extent on this order,

but a rule for determining some kind of "optimal" order is not available. Later, we will consider a good heuristic rule for arranging these sets.

For any nonempty set $A \subseteq E$, let the cover of A, denoted by C(A), consist of indices of sets in $\mathcal{R}$ that have a nonempty intersection with A, that is, $C(A) \equiv \{i | A \cap R_i \neq \phi, R_i \in \mathcal{R}\}$. $C(\phi)$ is defined to be ϕ. For example, if $\mathcal{R} = [\{1,2\},\{1,3,4\},\{2,3,4\},\{3,4,5,6\}]$, $C(\{2\}) = \{1,3\}$ and $C(\{3,4\}) = \{2,3,4\}$. Clearly, for $D \in d(\mathcal{R})$, it must be the case that $C(D) = N$, where N is taken to be the set of consecutive integers $\{1,2,\ldots,n\}$. Moreover, for $D \in d(\mathcal{R})$ and any $e \in D$, $C(D-\{e\})$ must be a proper subset of N, by the requirement that no proper subset of D may intersect all sets of $\mathcal{R}$.

A set $B \subseteq E$ will be called a basic set if for each $e \in B$, $C(B-\{e\})$ is a proper subset of C(B). Also, given a set $A \subseteq E$, let $\mathcal{B}(A)$ be the family of all subsets of A that are also basic sets. $\mathcal{B}(A)$ cannot be empty, since it contains at least the single element subsets of A. Though $\mathcal{B}(A)$ could have as many as $2^{|A|} - 1$ sets, this is not often the case for families $\mathcal{R}$ that arise in applications, such as minimal failure and success set families associated with complex systems. If A has, say, 12 or less elements, $\mathcal{B}(A)$ can usually be constructed in a small amount of computer time, and a simple procedure to accomplish this is outlined in the next paragraph. It might be observed that $d(\mathcal{R}) = [B \in \mathcal{B}(E) | C(B) = N]$, but this observation does not yield a suitable algorithm for finding a dual family, since in many applications of interest, the set E is large enough to make construction of $\mathcal{B}(E)$ impossible.

Let $A \subseteq E$ consist of elements $e_1,\ldots,e_v$. One way to obtain $\mathcal{B}(A)$ is as follows: Let $\mathcal{B}'$ consist of just the set $\{e_1\}$ and generate $\mathcal{B}^2,\ldots,\mathcal{B}^v$

according to the rule

$$\mathcal{B}^u \leftarrow [\{e_u\}] \cup [B \cup \{e_u\} | B \in \mathcal{B}^{u-1},\ C(B \cup \{e\}) \neq C(B)].$$

Clearly, $\mathcal{B}(A) \subseteq \bigcup_{u=1}^{v} \mathcal{B}^u$, but some sets in this union of families may not be basic. To remove these non-basic sets, first partition $\bigcup_{u=1}^{v} \mathcal{B}^u$ so that any two sets B and B' of this union belong to the same member of the partition if and only if C(B) = C(B'). Then discard any set that property contains another set in the same member of the partition. The sets constitute $\mathcal{B}(A)$.

As an illustration, for the example family $\mathcal{R}$ = [{1,2},{1,3,4}, {2,3,4},{3,4,5,6}] and A = {1,2,3,4}, we obtain

$\mathcal{B}^1$ [{1} , C({1}) = {1,2}

$\mathcal{B}^2$ [{2} , C({2}) = {1,3}
{1,2} , C({1,2}) = {1,2,3}

$\mathcal{B}^3$ [{3} , C({3}) = {2,3,4}
{1,3}, C({1,3}) = {1,2,3,4}
{2,3}, C({2,3}) = {1,2,3,4}
{1,2,3}, C({1,2,3}) = {1,2,3,4}

$\mathcal{B}^4$ [{4}, , C({4}) = {2,3,4}
{1,4}, C({1,4}) = {1,2,3,4}
{2,4}, C({2,4}) = {1,2,3,4}
{1,2,4}, C({1,2,4}) = {1,2,3,4}

Partitioning $\bigcup_{u=1}^{4} \mathcal{B}^u$ yields the families [{1}], [{2}], [{1,2}], [{3},{4}], [{1,3},{2,3},{1,2,3},{1,4},{2,4},{1,2,4}]. Sets {1,2,3} and {2,3,4} must be discarded because they contain another set in the same member of the partition. Thus, $\mathcal{B}(A)$ = [{1},{2},{1,2},{3,4},{1,3},{2,3},{1,4},{2,4}].

The sets for which we shall require all basic subsets are sets A_i, $1 \leq i \leq n$ defined by $A_i \equiv \{e \in R_i \mid e \notin R_j \; 1 < j < i\}$. Thus, A_i consists of elements of the set $R_i \in \mathcal{R}$ that do not appear in any preceding set of $\mathcal{R}$. Some of the sets A_i may be empty, but Algorithm PS explicitly ignores these sets. For the example family $\mathcal{R} = [\{1,3\},\{1,3,4\},\{2,3,4\},\{3,4,5,6\}]$, it is easy to see that $A_1 = \{1,2\}$, $A_2 = \{3,4\}$, $A_3 = \phi$, and $A_4 = \{5,6\}$; in addition, $\mathcal{B}(A_1) = [\{1\},\{2\},\{1,2\}]$, $\mathcal{B}(A_2) = [\{3\},\{4\}]$, and $\mathcal{B}(A_4) = [\{5,6\}]$. Note that the collection $[A_1,\ldots,A_n]$ forms a partition of E. The fact which motivates the PS Algorithm is that for any $D \in d(\mathcal{R})$, there is a unique partition of D, say $[B_1,\ldots,B_m]$, such that for each i, $1 \leq i \leq m$, either $B_i = \phi$ or B_i is a set in (A_i). (Evidently, $A_i = \phi \Rightarrow B_i = \phi$, but B_i can be empty even when A_i is not.) As an example, $\{1,2,5\} \in d(\mathcal{R})$ for $\mathcal{R}$ above, and this set may be written as $\{1,2\} \cup \phi \cup \phi \cup \{5\}$, where $\{1,2\} \in \mathcal{B}(A_1)$ and $\{5\} \in \mathcal{B}(A_4)$.

The general idea underlying Algorithm PS is thus to form appropriate unions of sets from distinct families $\mathcal{B}(A_i)$, for $A_i \neq \phi$. Let $[A_1,A_{i_2},\ldots,A_{i_s}]$ be the group of A_i's that are nonempty. The method considers these sets successively in s stages. Assume at stage q, there is on hand a table of sets $\mathcal{T}^{q-1}$ with each set $T \in \mathcal{T}^{q-1}$ composed of elements from $A_1,\ldots,A_{i_{q-1}}$ and having the properties:

(1) For each $T \in \mathcal{T}^{q-1}$, $\{1,\ldots,i_{q-1}\} \subseteq C(T)$ but $C(T) \neq N$.

(2) If D is any member of $d(\mathcal{R})$ such that D contains elements not in $A_1,\ldots,A_{i_{q-1}}$, then $D \cap \left(\bigcup_{p=1}^{q-1} A_{i_p}\right) \in \mathcal{T}^{q-1}$.

Condition (1) simply requires any $T \in \mathcal{T}^{q-1}$ to have a nonempty intersection with each of the sets $R_1,\ldots,R_{i_{q-1}}$ but not all sets of $\mathcal{R}$. Condition (2) requires $\mathcal{T}^{q-1}$ to contain, for any $D \in d(\mathcal{R})$, the largest proper subset of D that can be formed from elements in the sets $A_1,\ldots,A_{i_{q-1}}$.

At iteration q, we find a family $\mathcal{D}^q$, which contains each $D \in d(\mathcal{R})$ such that $D \not\subseteq \bigcup_{p=1}^{q-1} A_{i_p}$ but $D \subseteq \bigcup_{p=1}^{q} A_{i_p}$. Essentially, construction of D is completed by appending a subset of A_{i_p} to a proper subset of D contained in the family $\mathcal{T}^{q-1}$. We also generate a family $\mathcal{T}^q$ having properties (1) and (2) above for q replacing q - 1. Formally, $\mathcal{T}^q$ and $\mathcal{D}^q$ are obtained through operations described by the following symbolism:

$$\mathcal{T}^q \leftarrow [T \in \mathcal{T}^{q-1} \mid \{1,\ldots,i_q\} \subseteq C(T)]$$

$$\cup \left[T \cup B, T \in \mathcal{T}^{q-1}, B \in \mathcal{B}(A_{i_q}) \;\middle|\; \begin{array}{l} C(T \cup B) \neq N \\ C(T \cup B) \neq C(T) \\ \{1,\ldots,i_q\} \subseteq C(T \cup B) \end{array} \right]$$

$$\mathcal{D}^q \leftarrow [T \cup B, T \in \mathcal{T}^{q-1}, B \in \mathcal{B}(A_{i_q}) \; C(T \cup B) = N]$$

Algorithm PS actually utilizes more stringent conditions than $\{1,\ldots,i_q\} \subseteq C(T)$ and $\{1,\ldots,i_q\} \subseteq C(T \cup B)$ in assigning sets T and $T \cup B$ to the family $\mathcal{T}^q$, but it is convenient to delay introducing these alternate conditions. A final version of the algorithm will not be stated until near the end of this section. However, all remarks below may easily be extended to apply to the final version.

In practice, $\mathcal{T}^q$ and $\mathcal{D}^q$ can be constructed at the same time by considering successively each $T \in \mathcal{T}^{q-1}$. First, if T has a nonempty intersection with all of the sets $R_1,\ldots,R_{i_q}$, then T is included in $\mathcal{T}^q$. Next, each set $B \in \mathcal{B}(A_{i_q})$ is considered in conjunction with T. If $C(T \cup B) = N$, the T and B together have a nonempty intersection with all sets of $\mathcal{R}$, so the set $T \cup B$ is formed and placed in $\mathcal{D}^q$. If $C(T \cup B) \neq N$, it is determined whether T and B together have a nonempty intersection with more sets of $\mathcal{R}$ than T alone. If so, $T \cup B$ is formed and included in $\mathcal{T}^q$.

Proposition A.1 guarantees that all sets $D \in d(\ )$ are obtained.

Proposition I.8.1

Let the above iterative procedure start with $\mathcal{J}^1 = [B \in \mathcal{B}(A_1)\ C(B) = N]$ and $\mathcal{D}^1 = [B \in \mathcal{B}(A_1) | C(b) = N]$. Then $d(\mathcal{R}) \subseteq \bigcup_{q=1}^{s} \mathcal{D}^q$.

Proof:

For any $D \in d(\mathcal{R})$, we can write $D = \bigcup_{p=1}^{q} B_p$, where $1 \leq q \leq s$, B_1 and B_q are nonempty with $B_1 \in \mathcal{B}(A_1)$, $B_q \in \mathcal{B}(A_{i_q})$, and each B_p, $1 < p < q$ is either empty or a subset in $\mathcal{B}(A_{i_p})$. Using the fact that each subset of D is a basic subset, a simple induction argument establishes that $T = \bigcup_{p=1}^{q-1} B_p$ is a member of $\mathcal{J}^{q-1}$. Since $C(T \cup B_q) = N$, $D = T \cup B_q$ appears in $\mathcal{D}^q$. ∎

Since $d(\mathcal{R}) \subseteq \bigcup_{q=1}^{s} \mathcal{D}^q$ and each set in this union of families has a nonempty intersection with all sets of $\mathcal{R}$, $d(\mathcal{R}) = m(\bigcup_{q=1}^{s} \mathcal{D}^q)$, the clutter obtained by discarding sets of $\bigcup_{q=1}^{s} \mathcal{D}^q$ which contain another set of the family. Proposition I.e.2 allows us to conclude further that $d(\mathcal{R}) = \bigcup_{q=1}^{s} m(\mathcal{D}^q)$, and this fact usually leads to a considerable reduction of effort in removing redundant sets of $\bigcup_{q=1}^{s} \mathcal{D}^q$. Assume again that the procedure begins with $\mathcal{J}^1$ and $\mathcal{D}^1$ as in Proposition I.8.1.

Proposition I.8.2

Let $D \in \mathcal{D}^r$ for some r, $1 \leq r \leq s$ but suppose $D \notin d(\mathcal{R})$. Then there is a $D' \in \mathcal{D}^r$ such that $D' \subset D$.

Proof:

Evidently, there is a $D' \in \bigcup_{p=1}^{s} \mathcal{D}^p$ such that $D' \in d(\mathcal{R})$ and $D' \subset D$. We will show that $D' \in \mathcal{D}^r$. As in the previous proof, we can write $D = \bigcup_{p=1}^{r} B_p$, where $B_1 \in \mathcal{B}(A_1)$, $B_r \in \mathcal{B}(A_{i_r})$ and each B_p, $1 < p < r$ is either empty or a subset in $\mathcal{B}(A_{i_p})$. Similarly, we can write $D' = \bigcup_{p=1}^{q} B'_p$,

where $q \leq r$ since $D' \subset D$.

Let us assume $q < r$ and show this gives a contradiction; thus, $q = r$ and $D' \in \mathcal{D}^r$. Now $B'_p \subseteq B_p$ for $1 \leq p \leq q$, because $[A_1,\ldots,A_{i_s}]$ is a partition of E and $D' \subset D$. Hence,

$$C(\bigcup_{p=1}^{q} B'_p) = \bigcup_{p=1}^{q} C(B'_p) \subseteq \bigcup_{p=1}^{q} C(B_p) = C(\bigcup_{p=1}^{q} B_p)$$

Now $C(D') = N$, so $C(\bigcup_{p=1}^{q} B_p) = N$. But then $\bigcup_{p=1}^{q} B_p$ cannot appear in the family $\mathcal{J}^q$, and since $q \leq r - 1$, $\bigcup_{p=1}^{r-1} B_p$ is not a set in any of the families $\mathcal{J}^q,\ldots,\mathcal{J}^{r-1}$. But this contradicts the assumption that $D = \bigcup_{p=1}^{r} B_p$ is contained in $\mathcal{D}^r$.

The iterative procedure will be most successful when the number of sets in successive families $\mathcal{J}_q$ does not increase too rapidly. Ideally, we would like each $T \in \mathcal{J}^q$ to be a basic set, but to insure this requires a number of set comparisons at each iteration. Algorithm PS has performed well in computer applications without eliminating nonbasic sets from the families $\mathcal{J}^q$; however, it seems prudent to consider elimination of these sets as a possible variation to the method. Removal of nonbasic sets may be accomplished by the same technique suggested earlier in connection with obtaining a family $\mathcal{B}(A)$; namely, by first partitioning the family $\mathcal{J}^q$ so all sets T with the same C(T) appear in the same member of the partition. Redundant sets are then discarded from each member.

The arrangement of sets of $\mathcal{R}$ clearly influences the rate of increase in the size of successive families $\mathcal{J}^q$. Heuristically, it seems desirable that the sets A_{i_q} be small, so the families $\mathcal{B}(A_{i_q})$ will not contain a large number of sets. Also, the set $\{1,i_2,\ldots,i_s\}$ of indices of nonempty A_i's should be well distributed among the set $\{1,\ldots,n\}$ of subscripts of

$R_3 = \{2,3,4\}$, the set having $A_3 = \phi$. In general, for any set R_i with $A_i = \phi$, there is a maximum $i_q < i$ such that A_{i_q} is nonempty and intersects R_i. Intuitively, iteration q is the "last chance to cover" R_i, since any $T \cup B$ produced at this iteration which does not intersect R_i cannot be the core or a set that intersects all sets of $\mathcal{R}$. Let L_q consist of indices i of all sets R_i such that iteration q is the last chance to cover R_i. Specifically,

$$L_q = \left\{ i \geq i_q \;\middle|\; \begin{array}{ll} A_{i_q} \cap R_i \neq & \text{but} \\ A_{i_r} \cap R_i = & \text{for } r > q \end{array} \right\}$$

L_q is nonempty, since $i_q \in L_q$. The earlier requirements that $1,\ldots,i_q \subseteq C(T)$ and $1,\ldots,i_q \subseteq C(T \cup B)$ for sets T and $T \cup B$ to be included in $\mathcal{J}^q$ may now be replaced by the more restrictive conditions that $L_q \subseteq C(T)$ and $L_q \subseteq C(T \cup B)$. For the previous example, we now find $L_1 = \{1\}$, $L_2 = \{2,3\}$, $L_4 = \{4\}$; thus, the set $\{1\}$ is eliminated from $\mathcal{J}^2$.

The Partition Subsets Method is stated formally below. For simplicity, $\mathcal{J}^0$ is taken to be the family consisting only of the empty set, so on the first iteration, $\phi \cup B = B$ for each $B \in \mathcal{B}(A_1)$.

Algorithm PS

1. Initialization.

   Arrange sets of $\mathcal{R}$. Let R_1 be the smallest set in $\mathcal{R}$. After $R_1,\ldots,R_{i-1}$ have been chosen, let R_i be a remaining set having the smallest number of elements not in $R_i,\ldots,R_{i-1}$.

   Determine the family $[A_1,\ldots,A_{i_s}]$ of nonempty A_i's and the corresponding family $[L_1,\ldots,L_{i_s}]$.

$q \leftarrow 0$, $\mathcal{J}^0 \leftarrow [\{\phi\}]$, $\mathcal{D} \leftarrow \phi$

2. $q \leftarrow q + 1$ and generate $\mathcal{B}(A_{i_q})$

3. $\mathcal{J}^q \leftarrow [T \in \mathcal{J}^{q-1} \; L_r \subseteq C(T)]$

$$\cup \left[T \cup B, T \in \mathcal{J}^{q-1}, B \in \mathcal{B}(A_{i_q}) \;\middle|\; \begin{array}{l} C(T \cup B) \neq N \\ C(T) \neq C(T \cup B) \\ L_q \subseteq C(T \cup B) \end{array} \right]$$

$$\mathcal{D}^q \leftarrow [T \cup B, T \in \mathcal{J}^{q-1}, B \in \mathcal{B}(A_{i_q}) \,|\, C(T \cup B) = N]$$

4. $\mathcal{D} \leftarrow \mathcal{D} \cup m(\mathcal{D}^q)$

5. (Optional) Eliminate nonbasic sets from $\mathcal{J}^q$.

6. If $\mathcal{J}^q = \phi$ or $q = s$, stop. Otherwise, go to 2.

Note that Algorithm PS can also be used efficiently to find a subfamily of all important sets of $d(R)$ when an importance criterion is specified. In this case, sets arising during construction of $\mathcal{B}(A_{i_q})$ and $\mathcal{J}^q$ are simply discarded if they are not important.

In conclusion, we consider some aspects of a computer implementation of Algorithm PS. Each set V in the family $\mathcal{J}^q$ or $\mathcal{B}(A_{i_q})$ is most conveniently represented in computer memory as a set pair (v,C(v)). Furthermore, the set C(v) should have a "Boolean" format; that is, C(v) should occupy a fixed number of computer words with each bit position in a word identified with a particular set index in R. Presence or absence of an index in C(v) is then indicated by a 1 or 0 in the bit position corresponding to that index. With this format, set unions and intersections can be performed using computer AND and OR operations. If each element in the set V is an integer, the same storage format may be used for V. Alternatively, V can be represented in "vector" format, where the first

word for V contains the number of set elements, and succeeding locations contain the individual elements, ordered to facilitate set union and comparison operations.

Only a portion of each family $\mathcal{J}^q$ needs to be retained in main memory at any time; the rest can reside on magnetic disk. Since the family $\mathcal{J}^{q-1}$ is dispensible at the conclusion of iteration q, only two sequential files are required: Blocks of sets for $\mathcal{J}^{q-1}$ are read into main memory from one file as blocks of sets for $\mathcal{J}^q$ are prepared and written out on the other. For iteration q + 1, the roles of the two files are interchanged. In this case, it is usually not practical to eliminate all nonbasic sets from $\mathcal{J}^q$, though the technique suggested above for eliminating these sets may be applied locally to each block of sets, immediately before the block is written onto disk.

A subroutine package, mostly in FORTRAN, has been prepared for Algorithm PS and used in a wide variety of applications. The code consists of about 800 FORTRAN source statement. Several small assembler language routine perform operations involving sets stored in Boolean format.

PART II. FLOW NETWORK ANALYSIS PROGRAM

FNAP is a general purpose computer program for flow network reliability analysis employing algorithms FSF.1 and FSF.2 of Part I. The bulk of the program consists of about 3000 FORTRAN statements, segmented into a driver routine and about 25 subroutines. Assembler code performs several simple operations that cannot be done in the context of standard FORTRAN. The FORTRAN portion of FNAP is compatible with nearly all FORTRAN compilers, but assembler routine packages are currently available only for CDC 6600/7600 and IBM 360/370 series machines. However, versions of these routines can easily be prepared in any assembler language according to the specifications given in Section II.7.

Considerable effort has been expended to insure that FNAP will be easy to use. The input format is direct and unified, and input data is completely checked for correctness and consistency. Error messages are detailed, allowing the user to promptly identify problems involving program input or execution. Also, an ample number of comment cards are interspersed with the FORTRAN source code to relate program operation to statements of algorithms FSF.1 and FSF.2 in Part I. Finally, FNAP has been extensively tested for reliable operation.

Sections II.1 through II.4 describe program input and output; Section II.5 discusses the general procedure for implementing FNAP at a computer installation.

II.1 General Input Structure

The smallest logical units of input data are called program instructions, each of which is confined to a single 80-column punched card.

Program instructions are classified according to three major groups: edge definition, option, and execution. Edge definition instructions specify a flow network for analysis. These are always read first by the program, and if they are free of errors, a representation of the flow network is stored in main memory. Errors in network specification are messaged and cause processing to terminate.

One or more option instructions may follow network specification, and information provided by these instructions is checked and stored. Options allow the user to (1) specify supply and demand vertices and amounts, (2) choose the methodology for analysis, (3) modify the input flow network, (4) enable construction of an important subfamily of minimal failure sets, and (5) control program printed and punched output.

The next card to be read after the option group is an execution instruction, which instructs the program to begin obtaining failure and success set families. Option instructions affect the processing initiated by an immediately following execution instruction, and this processing will be referred to as a run. When a run is completed, the program reinitializes all memory locations except those associated with the input network and with supply and demand information, so a group of options and an execution instruction for a new run may immediately follow the execution instruction for the previous run.

The input data package for FNAP therefore has this general form:

network specification instructions

run 1 option instructions

run 1 execution instruction

run 2 option instructions

run 2 execution instructions

⋮

run n option instructions

run n execution instruction

For convenience, the same basic 80-column card format is used for all instructions and consists of eight fields across the width of the card. A particular instruction, however, will typically utilize only information punched in certain of these fields. Field 1 is composed of card columns 1-8. Fields 2 through 8 consist, respectively, of columns 11-18, 21-28, 31-38, 41-48, 51-58, 61-68, and 71-78. Data entered in columns outside these fields is ignored, with the exception of column 10 on edge definition instructions.

The entry in field 1 is either an edge name or an instruction name, left-justified in the field. Internally, FNAP identifies network edges with positive integers, but the analyst is allowed the luxury of choosing names to replace these edge numbers on program input and printed output. Edge names may consist of any combination of eight or less characters. Instruction names are fixed strings of eight or less characters and are discussed in Sections II.3 and II.4.

Depending on the instruction, an entry in field 2 is either a positive integer or a decimal number in FORTRAN E or F format. The entry may appear anywhere in field 2, except for an E-format decimal number, which must be right-justified.

Entries in fields 3 through 8 are either edge names or vertex names, left-justified in these fields. Like edge names, vertex names may consist of any combination of eight or less characters.

II.2 Flow Network Specification

The input flow network is specified through a series of edge definition instructions arranged in any order and followed by a card with the string "ENDGRAPH" left-justified in field 1. An edge definition instruction is provided for each network edge. Field 1 of this instruction contains the edge name. The integer edge capacity appears anywhere in field 2. Fields 3 and 4 contain the names, respectively, of the initial and terminal vertices for the edge. If column 10 of the instruction is blank, the edge is assumed to be directed from the initial to the terminal vertex, so the edge can only carry flow in this direction. If the character "u" occupies column 10, the program permits the edge to carry flow in either direction.

An an example of network specification, we consider again the network of Figure I.2.1, redrawn in Figure II.2.1 with an unimaginative choice of edge and vertex names.

All edges are assumed to be directed with the exception of E6 and E7. The network is specified as follows (where each line is to be interpreted as a single card):

```
Col. 1     11          21          31
↓          ↓           ↓           ↓
E1         20          V1          V2
E2         5           V1          V4
E3         10          V2          V3
E4         5           V2          V3
E5         10          V2          V4
E6        U7           V3          V5
E7        U12          V4          V5
```

II.3 Supply and Demand Options and the Execution Instruction (SUPPLY, DEMAND, *XEQ)

For the initial program run, the card with "ENDGRAPH" in field 1 is followed by a group of option instructions and an execution instruction. The execution instruction consists only of the string "*XEQ" left-justified in field 1 of the card; all other fields are blank.

The simplest group of options that are necessary for the initial run are those that designate supply and demand vertices and the flow amounts available or required at these vertices. Each SUPPLY instruction gives the supply amount available at a particular network vertex. Field 1 contains the instruction name ("SUPPLY"), field 2 gives the positive integer amount, and field 3 contains the vertex name. Fields 4 through 8 are blank. Two SUPPLY instructions may not refer to the same vertex. DEMAND instructions have the same format and designate demand vertices and amounts required. DEMAND instructions must also contain distinct vertex names; however, the same vertex name may appear on both a SUPPLY and DEMAND instruction.

As an example, the supply and demand amounts indicated in Figure II.3.1 can be utilized for an initial run through the following instruction sequence:

```
SUPPLY    25        V1
DEMAND    5         V3
DEMAND    8         V4
DEMAND    7         V5
*XEQ
```

For this run, the program determines the complete minimal success and failure set families associated with the network of Figure II.2.1:

Minimal Failure Set Family

1	E5	
2	E1	
3	E6	E7
4	E4	E7
5	E3	E7
6	E3	E4
7	E2	E6
8	E2	E3

Success Set Family

1	E1	E3	E4	E5	E6
2	E1	E2	E3	E5	E7
3	E1	E2	E4	E5	E7
4	E1	E3	E5	E6	E7

CPU Time For Run .289 Sec

SUPPLY and DEMAND are the only option instructions that may affect runs other than those for which they are specified. When SUPPLY options are absent for a given run, supply information remains unchanged from the preceding run. However, if any SUPPLY instruction appears, no supply information is retained from the preceding run; thus, all supply vertices and amounts must be respecified, even if the supply amount at only a single vertex is to be altered from the preceding run. The situation is precisely the same for DEMAND options. As an illustration, suppose an additional run is to follow the run of the previous example, with the demand amount at vertex V3 changed from 5 to 7. Appropriate instructions for these two runs are:

```
SUPPLY    25        V1
DEMAND    5         V3
DEMAND    8         V4
DEMAND    7         V5
*XEQ
DEMAND    7         V3
DEMAND    8         V4
DEMAND    7         V5
*XEQ
```

FNAP checks supply and demand totals as soon as the *XEQ instruction is encountered. If either total is zero, or if demand exceeds supply, the program will terminate the run and immediately begin processing options for the next run. Errors in individual SUPPLY or DEMAND instructions will also cause the run to be terminated.

II.4 Additional Option Instructions

The flexibility of FNAP for flow network analysis is due to the option instructions we now discuss. None of these instructions is strictly required for a run, but the analyst will nearly always wish to include one or more. Unlike SUPPLY and DEMAND, the options only affect runs for which they are provided.

The input sequence for run options is entirely arbitrary, with the exception that instructions assigning edge probability values for a product importance criterion must appear together as a group. Moreover, there is no restriction on the number of times a particular option is included for a run. However, data from a given instruction will replace conflicting data from a prior instruction of the same type in the input sequence. Syntax errors in these options cause termination of the run.

The options are discussed according to four functional categories.

II.4.1 Network and Supply-Demand Modification (FAILED, WORKING, SPERCENT, DPERCENT)

FAILED and WORKING instructions consist of the instruction name in field 1 and edge names in any or all of the fields 3 through 8. Field 2 is blank. When a FAILED instruction is included for a run, edges whose names are listed on the instruction are assumed to be absent from the network for the duration of the run. On the other hand, edges specified on a WORKING instruction are treated as present in the network but not subject to failure, so these edges will not appear in any minimal failure or success set determined for the run. As an example, for the input network of Figure II.3.1, consider a run specified by the following instructions:

```
SUPPLY    25        V1
DEMAND    5         V3
DEMAND    8         V4
DEMAND    7         V5
FAILED              E2        E4
WORKING             E1        E3        E5        E7
*XEQ
```

The network for analysis by this run is that of Figure II.4.1 with all edges perfect but E6. For this case, E6 is both the only minimal failure set and the only minimal success set.

When there is a feasible flow that uses only edges listed on WORKING instructions, no minimal failure set can be determined. If such a flow is encountered during processing, FNAP prints a message and terminates the run.

Use of instructions SPERCENT and DPERCENT greatly simplifies the task of analyzing the flow network under a variety of supply and demand conditions. These instructions contain the instruction name in field 1

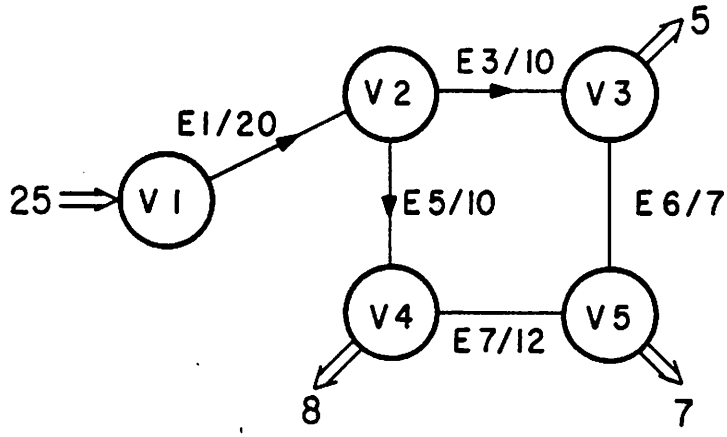


Fig. II.4.1

and a positive integer, say p, in field 2. The SPERCENT instruction indicates that all supply amounts which would otherwise be in effect for the run are to be multiplied by the factor p/100. If p is 120, for example, the amount available at each supply vertex will be increased to the largest integer amount not exceeding 120 percent of the value specified in field 2 of the last SUPPLY instruction for that vertex (which may or may not be an instruction for the current run). The DPERCENT instruction alters amounts required at demand vertices in precisely the same manner.

To illustrate the use of these instructions, suppose we wish to analyze the network of Figure II.2.1 for supply and demand amounts that are 200 percent, approximately 150 percent, and finally 100 percent of the amounts indicated there. A suitable sequence of instructions for these three runs is the following:

```
SUPPLY    25        V1
DEMAND    5         V3
DEMAND    8         V4
DEMAND    7         V5
SPERCENT  200
DPERCENT  200
*XEQ
SPERCENT  150
DPERCENT  150
*XEQ
*XEQ
```

Note that no options are present for the final run; supplies and demands for that run will then be the same as specified by SUPPLY and DEMAND instructions for the initial run.

II.4.2 Methodology Specification (FSF1, FSF2, WRKFILES)

Instructions FSF1 and FSF2 allow the analyst to explicitly specify which of the algorithms of Section I.6, FSF.1 or FSF.2, is to be utilized in determining failure and success set families. Each instruction consists only of the instruction name in field 1 of a card; other fields are blank. If neither instruction is included for the run, FNAP automatically chooses Algorithm FSF.2, unless a MAXSIZE option is specified, in which case FSF.1 is chosen.

When the success and failure set families to be derived are small, say less than 50 sets, the computation time and main memory requirements of the two methods should be roughly equivalent. However, if these families are large, Algorithm FSF.1 will usually be more efficient in terms of computation time, whereas FSF.2 will have the advantage in terms of main memory utilization. Algorithm FSF.1 employs the technique of Section I.8 for construction of dual families, but use of this technique at a particular iteration depends on convenient access to success sets found at all previous iterations, so these sets must be retained in main memory. Algorithm FSF.2, on the other hand, only requires access to success sets determined at the current iteration, so success sets found at previous iterations may either be retained in a compact format in memory or written on magnetic disk.

The WRKFILES option informs the program that sequentially organized file space on magnetic disk is available for use as working storage. The option consists only of the instruction name in field 1. If this option is included for a run which utilizes Algorithm FSF.1, FORTRAN file numbers 10,11, and 12 must be assigned to magnetic disk. These

files will then be available for use as working storage by subroutines that implement the dual algorithm. In connection with Algorithm FSF.2, the WRKFILES options causes the family of new success sets found at each iteration to be written on FORTRAN file 10 at the conclusion of the iteration, thereby freeing memory space that would otherwise be required to retain these sets. Only FORTRAN file number 10 need be assigned in this case.

If WRKFILES has not been included for a run which must be terminated because of insufficient working space in main memory, a message may be printed suggesting that memory could have been supplemented by magnetic disk storage. In this situation, it is reasonable for the analyst to try a rerun with a WRKFILES instruction.

II.4.3 Importance Criteria (MAXSIZE, IMPORT)

The MAXSIZE option indicates that only minimal failure sets not exceeding a fixed size are to be determined for a run. The instruction name is placed in filed 1 and the integer size in field 2; other fields are blank.

A subfamily of important minimal failure sets can also be determined, based on a product importance criterion. In this case, the subfamily will consist of all minimal failure sets such that the product of the probability values for set edges exceeds some fixed critical value, say c. This option requires a group of cards to specify probability values for network edges and the critical value c. The first card of the group contains only the option name "IMPORT" in field 1; other fields are blank.

Cards which assign edge probability values follow this initial card.

These cards have filed 1 blank, a positive decimal value between 0 and 1 in field 2, and edge names in fields 3 through 8. The value in field 2 may be in FORTRAN E or F format and must contain a decimal point. A F-format item, such as .5 or .001 may appear anywhere within the field, but E-format items, such as 1.25E-2 or .1E-1 must be right-justified. An edge whose name appears on one of these cards is assigned the probability value in field 2 of that card. FNAP assigns a default value of 1 to edges not represented on any of the cards.

The card that must terminate the group has the string "LIMIT" left-justified in field 1 and the decimal critial value c in field 2. Fields 3 through 8 are blank.

As a simple illustration of the above options, suppose for the example network of Figure II.3.1, only minimal failure sets not exceeding size 2 are desired, and failure sets containing either edge E3 or E4 are arenot of interest. Suitable instructions for such a run are

```
SUPPLY    25        V1
DEMAND    5         V3
DEMAND    8         V4
DEMAND    7         V5
MAXSIZE   2
IMPORT
          .4        E3        E4
LIMIT     .5
*XEQ
```

II.4.4 Control of Printed and Punched Output (STATUS, FSTATUS, DSTATUS, PUNCH, NOPRINT)

All of the instructions in this category consist only of the instruction name in field 1; fields 2 through 8 are blank.

The STATUS option yields information on the progress of determining failure and success set families. Statistics are provided for each

iteration of Algorithm FSF.1 or FSF.2, giving number of new failure and success sets determined for the iteration, the total number of failure and success sets determined up to that point, and the number of sets in the dual family derived at the end of the iteration. Below is a sample of this output for network of Figure 5.

```
Iteration

  Calls to subroutine /sset/          7          Time in /sset/    .036
  Number of new failure sets          6          Maximum length      2
  Number of new success sets          1          Maximum length      5

  Number of failure sets              8          Maximum length      2
  Number of success sets              5          Maximum length      6
  Dual
  Max number of work table sets       5                    Time    .074
  Number of dual sets                 8          Maximum length      2
```

Output for the STATUS option also records the amount of CPU time expended in checking for failure sets and determining the dual family. Subroutine SSET is responsible for classifying a set as a failure set or obtaining a disjoint success set, so most of the computational effort for this routine involves flow construction.

The FSTATUS option causes subroutine SSET to print out each feasible flow obtained for a run. For a particular flow, amounts and edge names are listed for all edges which carry nonzero flow, as in this example:

```
Feasible flow  (edge flow/edge name)
     20/E1          10/E3          10/E5          5/E6          2/E7
```

For an undirected edge, the flow amount might be negative, which indicates that flow passes from the terminal to the initial vertex established for the edge on the edge definition instruction.

When Algorithm FSF.1 is employed for a run, DSTATUS causes the subroutine package for deriving dual families to provide data on the sizes of various tables associated the algorithm discussed in Section I.8. For Algorithm FSF.2, DSTATUS provides data on the procedure of Step 4 of FSF.2 for extending the dual family. Information provided by the DSTATUS option might be of some use in estimating computer time and memory requirements for large networks, but for most applications, the analyst will probably choose not to include this option.

The PUNCH option allows failure and success sets to be punched on 80-column cards for input to other programs. FORTRAN file number 7 should be assigned to the card punch (or magnetic disk) if this instruction is used. Edges associated with the input network are represented on punched output by positive integers, consequtive numbers 1 to the total number of edges. The first group of punched cards for a run gives the correspondence between edge names and edge numbers. The initial card of the group has a FORTRAN format (5HNAMES,16) where the single integer field contains the total number of edges in the input network. On the remaining cards edge numbers are paired with edge names, with up to five pairs appearing on a card in the format (5(15,3H - ,2A4)).

Output for the success set family occupies the next group of cards, whose initial card has a (5HSSF ,216) format, containing in the integer fields, respectively, the run number and number of success sets. Following this leader card, each success set starts on a separate card with a (1615) format and may continue onto additional cards with the same format. On the first card for a set, field 1 always contains the number of edges in the set. These edges are represented in fields 2

through 16 on the first card and 1 through 16 on subsequent cards.

The final group of cards represents the failure set family. This output has the same structure as for the success set family, except the string "SSF" of the header card is replaced by "FSF."

Finally, the analyst may sometimes wish to obtain punched output but suppress printed output for large failure and success set families. In such cases, the NOPRINT option should accompany the PUNCH option.

II.5 Program Implementation

FNAP is designed for use on most general purpose computers. The code has been carefully prepared to ensure program logic is tight and efficient, and subroutines for minor tasks such as sorting and searching use good standard algorithms, such as given in [KN]. The FORTRAN portion of the program conforms to ANSI specifications, except for array subscripts, which are apt to consist of expressions using two or more simple FORTRAN integer variables with addition, subtraction, and multiplication operations. Most FORTRAN compilers allow such expressions.

Main memory work space for FNAP is confined to one single subscripted integer array, denoted by the FORTRAN name IA. Storage in this array is dynamically allocated for maximum efficiency in use of main memory. Because flow networks of approximately the same size may differ considerably in their structure, it is difficult to state even roughly how large IA should be to accommodate analysis of a network with some given number of edges. The analyst should make IA as large as feasible for the environment in which the program is implemented; for instance, if the program is required to execute in a fixed partition of computer main memory, then the object code length plus storage for IA should fill the partition.

If the environment is such that program use becomes more inconvenient as storage requirements increase, an initial length of IA should be chosen perhaps 500 times the maximum number of edges in any network to be analyzed; this length may then be increased as necessary.

Implementation of FNAP is accomplished according to the following steps:

1. The desired dimension of the array IA should be set in the declarative statement for this array near the beginning of the main program. Since the code must be capable of determining when storage requirements exceed availability, the length of the array must be provided for internal program use. This is done by initializing the variable IASIZE through a FORTRAN DATA statement, which also appears near the beginning of the main program.
2. The first executable statement in the main program assigns a positive integer value to the variable LWORD. This value should be set to the length of a computer word less one.
3. A small number of program statements associated with input and output operations, such as testing for end of input file, may not be acceptable to all FORTRAN compilers. These are clearly flagged in the program listing and should be modified if necessary.
4. When accessed by other routines, the REAL function TIME returns the amount of elapsed time since the beginning of the computer job. The proper form of the subroutine CALL statement in function TIME may depend on the particular computer installation, and this statement should be modified accordingly.
5. The group of assembler language routines should be chosen to correspond

to both the computer and the FORTRAN compiler used. Three groups of assembler routines are supplied with FNAP: for use with (1) CDC 6600/7600 machines and RUN compiler linkage convention, (2) CDC 6600/7600 machines and FTN compiler linkage convention, or (3) IBM 370 machines (G or H compiler linkage convention). To implement FNAP on other machines, six small assembler routines must be prepared according to specifications given in the following section.

II.6 Specification for Assembler Language Routines

The routines discussed in this section are all very simple, and the largest should not require more than 25 statements in any assembler language.

1. CCM (IW1, IW2, ITEST):

CCM logically compares the contents of computer words IW1 and IW2 and returns the result of the comparison in word ITEST. If contents of IW1 and IW2 are identical, then ITEST will contain a 0; otherwise the value in ITEST depends on the highest bit in which the words differ. When this bit is 1 in IW1 and 0 in IW2, ITEST is returned as 1; in the reverse situation ITEST is returned as -1.

2. ORM (IW1, IW2, IOR):

The contents of word IOR returned by this routine is simply a logical OR of words IW1 and IW2.

3. ANDM (IW1, IW2, IAND):

ANDM returns in word IAND the result of a logical AND of IW1 and IW2.

4. PUTM (LV, IV, IW):

When PUTM is accessed, IV is an array of successive words containing positive integer values in increasing order. No value exceeds the number of bits in a computer word. The location LV contains a positive integer representing the length of IV. The function of PUTM is to place a 1 in each bit position of word IW numbered by an integer in array IV; other bit positions are set to 0. As an example, suppose the computer word length is 16, and PUTM is accessed with 4 in LV, and IV(1) through IV(4) contain, respectively, 2, 5, 7, and 16. On return, IW then contains the bit pattern "1000000001010010." The bit numbering for this example is increasing from right-to-left.

5. GETM (IW, LV, IV):

GETM performs the reverse operation of PUTM. On return, IV is a vector of consecutive words containing bit numbers for all bits that are 1 in word IW. These integers are in increasing order in IV, and the number of integers in this vector is returned in LV. GETM may be accessed with IW having 0's in all bit positions, in which case LV is returned with an integer value of 0.

6. BCM (IW, NBITON):

BCM returns in NBITON the count of bit positions containing 1 in word IW. The value in NBITON is thus an integer between 0 and the number of bits in a computer word.

REFERENCES

[BV] Ball, M. and Van Slyke, R., "Backtracking algorithms for reliability analysis," Annals. of Discrete Math., 1 (1977) 49-64.

[BP] Barlow, R. and Proschan, F., Statistical Theory of Reliability and Life Testing, Holt, Rinehart, and Winston, New York, 1975.

[BW] Barlow, R. and Wu, A., "Coherent systems with multi-state components," Mathematics of Operations Research, Vol. 3, N. 4 (Nov. 1978) 275-281.

[BU] Butler, D., "An importance ranking for system components based on cuts," Operations Research, Vol. 25, N. 5 (Sept.-Oct. 1977).

[EF] Edmonds, J. and Fulkerson, D., "Bottleneck extrema," Journal of Combinatorial Theory, Vol. 8 (1970) 299-306.

[EK] Edmonds, J. and Karp, R., "Theoretical improvement in algorithm efficiency for the network flow problem," Journal of the ACM, Vol. 19, No. 2 (1972), 248-264.

[FF] Ford, L. and Fulkerson, D., Flows in Networks, Princeton University Press, New Jersey, 1962.

[HU] Hunter, D., "Bounds on the probability of a union," Technical Report No. 73, Series 2, Dept. of Statistics, Princeton Univeristy, Dec. 1974.

[JD] Doulliez, P. and Jamoulle, E., "Transportation networks with random are capacities," Revue Francaise d'Automatique, Informatique et Recherche Operationannelle, Vol. 3, Nov. 1972, 45-59.

[KN] Knuth, D., The Art of Computer Programming, Vol. 3, Sorting and Searching, Addision-Wesley, Reading, Mass., 1970.

[RE] Reusch, B., "On the generation of prime implicants," Report TR 76-266, Dept. of Computer Science, Cornell Univ., Ithaca, New York (1976).

[RO] Rosenthal, A., "A computer scientist looks at reliability analysis," in Reliability and Fault Tree Analysis, R. Barlow, J. Fussell, and N. Singpurwalla (editors), SIAM, 1975.

[WI] Willie, R., "Computer-aided fault tree analysis," Report ORC78-14, Operations Research Center, Univ. of California, Berkeley (1978).

Structural Concepts for the Analysis of Stochastic Flow Networks

Jane Nichols Hagstrom

University of California, Berkeley

ABSTRACT

The model considered is that of a transshipment network with random capacities on the arcs, random supplies at source nodes, and random demands at sink nodes. The concepts of relevance, minimal cutsets, and modules, developed for the Boolean network representation of two-state component reliability problems, are extended to apply to the problem of determining the probability of meeting all demands.

1. Summary

This paper is motivated by the problem of determining the reliability of an electrical power system transmission network. The problem is simplified by treating the system as

a pipeline system for power. Then the system can be modelled as a transshipment network with random capacities on the arcs (transmission lines), random supply at the source nodes (generators), and random demand at the sink nodes (load distribution points). Concepts similar to those used in the analysis of Boolean network representations of on-off reliability problems arise.

We define a vulnerable region in such a way that its boundary arcs correspond to a minimal cutset. In terms of this, we can define relevance. We can define a measure on the space of vectors describing the state of each arc, source, and sink, which may be used to describe the state of a vulnerable region. This measure is increasing in each coordinate.

This leads to two possible definitions of a module. The definition of a structural module leads to a unique decomposition of the system into modular factors. The question of when the two definitions are equivalent is investigated.

2. Notation

Let (N,A(N)) be a connected undirected graph (power system network) where N is the set of nodes (busses) and A(N) is the set of arcs (transmission lines).

Let X be a subset of N. $\overline{X}$=N-X.

Let $A(X) \subseteq A(N)$ be the set of arcs whose endpoints are both in X. $(X,A(X))$ is a subgraph of $(N,A(N))$

Let $T(X) \subseteq A(N)$ be the set of arcs, one of whose endpoints is in X, the other in $\bar{X}$. Note $T(X)A(X)=\emptyset$.

Let G(X) be the set of sources (generators) at nodes in X. Let L(X) be the set of sinks (load distribution points) at nodes in X. $G(X)L(X)=\emptyset$, although a generator and a load can occur at the same node.

Let $C(X) = G(X) \cup L(X) \cup T(X)$.

For $i \in G(N) \cup A(N)$, let c_i be its capacity; $c_i \geq 0$. For $i \in L(N)$, let c_i be its demand; $c_i \leq 0$.

Let V be the set of state vectors c.

Definition: Let $\Gamma_c(X) = \sum_{i \in C(X)} c_i$ be the Gale measure of X evaluated at the state vector c.

Let $V_X = \{c \in V \mid \Gamma_c(X) < 0\}$.

For any set S, let $|S|$ be the number of elements in S.

3. Structural Theory

The following theory is based on the structure of $(N,A(N),G(N),L(N))$. No information about the space V of state vectors is used.

Gale's Supply Theorem. Given $c \in V$, the system's demands can be met if and only if $\Gamma_c(X) \geq 0$ for all $X \subseteq N$.

Proof: This is Theorem 5.3 in Gale's The Theory of Linear Economic Models [G,1950].

Lemma 1. Let $\{(X^j,A(X^j))\,|\,j=1,2,\ldots,n\}$ be the set of components (maximal connected subgraphs) of $(X,A(X))$. Then $\Gamma_c(X) = \sum_{j=1}^{n} \Gamma_c(X^j)$.

Proof: Certainly $\{G(X^j)\}$ and $\{L(X^j)\}$ are partitions of $G(X)$ and $L(X)$. Since the subgraphs $\{(X^j,A(X^j))\}$ are maximal, $\{T(X^j)\,|\,j=1,2,\ldots,n\}$ is a partition of $T(X)$.

Definition: Let $(X,A(X))$ be a connected subgraph such that $L(X) \neq \emptyset$ and for all components $(P,A(P))$ of $(\overline{X},A(\overline{X}))$, $G(P) \neq \emptyset$. Then X is a vulnerable region.

Definition: A member i of $C(N) \cup A(N)$ is relevant if $i \in C(X)$ for some vulnerable region X.

Theorem 1. Let $X \subseteq N$; $c \in V_X$. Then there exists a vulnerable region Y such that $c \in V_Y$.

Proof:

i) Let $\{(X^j,A(X^j))\,|\,j=1,2,\ldots,n\}$ be the set of components of $(X,A(X))$. By Lemma 1, $\sum_{j=1}^{n} \Gamma_c(X^j) = \Gamma_c(X) < \emptyset$. Then there exists j such that $\Gamma_c(X^j) < \emptyset$.

ii) Let $\{(P^k,A(P^k))\,|\,k=1,2,\ldots,m\}$ be the set of components of $(\overline{X}^j,A(\overline{X}^j))$. Let $S = \{k\,|\,G(P^k)=\emptyset\}$. Let $Y = X^j \cup \bigcup_{k \in S} P^k$.

$(Y,A(Y))$ is connected and the components of $(\overline{Y},A(\overline{Y}))$ are $\{(P^k,A(P^k))\mid G(P^k)\neq\emptyset\}$.

Note $G(Y) = G(X^j)$

$$L(Y) = L(X^j) \cup \bigcup_{k\in S} L(P^k)$$

$$T(X^j) = T(Y) \cup \bigcup_{k\in S} T(P^k)$$

where the right hand sides are partitions.

$$\Gamma_c(X^j) = \sum_{i\in G(X^j)} c_i + \sum_{i\in L(X^j)} c_i + \sum_{i\in T(X^j)} c_i < 0$$

$$\text{or} \quad \sum_{i\in G(Y)} c_i + \sum_{i\in L(X^j)} c_i + \sum_{i\in T(X^j)} c_i < 0.$$

Adding $\sum_{k\in S}\sum_{i\in L(P^k)} c_i$ to both sides,

$$\sum_{i\in G(Y)} c_i + \sum_{i\in L(Y)} c_i + \sum_{i\in T(X^j)} c_i < \sum_{k\in S}\sum_{i\in L(P^k)} c_i \leq 0.$$

Subtracting $\sum_{k\in S}\sum_{i\in T(P^k)} c_i$ from both ends,

$$\sum_{i\in G(Y)} c_i + \sum_{i\in L(Y)} c_i + \sum_{i\in T(Y)} c_i < -\sum_{k\in S}\sum_{i\in T(P^k)} c_i \leq 0,$$

$$\text{or } \Gamma_c(Y) < 0.$$

Hence,

$$c\in V_Y.$$

iii) Suppose $L(Y)=\emptyset$. Then

$$\Gamma_c(Y) = \sum_{i\in G(Y)} c_i + \sum_{i\in T(Y)} c_i \geq 0,$$

contradicting ii).

Thus we have exhibited a vulnerable region Y such that

$c \in V_Y$.

Lemma 2. Let X, Y $\subseteq$ N; X a vulnerable region; $Y \neq \emptyset$. Then $T(Y) \subseteq T(X)$ and $G(Y) \subseteq G(X)$ implies X = Y.

Proof: $T(Y) \subseteq T(X)$ implies $(Y-X, A(Y-X))$ is one or more components of $(\overline{X}, A(\overline{X}))$. $G(Y) \subseteq G(X)$ implies $G(Y-X) = \emptyset$. Since X is a vulnerable region, $Y-X = \emptyset$.

$T(Y) \subseteq T(X)$ implies either $X \subseteq Y$ or $XY = \emptyset$. Then $Y-X = \emptyset$ implies X = Y.

Lemma 3. Fix $\alpha \in N$. If all elements of $C(N) \cup A(N)$ are relevant, then every component of $(N-\{\alpha\}, A(N-\{\alpha\}))$ contains a generator or a load.

Proof: Let $(Y, A(Y))$ be such a component. Suppose $L(Y) \cup G(Y) = \emptyset$. Let X be a vulnerable region.

Suppose $\alpha \in X$. Then $Y \subseteq X$, for otherwise Y-X is a component of $\overline{X}$ without any generators. Suppose $\alpha \in \overline{X}$. Then $Y \subseteq \overline{X}$, for otherwise YX is a component of X without any loads.

Note that the arcs in C(Y) all have one endpoint at α. Then for any X, $C(Y) \subseteq A(X)$ or $C(Y) \subseteq A(\overline{X})$. Then the arcs in C(Y) are irrelevant, contrary to hypothesis.

Lemma 4. Arc $i \in A(N)$ is relevant if and only if there exists a simple path from a load to a generator which includes i.

Proof: Suppose there exists a simple path from a load to a generator which includes i. Let P be the set of nodes in the path from the load to and including the nearest endpoint of i. Let $\{(Q_k, A(Q_k)) \mid k=1,2,\ldots,n\}$ be the components of $(\bar{P}, A(\bar{P}))$.

Since the path is simple, the rest of its nodes are all contained in one component, say the kth, of $(\bar{P}, A(\bar{P}))$. Then $P \cup \bigcup_{j \neq k} Q_j$ is a vulnerable region, since it is connected, it contains a load, and its complement is connected and contains a generator. Furthermore, $i \notin T(P \cup \bigcup_{j \neq k} Q_j)$, so i is relevant.

Suppose i is relevant. Then there exists a vulnerable region X such that $i \notin C(X)$. Let $P \subseteq X$ be a simple path in $(X, A(X))$ from a load to the endpoint of i in X. Let $Q \subseteq \bar{X}$ be a simple path in $(\bar{X}, A(\bar{X}))$ from a generator to the endpoint of i in $\bar{X}$. P and Q are arc-disjoint as well as node-disjoint. Then PUQ is a simple path from a load to a generator which includes i.

Remark: If there is a load in the network, all loads and generators are relevant, since they all occur in C(N).

Lemma 5. Suppose all elements of $C(N) \cup A(N)$ are relevant. Let $P \subseteq N$ be a simple path. Then there exists a vulnerable region X such that exactly one arc of P occurs in C(X).

Proof: By induction on k, the number of nodes in P.

True for k = 2, since all arcs are relevant. Suppose it is true for all paths with fewer than k nodes.

Let α be one endpoint of P, β the other. Then by the induction hypothesis, there exists a vulnerable region Y such that exactly one arc of the path defined by $P-\{\alpha\}$ occurs in C(Y). If $\alpha \in Y$, $\beta \in \bar{Y}$, or vice-versa, we are done.

Assume $\alpha, \beta \in \bar{Y}$. The argument when $\alpha, \beta \in Y$ is similar.

Let $(Z,A(Z))$ be the component of $(\bar{Y},A(\bar{Y}))$ that contains β. Let $Q = (P-\{\alpha\})\bar{Y}$. Then Q is a simple path which is contained in Z.

If α is not in Z, let $X = Y \cup Z$, and X is the desired vulnerable region. Otherwise, we now have $\alpha, \beta \in Z$. We are left with the following five cases.

i) $(Z-\{\alpha\},A(Z-\{\alpha\}))$ is connected and $G(Z-\{\alpha\}) \neq \emptyset$. Then let $X = Y \cup \{\alpha\}$.

ii) $(Z-\{\alpha\},A(Z-\{\alpha\}))$ is not connected. Let $(W,A(W))$ be the component of $(Z-\{\alpha\},A(Z-\{\alpha\}))$ that contains Q.

ii.a) $G(W) \neq \emptyset$. Then let $X = Y \cup [Z-W]$.

ii.b) $G(W) = \emptyset$. Then let $X = Y \cup W$.

iii) $(Z-\{\alpha\},A(Z-\{\alpha\}))$ is connected and $G(Z) = G(\{\alpha\})$.

iii.a) (Z-Q,A(Z-Q)) is connected. Then let X = YUQ.

iii.b) (Z-Q,A(Z-Q)) is not connected. Let (W,A(W)) be the component of (Z-Q,A(Z-Q)) which contains α. Then let X = YU[Z-W].

Definition: Two elements i, j of C(N)UA(N) are in parallel if they satisfy one of the following:

1) i, j$\in$G({α})UL({α}) for some node α.

2) i, j$\in$A({α, β}) for some node pair {α, β}.

Definition: Two arcs i, j appear in series if there exists a node α such that C({α}) = {i, j}. A set of arcs which form a simple path are in series if every pair of adjacent arcs appear in series.

Lemma 6. Suppose all elements of C(N)UA(N) are relevant and i, j are elements such that for any vulnerable region X, i$\in$C(X) if and only if j$\in$C(X). Then i and j are in parallel.

Proof:

i) Suppose i$\in$C({α}), j$\in$C({β}), and $\alpha \neq \beta$. Then there exists a simple path from α to β and by Lemma 5, there exists a vulnerable region X such that only one of α and β is in X. Then only one of i and j is in C(X), contrary to hypothesis. Hence $\alpha = \beta$.

ii) Suppose i$\in$A(N). Then since i is not in C(N), j is not in C(N) and must be in A(N).

If i and j have exactly one endpoint in common, they form a simple path, and by Lemma 5, there exists a vulnerable region X such that C(X) intersects exactly one of them, contrary to hypothesis.

Assume then that i and j have no endpoints in common. Let X be a vulnerable region such that $i, j \in T(X)$. Let α, β be the endpoints of i and j which are in $\overline{X}$. Let P be a path which includes i, j and a simple path in X from α to β. By our assumptions, P is simple.

Then the same construction as in the proof of Lemma 5 can be performed and we will have a vulnerable region Y such that only one of i and j is in C(Y). Since this is contrary to hypothesis, we are left with i and j having both endpoints in common.

4. Modules

The following section makes two possible definitions for a module and begins to investigate the relation between them.

Definition: Let $M \subseteq C(N) \cup A(N)$. M is a structural module if the following holds:

Let X, Y be any vulnerable regions such that $C(X)M \neq \emptyset$, $C(Y)M \neq \emptyset$. Then there exists a vulnerable region Z such that

$$C(Z) = [C(X)M] \cup [C(Y)-M].$$

Definition: Let $M \subseteq C(N) \cup A(N)$. M is a functional module if there exists a real valued function $\not\prec$ of c^M (c restricted to M) such that the following holds:

Let X be a vulnerable region such that $C(X)M \neq \emptyset$.

$$\sum_{i \in C(X)-M} c_i + \not\prec(c^M) < 0$$

if and only if there exists a vulnerable region Z such that

$$C(Z)-M = C(X)-M \text{ and } \Gamma_c(Z) < 0.$$

Definition: A module M is proper if M is a proper subset of $C(N) \cup A(N)$.

Theorem 2. Every structural module is a functional module.

Proof: Let M be a structural module. Let

$$\not\prec(c^M) = \text{Min}\{ \sum_{i \in C(X)M} c_i \mid X \text{ is a vulnerable region and } C(X)M \neq \emptyset \}$$

i) Let Y be a vulnerable region such that $C(Y)M \neq \emptyset$ and let $c \in V_Y$. By definition, $\not\prec(c^M) \leq \sum_{i \in C(Y)M} c_i$. Then

$$\sum_{i \in C(Y)-M} c_i + \not\prec(c^M) \leq \sum_{i \in C(Y)-M} c_i + \sum_{i \in C(Y)M} c_i < 0.$$

ii) Let Y be a vulnerable region such that $C(Y)M \neq \emptyset$. Suppose $\sum_{i \in C(Y)-M} c_i + \not\prec(c^M) < 0$ for some c.

Let X be a vulnerable region such that $C(X)M \neq \emptyset$ and $\not\prec(c^M) = \sum_{i \in C(X)M} c_i$. Let Z be the vulnerable region such that

$C(Z) = [C(X)M]U[C(Y)-M]$.

$$\Gamma_c(Z) = \sum_{i \in C(Y)-M} c_i + \sum_{i \in C(X)M} c_i = \sum_{i \in C(Y)-M} c_i + \phi(c^M) < \emptyset.$$

Hence M is a functional module.

Lemma 7. Suppose all elements are relevant. Let M be a proper structural module. Then there is no vulnerable region X such that $C(X) \subseteq M$.

Proof: Suppose there exists a vulnerable region X such that $C(X) \subseteq M$. Let Y be another vulnerable region such that $C(Y)M \neq \emptyset$. Then there exists a vulnerable region Z such that

$$C(Z) = [C(Y)M] \cup [C(X)-M] = C(Y)M$$

Then $C(Z) \subseteq C(Y)$ implies $C(Z) = C(Y)$ and $C(Y) \subseteq M$.

In particular, $L(X) \subseteq M$, so $L(N)M \neq \emptyset$, and $C(N) \subseteq M$. But then $L(Y) \subseteq M$ for all vulnerable regions Y, and thus $C(Y) \subseteq M$ for all vulnerable regions Y. This is impossible since M is proper and the elements outside of M are relevant.

Lemma 8. Let all elements of $C(N) \cup A(N)$ be relevant. Suppose $M_1 \cup M_2$, $M_2 \cup M_3$ are structural modules such that M_1, M_2, M_3 are nonempty and mutually disjoint. then one of the following properties holds.

1) For all vulnerable regions X such that $C(X)M_i \neq \emptyset$ for some i, $C(X)M_j \neq \emptyset$ for all j.

2) For all vulnerable regions X such that $C(X)M_i \neq \emptyset$ for some i, $C(X)M_j = \emptyset$ for all $j \neq i$.

Proof: Since the M_i's are nonempty and all elements are relevant, there exist vulnerable regions X_1, X_2, X_3 (not necessarily distinct) such that

$$C(X_i)M_i \neq \text{phi}.$$

We will show that for any choice of X_1, X_2, X_3, $C(X_i)M_j \neq \emptyset$ for some $i \neq j$ implies $C(X_i)M_j \neq \emptyset$ for all i, j.

Let $P = [A(N) \cup C(N)] - [M_1 \cup M_2 \cup M_3]$.

i) Suppose $C(X_1)M_2 \neq \emptyset$.

$M_2 \cup M_3$ a structural module and $C(X_1)M_2 \neq \emptyset$ implies there exists a vulnerable region Y such that

$$C(Y) = [C(X_3)M_1] \cup [C(X_1)M_2] \cup [C(X_1)M_3] \cup [C(X_3)P].$$

$M_1 \cup M_2$ a structural module and $C(Y)M_2 \neq \emptyset$ implies there exists a vulnerable region such that

$$C(Z) = [C(Y)M_1] \cup [C(Y)M_2] \cup [C(X_1)M_3] \cup [C(X_1)P]$$
$$= [C(X_3)M_1] \cup [C(X_1)M_2] \cup [C(X_1)M_3] \cup [C(X_1)P]$$

Since $C(X_1)M_1 \neq \emptyset$,

$$C(X_3)M_1 \neq \emptyset,$$

for otherwise $C(Z) \subset C(X_1)$.

ii) Suppose $C(X_3)M_1 \neq \emptyset$.

$M_2 \cup M_3$ a structural module implies there exists a

vulnerable region Y such that

$$C(Y) = [C(X_3)M_1] \cup [C(X_2)M_2] \cup [C(X_2)M_3] \cup [C(X_3)P].$$

$M_1 \cup M_2$ a structural module and $C(X_3)M_1 \neq \emptyset$ implies there exists a vulnerable region Z such that

$$C(Z) = [C(X_3)M_1] \cup [C(X_3)M_2] \cup [C(Y)M_3] \cup [C(Y)P]$$
$$= [C(X_3)M_1] \cup [C(X_3)M_2] \cup [C(X_2)M_3] \cup [C(X_3)P].$$

Then since $C(X_3)M_3 \neq \emptyset$, we may conclude

$$C(X_2)M_3 \neq \emptyset$$

iii) Suppose $C(X_2)M_3 \neq \emptyset$.

$M_1 \cup M_2$ a structural module implies there exists a vulnerable region Y such that

$$C(Y) = [C(X_2)M_1] \cup [C(X_2)M_2] \cup [C(X_1)M_3] \cup [C(X_1)P]$$

$M_2 \cup M_3$ a structural module and $C(X_2)M_2 \neq \emptyset$ implies there exists a vulnerable region Z such that

$$C(Z) = [C(X_2)M_1] \cup [C(Y)M_2] \cup [C(Y)M_3] \cup [C(X_2)P]$$
$$= [C(X_2)M_1] \cup [C(X_2)M_2] \cup [C(X_1)M_3] \cup [C(X_2)P].$$

Since $C(X_2)M_3 \neq \emptyset$,

$$C(X_1)M_3 \neq \emptyset$$

iv) Suppose $C(X_1)M_3 \neq \emptyset$.

$M_1 \cup M_2$ a structural module implies there exists a vulnerable region Y such that

$$C(Y) = [C(X_2)M_1] \cup [C(X_2)M_2] \cup [C(X_1)M_3] \cup [C(X_1)P].$$

$M_2 \cup M_3$ a structural module and $C(X_1)M_3 \neq \emptyset$ implies there exists a vulnerable region Z such that

$$C(Z) = [C(X_1)M_1] \cup [C(Y)M_2] \cup [C(Y)M_3] \cup [C(X_1)P]$$
$$= [C(X_1)M_1] \cup [C(X_2)M_2] \cup [C(X_1)M_3] \cup [C(X_1)P].$$

Since $C(X_2)M_2 \neq \emptyset$,

$$C(X_1)M_2 \neq \emptyset,$$

for otherwise $C(X_1) \subset C(Z)$.

We now have the following chain:

$$C(X_1)M_2 \neq \emptyset \rightarrow C(X_3)M_1 \neq \emptyset \rightarrow C(X_2)M_3 \neq \emptyset$$
$$\rightarrow C(X_1)M_3 \neq \emptyset \rightarrow C(X_1)M_2 \neq \emptyset,$$

so we can replace implications by equivalences:

$$C(X_1)M_2 \neq \emptyset \leftrightarrow C(X_3)M_1 \neq \emptyset$$
$$\leftrightarrow C(X_1)M_3 \neq \emptyset \leftrightarrow C(X_2)M_3 \neq \emptyset.$$

Noting that the roles of 1 and 3 are completely symmetric, we write

$$C(X_3)M_2 \neq \emptyset \leftrightarrow C(X_1)M_3 \neq \emptyset$$
$$\leftrightarrow C(X_3)M_1 \neq \emptyset \leftrightarrow C(X_2)M_1 \neq \emptyset.$$

Combining these two, since they have a clause in common, we get the desired result.

Theorem 3: The Three Modules Theorem. Let every element of $C(N) \cup A(N)$ be relevant. Suppose that $M_1 \cup M_2$, $M_2 \cup M_3$ are structural modules such that M_1, M_2, M_3 are nonempty and mutually disjoint. Then M_1, M_2, M_3, $M_1 \cup M_3$, and $M_1 \cup M_2 \cup M_3$ are

structural modules.

Proof: Let X, Y be vulnerable regions such that

$$C(X)[M_1 \cup M_2 \cup M_3] \neq \emptyset$$

$$C(Y)[M_1 \cup M_2 \cup M_3] \neq \emptyset.$$

Let P be defined as in Lemma 8.

We wish to prove the following. Given the conditions specified below, there are vulnerable regions Z whose C(Z)'s are defined as given below.

$$C(Z) = [C(X)M_1] \cup [C(X)M_2] \cup [C(X)M_3] \cup [C(Y)P] \quad (a)$$

$C(X)[M_1 \cup M_3] \neq \emptyset$ and
$C(Y)[M_1 \cup M_3] \neq \emptyset$ implies

$$C(Z) = [C(X)M_1] \cup [C(Y)M_2] \cup [C(X)M_3] \cup [C(Y)P] \quad (b)$$

$C(X)M_1 \neq \emptyset$ and
$C(Y)M_1 \neq \emptyset$ implies

$$C(Z) = [C(X)M_1] \cup [C(Y)M_2] \cup [C(Y)M_3] \cup [C(Y)P] \quad (c)$$

Similar equations to (c) hold for M_2, M_3.

i) Suppose condition (1) of Lemma 8 holds. $M_1 \cup M_2$ a structural module implies there exists a vulnerable region W such that

$$C(W) = [C(X)M_1] \cup [C(X)M_2] \cup [C(Y)M_3] \cup [C(Y)P]$$

$M_2 \cup M_3$ a structural module implies there exists a vulnerable region Z such that

$$C(Z) = [C(W)M_1] \cup [C(X)M_2] \cup [C(X)M_3] \cup [C(W)P]$$
$$= [C(X)M_1] \cup [C(X)M_2] \cup [C(X)M_3] \cup [C(Y)P] \quad (a)$$

$M_1 \cup M_2$ a structural module implies there exists a vulnerable region V such that

$$C(V) = [C(Y)M_1] \cup [C(Y)M_2] \cup [C(X)M_3] \cup [C(X)P]$$

$M_2 \cup M_3$ a structural module implies there exists a vulnerable region Z such that

$$C(Z) = [C(W)M_1] \cup [C(V)M_2] \cup [C(V)M_3] \cup [C(W)P]$$
$$= [C(X)M_1] \cup [C(Y)M_2] \cup [C(X)M_3] \cup [C(Y)P] \quad (b)$$

$M_2 \cup M_3$ a structural module implies there exists a vulnerable region Z such that

$$C(Z) = [C(W)M_1] \cup [C(Y)M_2] \cup [C(Y)M_3] \cup [C(W)P]$$
$$= [C(X)M_1] \cup [C(Y)M_2] \cup [C(Y)M_3] \cup [C(Y)P] \quad (c)$$

The cases of M_2 and M_3 are proved similarly.

ii) Cases ii) and iii) are really subcases of condition (2) of Lemma 8. In case ii) we suppose

$$C(X)M_1 \neq \emptyset,\ C(X)M_2 = \emptyset,\ C(X)M_3 = \emptyset$$
$$C(Y)M_1 \neq \emptyset,\ C(Y)M_2 = \emptyset,\ C(Y)M_3 = \emptyset$$

$M_1 \cup M_2$ a structural module implies there exists a vulnerable region Z such that

$$C(Z) = [C(X)M_1] \cup [C(Y)P] \qquad (a),(b),(c)$$

The proof of (c) is complete.

iii) Suppose

$$C(X)M_1 \neq \emptyset,\ C(X)M_2 = \emptyset,\ C(X)M_3 = \emptyset$$
$$C(Y)M_1 = \emptyset,\ C(Y)M_2 = \emptyset,\ C(Y)M_3 \neq \emptyset$$

Let R be a vulnerable region such that

$$C(R)M_1 = \emptyset,\ C(R)M_2 \neq \emptyset,\ C(R)M_3 = \emptyset$$

$M_2 \cup M_3$ a structural module implies there exists a vulnerable region V such that

$$C(V) = [C(R)M_2] \cup [C(Y)P]$$

$M_1 \cup M_2$ a structural module implies there exists a vulnerable region Z such that

$$C(Z) = [C(X)M_1] \cup [C(V)P]$$
$$= [C(X)M_1] \cup [C(Y)P] \qquad (a),(b)$$

The proof of (b) is complete. The proof of (a) involves repeating iii) for other permutations of subscripts.

Modular Factorization. The discussion of Birnbaum and Esary in section 5.2 of their paper on modules [BE,1965] applies, although for the moment we should replace the concepts of "modules in parallel" by "modules satisfying (1) below" and "modules in series" by "modules satisfying (2) below."

(1) For all vulnerable regions X such that $C(X)M_i \neq \emptyset$ for some i, $C(X)M_j \neq \emptyset$ for all j.

(2) For all vulnerable regions X such that $C(X)M_i \neq \emptyset$ for some i, $C(X)M_j = \emptyset$ for all $j \neq i$.

Then the statement of modular factorization is that either the system may be factored into a set of disjoint maximal proper structural modules or it may be factored into a set of disjoint structural modules which satisfy (1) or (2).

Theorem 4. Suppose all elements of C(N)UA(N) are relevant. Let M be a proper structural module. Then one of the following two conditions holds.

(1) For all vulnerable regions X such that $C(X)M \neq \emptyset$, $L(X)M = L(N)M$.

(2) For all vulnerable regions X such that $C(X)M \neq \emptyset$, $C(X)-M = L(N)-M$.

Proof: We must show that for a specific vulnerable region $X \neq N$ such that $C(X)M \neq \emptyset$, either $L(X)M = L(N)M$ or $C(X)-M = L(N)-M$, and if Y is another vulnerable region such that $C(Y)M \neq \emptyset$, the same condition holds for it.

Note that if $L(N)M = \emptyset$, (1) holds trivially, so assume $L(N)M \neq \emptyset$.

Since X, N are vulnerable regions, there exist vulnerable regions Z, W such that

$$C(Z) = [C(X)M] \cup [C(N)-M]$$

$$C(W) = [C(N)M] \cup [C(X)-M].$$

Note that $C(N) \subseteq C(Z) \cup C(W)$.

We will examine $T(X)$.

i) If $T(X)M = \emptyset$, $Z = N$, and

$$C(X)M = C(N)M.$$

ii) If $T(X)-M = \emptyset$, $W = N$, and

$$C(X)-M = C(N)-M.$$

In this case, $G(X) = \emptyset$, for otherwise $\overline{X}$ is a vulnerable region and $C(\overline{X}) \subseteq M$. Then

$$C(X)-M = L(N)-M.$$

iii) Otherwise both $T(X)M$ and $T(X)-M$ are cuts in $(N,A(N))$.

Let $(P,A(P))$ be the subgraph of $(\overline{X},A(\overline{X}))$ such that $T(P) = T(X)M$. Let $(Q,A(Q))$ be the subgraph of $(\overline{X},A(\overline{X}))$ such that $T(Q) = T(X)-M$. These are each made up of components of $(\overline{X},A(\overline{X}))$, and $PQ = \emptyset$.

Since X is a vulnerable region, $G(P) \neq \emptyset$, $G(Q) \neq \emptyset$. Then

$$Z = X \cup Q \quad \text{or} \quad Z = P \qquad \text{and}$$

$$W = X \cup P \quad \text{or} \quad W = Q$$

If $Z = P$ and $W = Q$, $L(X)M = \emptyset$ and $L(X)-M = \emptyset$, which is impossible, since X is a vulnerable region.

If $Z = P$ and $W = XUP$, then $C(N) \subseteq C(Z) \cup C(W)$ contradicts $G(Q) \neq \emptyset$. Similarly, if $Z = XUQ$ and $W = Q$.

Hence $Z = XUQ$ and $W = XUP$.

Suppose $L(P) \neq \emptyset$. Then $(Z, A(Z))$ connected and $G(Q) \neq \emptyset$ implies P is a vulnerable region.

$$C(P) = [L(W)-L(X)] \cup [G(W)-G(X)] \cup [\Gamma(X)M]$$
$$= [(L(N)M)-L(X)] \cup [(G(N)M)-G(X)] \cup [Y(X)M].$$

then $C(P) \subseteq M$, contrary to Lemma 7.

Hence $L(P) = (L(N)M)-L(X) = \emptyset$, or

$$L(X)M = L(N)M.$$

Suppose there exist vulnerable regions X, Y such that

$$C(X)-M \neq L(N)-M \qquad (a)$$
$$L(Y)M \neq L(N)M \qquad (b)$$
$$C(X)M \neq \emptyset$$
$$C(Y)M \neq \emptyset$$

By the above argument, we know

$$L(X)M = L(N)M$$
$$C(Y)-M = L(N)-M, \; G(Y)M = \emptyset$$

Let Z, W be vulnerable regions such that

$$C(Z) = [C(X)M] \cup [C(Y)-M]$$
$$= [C(X)M] \cup [L(N)-M]$$
$$C(W) = [C(Y)M] \cup [C(X)-M]$$

Suppose $T(X)M = \emptyset$. Then

$$T(W) = T(X) \cup T(Y),\ T(X)T(Y) = \emptyset$$

$L(W) \neq \emptyset$ implies WX or $WY \neq \emptyset$.

$T(X)$, $T(Y)$ disjoint and nonempty (since $X \neq N$, $Y \neq N$) implies

$XY = W$

$G(X-W) \neq \emptyset$, $G(Y-W) \neq \emptyset$

$L(X-W) = [L(X)M]-L(Y) = [L(N)M]-L(Y)$

$L(X-W) \neq \emptyset$ by assumption (b). Then $X-W$ is a vulnerable region such that

$$C(X-W) = [\Gamma(X)M] \cup [(L(X)M)-L(Y)] \cup [(G(X)M)-G(Y)] \subseteq M$$

This is impossible, so $T(X)M \neq \emptyset$.

$T(X)-M = \emptyset$ implies $T(Z) = T(X)$.

$L(Z)L(X) \neq \emptyset$ implies

$ZX \neq \emptyset$ implies

$Z = X$ implies

$C(X)-M = C(N)-M = L(N)-M$

contrary to assumption (a).

Hence $T(X)-M \neq \emptyset$.

Then since $L(X)L(Z) \neq \emptyset$, and $T(X)-M \neq \emptyset$, $T(X)M \neq \emptyset$,

$$X \subset Z$$

Let $(P,A(P))$ be a component of $(Z-X,A(Z-X))$. $(P,A(P))$

is also a component of $(\overline{X},A(\overline{X}))$. $G(P) = G(Z)-G(X) = \emptyset$.

But then X is not a vulnerable region. We conclude that it is impossible that both (a) and (b) hold for the same module in the same system.

Remark: Figure 1 exhibits a case where (2) holds and (1) does not.

5. The Connectivity Problem

Assume we have a network $(N,A(N),G(N),L(N))$ in which all elements of $A(N) \cup G(N) \cup L(N)$ are relevant. Let the state space V consist of all binary vectors, i.e., $c_i = 0$ or 1 if $i \in G(N) \cup A(N)$, $c_i = 0$ or -1 if $i \in L(N)$. We will refer to this problem as the connectivity problem.

We may visualize the problem as follows. If an element of L(N) is in state -1, service is demanded at that node. If an element of G(N) is in state 1, service is available at that node. If an element of A(N) is in state 1, it is available as a communication path. When an element of L(N) is being served, it ties up both its server and the communication path between them. Then given a state vector c, will all those who demand service get it?

This section will show that the structural theory will tell us everything we can find out about this system unless we know something about the actual state.

Theorem 5. Suppose $(N,A(N),G(N),L(N)),V$ describes the

connectivity problem. Let X be a vulnerable region. Then there exists $c \in V_X$ such that c is not in V_Y for all $Y \neq X$, $Y \subseteq N$.

Proof: Choose $k \in L(X)$. Let $c_k = -1$. Let $c_j = 1$ for all $j \in [(G(N) \cup A(N)) - C(X)]$. Let $c_j = \emptyset$ for all other j. Then $\Gamma_c(X) = c_k < \emptyset$.

Let $Y \subseteq N$, $Y \neq X$. By Lemma 2, $Y \neq X$ implies $T(Y)-T(X) \neq \emptyset$ or $G(Y)-G(X) \neq \emptyset$. Then

$$\begin{aligned}\Gamma_c(Y) &= \sum_{i \in C(Y)-C(X)} c_i + \sum_{i \in C(Y)C(X)} c_i \\ &= \sum_{i \in G(Y)-G(X)} c_i + \sum_{i \in T(Y)-T(X)} c_i + \sum_{i \in L(Y)L(X)} c_i \\ &= |G(Y)-G(X)| + |T(Y)-T(X)| + c_k \geq \emptyset.\end{aligned}$$

Remark. Theorem 5 does not necessarily hold for other state spaces V. An example where it does not hold is given in Figure 2.

Lemma 9. Suppose $(N, A(N), G(N), L(N)), V$ describes the connectivity problem. Let M be a proper functional module. Let X be a vulnerable region. Then $C(X)-M \neq \emptyset$.

Proof: Suppose $C(X) \subseteq M$. Let Y be a vulnerable region such that $C(Y)M \neq \emptyset$.

Choose $k \in L(X)$. Set $c_k = -1$. For all $i \in [(A(N) \cup G(N))M] - C(X)$, set $c_i = 1$. For all other i, set $c_i = \emptyset$.

$\Gamma_c(X) = c_k < \emptyset$ implies $\not\in(c^M) < \emptyset$, which in turn implies there exists a vulnerable region Z such that

$$C(Z)-M = C(Y)-M$$

$$\Gamma_c(Z) < \emptyset.$$

$$T(Z)M \subseteq \Gamma(X)M = T(X),$$

$$G(Z)M \subseteq G(X)M = G(X)$$

since otherwise

$$\begin{aligned}\Gamma_c(Z) &= \sum_{i\in C(Z)M} c_i \\ &= \sum_{i\in L(Z)M} c_i + \sum_{i\in[G(Z)M]-G(X)} c_i + \sum_{i\in[\Gamma(Z)M]-\Gamma(X)} c_i \\ &= c_k + |[G(Z)M]-G(X)| + |[T(Z)M]-T(X)| > \emptyset.\end{aligned}$$

Hence we have constructed a vulnerable region Z such that

$$\begin{aligned}&C(Z)-M = C(Y)-M \\ &G(Z)M \subseteq G(X)M = G(X) \\ &T(Z)M \subseteq T(X)M = T(X).\end{aligned} \tag{1}$$

Note $L(Z)M \neq \emptyset$ for otherwise $\Gamma_c(Z) = \emptyset$.

Now construct a new state vector c as follows: Choose $k\in L(Z)M$. Set $c_k = -1$. For all $i\in[A(N)UG(N))M]-C(Z)$, set $c_i = 1$. For all other i, set $c_i = 0$.

$\Gamma_c(Z) = c_k < \emptyset$ implies $\not\in(c^M) < \emptyset$, which in turn implies that there exists a vulnerable region W such that

$$C(W)-M = C(X)-M = \emptyset$$

$$\Gamma_c(W) < \emptyset.$$

$T(W)M \subseteq T(Z)M$ and $G(W)M \subseteq G(Z)M$

by the same argument as before.

Thus we have constructed a vulnerable region W such that

$$\begin{aligned} &C(W)-M = C(X)-M = \phi \\ &G(W)M \subseteq G(Z)M \\ &T(W)M \subseteq T(Z)M \end{aligned} \qquad (2)$$

Combining the sets of relations (1) and (2), we have

$$\begin{aligned} &C(W)-M = \phi \\ &G(W)M \subseteq G(X) \\ &T(W)M \subseteq T(X) \end{aligned}$$

or

$$\begin{aligned} &G(W) \subseteq G(X) \\ &T(W) \subseteq T(X) \end{aligned}$$

By Lemma 2, W = X. Then $T(W)M = T(X)$, and

$$T(W)M \subseteq T(Z)M \subseteq T(X)$$

implies $T(X) = T(Z)M$. Similarly, $G(X) = G(Z)M$.

Then $T(X) \subseteq T(Z)$ and $G(X) \subseteq G(Z)$ implies X = Z. Then

$$C(Y)-M = C(Z)-M = C(X)-M = \phi.$$

Hence $C(Y) \subseteq M$ for all vulnerable regions Y.

Since all members of G(N)UA(N) are relevant,

$$C(N)UA(N) \subseteq M.$$

Since our hypothesis is that M is proper, we have a contradiction and

$$C(X)-M \neq \emptyset.$$

Lemma 10. Let $(N,A(N),G(N),L(N)),V$ be the connectivity problem. Let M be a proper functional module. Tnen one of the following holds:

(1) For all vulnerable regions X such that $C(X)M \neq \emptyset$, $L(X)M = L(N)M$.

(2) For all vulnerable regions X such that $C(X)M \neq \emptyset$, $C(X)-M = L(N)-M$.

Proof: If $L(N)M = \emptyset$, condition (1) is automatically satisfied, so suppose $L(N)M \neq \emptyset$.

i) Suppose there exists X a vulnerable region such that $C(X)M \neq \emptyset$ and $L(X)M \neq L(N)M$. Then there exists $k \in L(\overline{X})M$.

Set $c_k = -1$. For all $i \in A(N)M$, set $c_i = 1$. For all other i, set $c_i = \emptyset$.

$\Gamma_c(N) = c_k < \emptyset$ implies $\not\in(c^M) < \emptyset$, which in turn implies there exists a vulnerable region Z such that

$$C(Z)-M = C(X)-M$$

$$\Gamma_c(Z) < \emptyset$$

$k \in L(Z)$, for otherwise $\Gamma_c(Z) = \sum_{i \in \Gamma(Z)M} c_i \geq \emptyset$. $T(Z) = T(X)-M$ for otherwise $\Gamma_c(Z) = |T(Z)M| + -1 \geq \emptyset$.

$T(Z) = T(X)-M$ implies $Z-X = \bigcup_{P \in S} P$, where

$S=\{P \subseteq N \mid (P,A(P))$ is a component of $(\overline{X},A(\overline{X}))$ and $T(P) \subseteq T(X)M\}$.

Let $P^o \in S$ be such that $k \in L(P^o)$.

$T(P^o) \subseteq T(X)M$

$G(P^o) \subseteq G(Z)M$

$L(P^o) \subseteq L(Z)M$

Hence, $C(P^o) \subseteq M$. Also P^o is connected and $L(P^o) \neq \emptyset$. Furthermore, $\overline{P}^o$ is connected; $X \subseteq \overline{P}^o$.

a) Suppose $T(X)-M \neq \emptyset$. Then $T(Z)-M \neq \emptyset$ and $G(\overline{Z}) \neq \emptyset$.

Then $P^o \subseteq Z$ implies

$\overline{Z} \subseteq \overline{P}^o$ implies

$G(\overline{Z}) \subseteq G(\overline{P}^o)$ implies

$G(\overline{P}^o) \neq \emptyset$ implies

P^o is a vulnerable region, but $C(P^o) \subseteq M$ contradicts the condition that M is proper.

b) Suppose $T(X) \subseteq M$. Then $P^o = \overline{X}$.

If $G(X) \neq \emptyset$, then P^o is a vulnerable region and $C(P^o) \subseteq M$ contradicts the condition that M is proper.

Otherwise $G(X) = \emptyset$ and $C(X)-M \neq \emptyset$, so

$$C(X)-M = L(N)-M$$

Thus we have shown that any given X must satisfy (1) or

(2). Now it remains to be shown that we cannot have two vulnerable regions in the same network, one of which satisfies (2), the other of which does not.

ii) Suppose $X, Y \neq \emptyset$, $C(X)M \neq \emptyset$, $C(Y)M \neq \emptyset$, $C(X)-M \neq L(N)-M$, and $C(Y)-M = L(N)-M$. Then $L(X)M = L(N)M$.

Choose $k \in L(N)M$. Set $c_k = -1$. For all $i \in [(A(N) \cup G(N))M]-C(X)$, set $c_i = 1$. For all other i, set $c_i = \emptyset$.

$\Gamma_c(X) = c_k < \emptyset$ implies $\not\in(c^M) < \emptyset$, which in turn implies there exists a vulnerable region Z such that

$$C(Z)-M = C(Y)-M$$

$$\Gamma_c(Z) < \emptyset.$$

$G(Z)M \subseteq G(X)M$ and $T(Z)M \subseteq T(X)M$ for otherwise

$$\Gamma_c(Z) = c_k + |[G(Z)M]-G(X)| + |[T(Z)M]-T(X)| \geq \emptyset.$$

Then $C(Z)-M = C(Y)-M = L(N)-M$ implies

$$G(Z) \subseteq G(X)M$$

$$T(Z) \subseteq T(X)M$$

Lemma 2 implies $Z = X$ and thus

$$C(X)-M = C(Z)-M = C(Y)-M = L(N)-M.$$

We have a contradiction, so if (2) holds for any vulnerable region other than N, it holds for all.

Theorem 6. Let $(N, A(N), G(N), L(N)), V$ represent the con-

nectivity problem. Then every proper functional module is a structural module.

Proof: Let M be a proper functional module. Let X, Y be vulnerable regions such that $C(X)M \neq \emptyset$, $C(Y)M \neq \emptyset$. We must show there exists a vulnerable region Z such that

$$C(Z) = [C(X)M] \cup [C(Y)-M].$$

Assume $C(X)M \neq C(Y)M$, $C(X)-M \neq C(Y)-M$, for otherwise the proof is trivial.

i) Suppose $L(N)M = \emptyset$.

If $L(X)L(Y) \neq \emptyset$, choose $k \in L(X)L(Y)$ and set $c_k = -1$. Otherwise, choose $j \in L(X)$, $k \in L(Y)$, and set $c_j = c_k = -1$. For all $i \in [M(A(N) \cup G(N))]-C(X)$, set $c_i = 1$. For all other i, set $c_i = \emptyset$.

$\Gamma_c(X) = c_k < \emptyset$ implies

$c_k + \not\in(c^M) < \emptyset$ implies

$\sum_{i \in C(Y)-M} c_i + \not\in(c^M) < \emptyset$ implies

there exists a vulnerable region Z such that

$$C(Z)-M = C(Y)-M$$

$$\Gamma_c(Z) < \emptyset.$$

$G(Z)M \subseteq G(X)M$

$T(Z)M \subseteq T(X)M$ (1)

since otherwise

$$\Gamma_c(Z) = c_k + |[C(Y)M]-C(X)| \geq \emptyset.$$

Replacing Y by X and X by Z in the argument above, we can construct W such that

$$\begin{aligned} C(W)-M &= C(X)-M \\ G(W)M &\subseteq G(Z)M \\ T(W)M &\subseteq T(Z)M. \end{aligned} \tag{2}$$

Combining relations (1) and (2), we get

$$G(W) \subseteq G(X)$$
$$T(W) \subseteq T(X)$$

and by Lemma 2, W = X.

Then $G(W)M \subseteq G(Z)M \subseteq G(X)M$ implies $G(Z)M = G(X)M$. Similarly, $T(Z)M = T(X)M$. Also, $L(Z)M = \emptyset = L(X)M$. Then

$$C(Z) = [C(X)M] \cup [C(Y)-M].$$

ii) Suppose $L(N)M \neq \emptyset$. By Lemma 10, $C(X)-M \neq C(Y)-M$ implies $L(X)M = L(Y)M = L(N)M$.

Choose $k \in L(N)M$. Set $c_k = -1$. For all $i \in [M(G(N) \cup A(N))]-C(X)$, set $c_i = 1$. For all other i, set $c_i = \emptyset$.

$\Gamma_c(X) = c_k < \emptyset$ implies

$\not\vdash(c^M) < \emptyset$ implies

there exists a vulnerable region Z such that

$$C(Z)-M = C(Y)-M$$

$$\Gamma_c(Z) < 0.$$

$G(Z)M \subseteq G(X)M$ and $T(Z)M \subseteq T(X)M$ by the same argument as in i). $L(Z)M = L(N)M$ by Lemma 10.

Then applying the same construction, replacing Y by X and X by Z, there exists a vulnerable region W such that

$$C(W)-M = C(X)-M$$

$$G(W)M \subseteq G(Z)M$$

$$T(W)M \subseteq T(Z)M.$$

Then $G(W) \subseteq G(X)$ and $T(W) \subseteq T(X)$, so $W = X$.

From this,

$$G(Z)M = G(X)M$$

$$T(Z)M = T(X)M$$

and we have already stated

$$L(Z)M = L(N)M = L(X)M.$$

Then

$$C(Z) = [C(X)M] \cup [C(Y)-M]$$

Figure 1.

Example of Condition (2) Holding for a Structural Module

$$M = \{1, 2, 3\}$$

$$C(\{x\}) = \{2, 4\}$$

$$C(\{x,y\}) = \{1, 3, 4\}$$

$C(\{x\})M = \{2\}$ $\quad$ $C(\{x\})-M = \{4\}$

$C(\{x,y\})M = \{1, 3\}$ $\quad$ $C(\{x,y\})-M = \{4\}$

$C(\{x,y\})-M = C(\{x\})-M$ implies this is a valid structural module.

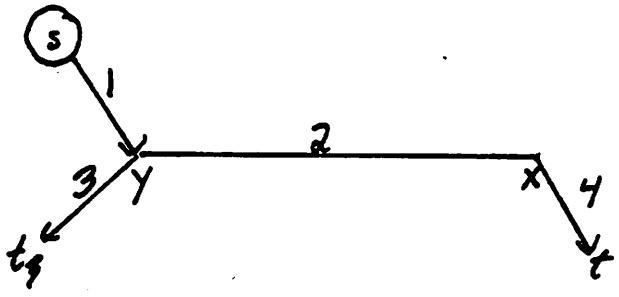


Figure 2.

Example of a State Space V for which Theorem 5 is Invalid

Let

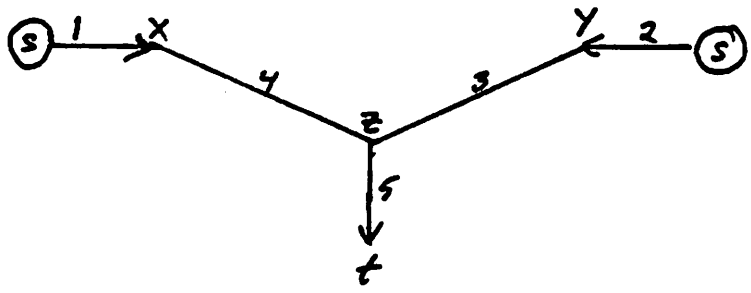
$c_1 = 5$ or 0

$c_2 = 2$ or 0

$c_3 = 2$

$c_4 = 5$ or 0

$c_5 = -7$



The vulnerable regions are {x,z}, {y,z}, {x,y,z}, {z}.

State Vector					Gale Measure of			
					{x,z}	{y,z}	{x,y,z}	{z}
5	2	2	5	-7	0	0	0	0
0	2	2	5	-7	-5	0	-5	0
5	0	2	5	-7	0	-2	-2	0
0	0	2	5	-7	-5	-2	-7	0
5	2	2	0	-7	0	-5	0	-5
0	2	2	0	-7	-5	-5	-5	-5
5	0	2	0	-7	0	-7	-2	-5
0	0	2	0	-7	-5	-7	-7	-5

$V_{\{z\}} \subseteq V_{\{y,z\}}$ $\quad V_{\{y,z\}} \subseteq V_{\{z\}} \cup V_{\{x,y,z\}}$ $\quad V_{\{x,z\}} \subseteq V_{\{x,y,\bar{z}\}}$

References

[BE,1965] Z. W. Birnbaum and J.D. Esary: Modules of Coherent Binary Systems. SIAM Journal on Applied Math. 13: 444-462.

[G,1960] David Gale: The Theory of Linear Economic Models. McGraw-Hill. 148, Theorem 5.3.

MODULAR DECOMPOSITION IN STOCHASTIC TRANSPORTATION NETWORKS

Andrew W. Shogan
University of California, Berkeley

ABSTRACT

Consider a flow network having random arc capacities and having associated with each node n a "supply-demand random variable" Y_n whose absolute value equals the supply available at the node when Y_n assumes a non-negative value and the demand required by the node when Y_n assumes a non-positive value. A fundamental problem is the computation of the reliability R, that is, the probability that the random variables will assume values that permit a feasible flow. Upon adapting the graph theoretic concepts of "cutnode" and "block," it is possible to identify a "block-module," an independent, non-trivial subnetwork that has one and only one node (the "cutnode") connected to nodes outside the subnetwork. The reliability of the network will increase by a known factor after a "block-modular decomposition" that consists of a transformation of the cutnode's supply-demand random variable and the deletion of the remainder of the block-module. Provided the original network possesses at least one block-module, R can be determined from a sequence of block-modular decompositions that reduce the original network to a single node whose reliability is easily computed. A discussion of the application of such a decomposition method to the analysis of electrical power networks is included.

1. INTRODUCTION

For the purposes of this paper, a stochastic transportation network is a flow network having random arc capacities and having associated with each node n a supply-demand random variable Y_n whose absolute value equals the supply available at the node when Y_n assumes a non-negative value and the demand required by the node when Y_n assumes a non-positive value. An important application of stochastic transportation networks, and one that motivated their consideration in this paper, is their use in models of electrical power networks.

The range of a random variable is a set consisting of values the random variable assumes with non-zero probabilities. The range of a random arc capacity is assumed to be a finite set of non-negative integers, and the range of a node's supply-demand random variable is assumed to be a finite set of integers. In general, then, the range of a node's supply-demand random variable may include both positive and negative integers so that the node may supply units in some realizations of the network and demand units in others. Based on the range of its supply-demand random variable Y_n, a node n is referred to in one of four ways: source, sink, intermediate node, or random source-sink. In particular, if the range of Y_n consists solely of a single value v (i.e., Y_n is a constant), the node is a source if $v > 0$, a sink if $v < 0$, and an intermediate node if $v = 0$; however, if the range of Y_n consists of at least two values, node n is a random source-sink. Reference is made to two special types of random source-sinks; a random source (random sink) is a random source-sink whose range includes only non-negative (non-positive) integers.

It is assumed that the joint probability mass function of the random variables of the network is known. The exact degree to which independence must be assumed is discussed briefly later in this introduction and in greater detail in Section 4.

Given a realization of the stochastic transportation network, the realization is feasible (infeasible) and the network functions (fails) if a (no) flow exists satisfying the following constraints: (i) the flow in each arc is no greater than the value assumed by its random arc capacity, and (ii) for each node n with Y_n assuming a value v, the flow out of node n minus the flow into node n is at most v when $v \neq 0$ and equals 0 when $v = 0$. The fundamental problem is the computation of the reliability, that is, the probability that the random variables will assume values such that the stochastic transportation network functions.

A special class of stochastic transportation networks is that in which every arc capacity is a binary random variable, one node is a source having a supply of 1, one node is a sink having a demand of 1, and all other nodes are intermediate nodes. The literature refers to such networks by a variety of names; binary reliability networks is used herein.

As the bibliographies in [12] and [13] illustrate, there exists an extensive literature treating binary reliability networks. However, despite the importance of the problem, the literature treating the most general stochastic transportation network is scanty. The state-of-the-art is best represented by the algorithm of Doulliez and Jamoulle [6] which computes the reliability of a stochastic transportation network by efficiently partitioning the set of all possible network realizations. The algorithm as presented in [6] assumes that each node of the network

is either an intermediate node, a sink, or a random source; however, in light of the procedure of Section 4 of this paper for eliminating random source-sinks, the algorithm can be applied to the most general stochastic transportation networks. An earlier algorithm of Doulliez [5] and a later algorithm of Pang and Wood [11] are similar to [6] but not as efficient.

A technique long used in analyzing binary reliability networks is modular decomposition (cf. [2] and [3]). Intuitively, modular decomposition reduces computational effort by first identifying a complex but specially structured subnetwork whose random variables are independent of the random variables outside the subnetwork and then replacing this complex subnetwork with a simpler subnetwork. This paper demonstrates that the technique of modular decomposition is also useful in analyzing stochastic transportation networks. After Section 2 introduces some additional notation and definitions, Sections 3 and 4 develop two types of modular decompositions: series-parallel-modular decomposition and block-modular decomposition. Section 5 describes an algorithm used in block-modular decomposition; Section 6 contains a detailed example; and Section 7 discusses the use of block-modular decomposition in analyzing electrical power networks. Finally, Section 8 discusses both computational aspects of block-modular decompositions and an area of future research.

2. ADDITIONAL NOTATION AND DEFINITIONS

The concepts of undirected arc, path, cut, and flow have their usual meanings (cf. [7], pp. 2-10). All arcs are undirected; nodes are adjacent if an arc of the network connects them, and an arc is incident

to the two nodes it connects. An arbitrary indexing of the arcs of the network permits reference to the arc connecting node i and node j and having index k in one of two ways: arc (i,j) or arc k.

Given a network N, the subnetwork defined by a set of nodes S consists of the nodes of S and every arc of N incident to a pair of nodes both in S. The subnetwork is proper if it is not N itself and is nontrivial if it contains more than one node.

"Random variable" and "probability mass function" are denoted by "r.v." and "p.m.f.", respectively; r.v.'s and p.m.f.'s denote their plurals. A r.v. of a stochastic transportation network is an independent r.v. if it is statistically independent of all other r.v.'s of the network. A subnetwork of the stochastic transportation network is an independent subnetwork if all its r.v.'s are statistically independent of the r.v.'s outside the subnetwork, even though they may have arbitrary dependence among themselves.

Given a stochastic transportation network having reliability R, another network having reliability R* is c-equivalent, where c is a known constant, if $R = cR^*$. In cases where c=1, c-equivalent is shortened to equivalent.

Given a subset S of a set T having a finite number of elements, $\bar{S}$ denotes its complement in T, $|S|$ denotes its cardinality, and Pr[S] denotes its probability under some probability measure defined over T. A partition $\{S_k\}$ of S consists of disjoint subsets of S whose union equals S.

If every node i in a set S of nodes has a value v(i) associated with it, v(S) equals $\Sigma_{i \in S} v(i)$. For example, if the v(i) denotes the supplies (demands) at a set S of sources (sinks), then v(S) equals the

total supply (demand) within the set. Given two subsets of nodes S and T, (S,T) denotes the set of all arcs connecting a node of S to a node of T. If every arc (i,j) of the subset (S,T) has a value v(i,j) associated with it, v(S,T) equals $\Sigma_{(i,j)\varepsilon(S,T)} v(i,j)$. For example, if the v(i,j) denote the capacities of the arcs of a cut $(S,\bar{S})$, $v(S,\bar{S})$ equals the capacity of the cut.

3. SERIES-PARALLEL-MODULAR DECOMPOSITION

Given a path between two nodes r and s, the interior nodes are all nodes of the path except r and s. A series-module (s-module) is a path between two nodes r and s that satisfies four conditions: (i) the path contains at least one interior node, (ii) every interior node is an intermediate node, (iii) every interior node of the path is adjacent only to other nodes of the path, and (iv) the $n \geq 2$ arc capacities $X_1, X_2, \ldots, X_n$ of the path are independent of all other r.v.'s of the network. An equivalent network results from the replacement of the arcs of the s-module by a single arc from r to s having capacity $\min[X_1, X_2, \ldots, X_n]$. Hereafter, s-modular decomposition refers to such a replacement.

A parallel-module (p-module) is a subnetwork consisting of a pair of nodes r and s joined by $n \geq 2$ arcs whose capacities $X_1, X_2, \ldots, X_n$ are independent of all other r.v.'s of the network. An equivalent network results from the replacement of the p-module by a single arc from r to s having capacity $X_1+X_2+\ldots+X_n$. Hereafter, p-modular decomposition refers to such a replacement.

A subnetwork is a series-parallel module (s-p-module) if it can be reduced to a pair of nodes joined by a single arc through a sequence of s-modular and p-modular decompositions. Hereafter, s-p-modular decomposition refers to such a reduction. The concept of s-p-modular decomposition as used here is a straightforward adaptation of a similar concept long used in the analysis of binary reliability networks and first defined by Bodin [4]. Both an s-module and a p-module are special cases of an s-p-module; Figure 1 contains a more complex s-p-module (assuming every node except the left-most and right-most is an intermediate node).

Hereafter, it is assumed without loss of generality that the original network under consideration contains no s-p-modules. This assumption simplifies both notation and computation.

4. BLOCK-MODULAR DECOMPOSITION

A block-module (b-module) is an independent, proper, and nontrivial subnetwork containing one and only one node (referred to as the cut-node) adjacent to nodes outside the subnetwork; a minimal block-module is a b-module containing no proper subnetworks that are also b-modules.

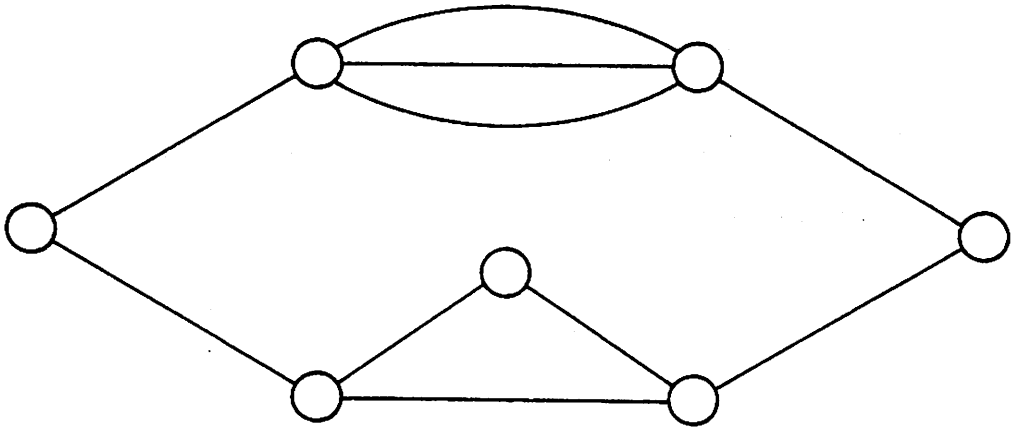


Figure 1

An explanation of the relationship of block-modules and cutnodes to the graph theoretic concepts of "cutnode" and "block" is omitted here but may be found in [14].

In order to simplify the discussion, it is assumed hereafter that a b-module possesses two additional characteristics: (i) the cutnode is an intermediate node, and (ii) the b-module contains no random source-sinks. As will now be shown, these assumptions are made without loss of generality. If (i) is not true, an equivalent network results from rerouting all arcs of the b-module incident to the cutnode n into a new artificial intermediate node n' and then connecting n' and n by an artificial arc having infinite capacity; node n' then serves as the new cutnode for the b-module. Hereafter, such a procedure is referred to as inserting an artificial intermediate node n' into the subnetwork at the node n. If (ii) is not true, there exists a random source-sink n with a supply-demand r.v. Y_n that can take on at least two values k for $-d \leq k \leq s$ where $d \geq 0$ and $s \geq 0$. An equivalent network with one less random source-sink results from changing node n into a sink with a constant demand of d, adding a new node n' having a constant supply of s + d, and adding a new arc (n',n) having a random capacity $X_n = Y_n + d$. When $X_n = k + d$ for $0 \leq k \leq s$, node n effectively becomes a source of k units since n' is able to supply the demand of d at n and still serve as a source of k units for the remainder of the network. Similarly, when $X_n = k + d$ for $-d \leq k \leq 0$, node n effectively becomes a sink for

-k units since only d + k of its demand of d can be supplied by n'. Note that the elimination of the random source-sink does not require that the supply-demand r.v. Y_n be independent.

The definition of a b-module motivates an attempt to replace the b-module with a simpler subnetwork. Given a particular realization of the random capacities of the arcs of a b-module, one of three cases may occur: (i) the sources within the b-module not only can meet all demands within the b-module but also can "export" units via the cutnode to the remainder of the network, (ii) the demands within the b-module can only be met if it is possible to "import" units via the cutnode from the remainder of the network, or (iii) the demands within the b-module cannot be met, regardless of how many units can be imported via the cutnode from the remainder of the network. Intuitively, then, with respect to the remainder of the network, the b-module acts in one of three ways: (i) a source, (ii) a sink with a demand that is possible to meet, or (iii) a sink with a demand that is impossible to meet. As will now be shown, a c-equivalent network results from changing the cutnode from an intermediate node into a random source-sink and deleting the remainder of the b-module from the network.

Consider a b-module N having cutnode n. N' denotes the subnetwork of the original network defined by the cutnode n and all nodes not belonging to the b-module N. (Actually, N' is itself a b-module.) S (T) denotes the set of nodes in N that are sources (sinks); for any source i, a(i) denotes its supply; for any sink i, b(i) denotes its demand. Suppose N contains r arcs having indices 1,2,...,r and random capacities $X = (X_1, X_2, \ldots, X_r)$; let $\Omega^+ = \{x \mid Pr[X=x] > 0\}$. X is the state vector of the b-module, and Ω^+ is the state space of the b-module.

Given a b-module N with cutnode n, N_k for $-\infty < k < \infty$ denotes a transportation network identical to N except that n is now either a source for k units if $k \geq 0$ or a sink for -k units if $k < 0$. For $-\infty < k < \infty$, let $H_k = \{x \in \Omega^+ | \text{when } X = x, N_k \text{ functions but } N_{k+1} \text{ fails}\}$; when $0 \leq k < \infty$, H_k is a set of states for which the sources within N not only can meet all the demands within N but can also "export" at most k units to N' via n; when $-\infty < k < 0$, H_k is a set of states for which the demands within N can only be met if it is possible to "import" at least -k units from N' via n. Let k_{max} equal the maximum of zero and the largest value of k for which H_k is non-empty; similarly, let k_{min} equal the minimum of 0 and the smallest value of k for which H_k is non-empty. Note that $0 \leq k_{max} \leq \max [0, a(S)-b(T)]$ and $-b(T) \leq k_{min} \leq 0$. Finally, let $H_{-\infty} = \{x \in \Omega^+ | x \in H_k \text{ does } \underline{\text{not}} \text{ hold for } k_{min} \leq k \leq k_{max}\}$; that is, $H_{-\infty}$ is a set of states for which the demands within N can never be met, regardless of how many units can be "imported" from N' via n.

It is clear that the subsets H_k for $k = -\infty$ and $k_{min} \leq k \leq k_{max}$ are a partition of Ω^+. In the remainder of this section, it will be assumed that H_k and $Pr[H_k]$ are known for $k = -\infty$ and $k_{max} \leq k \leq k_{min}$; Section 5 contains an efficient algorithm for their computation.

Since the entire network fails if $X = x$ where $x \epsilon H_{-\infty}$, conditioning upon whether or not $x \epsilon H_{-\infty}$ results in $R = R^*(1-\Pr[H_{-\infty}])$ where R^* is the reliability of the entire network given $x \epsilon H_{-\infty}$ does not occur. Consider the network obtained as follows: (i) change the cutnode from an intermediate node into a random source-sink having a supply-demand r.v. Y_n for which

$$\Pr[Y_n = k] = \begin{cases} \dfrac{\Pr[H_k]}{1 - \Pr[H_{-\infty}]} & \text{for } k_{min} \leq k \leq k_{max} \\ 0 & \text{otherwise} \end{cases}$$

and (ii) delete the remainder of the b-module from the network. Hereafter, b-modular decomposition refers to such a procedure. It is clear that b-modular decomposition results in new network having reliability R^*; that is, the new network is $(1 - \Pr[H_{-\infty}])$-equivalent to the original network.

B-modular decomposition will consist of an additional step in those instances where the cutnode for the b-module is an artificial intermediate node n' inserted into the original b-module at a random source-sink n. In particular, an equivalent network results from changing the supply-demand r.v. of n to $Y_n + Y_{n'}$, and then deleting the node n' and the arc (n',n) from the network.

As illustrated by the example in Section 7, a b-modular decomposition may create a new b-module in the revised network. Hence, the number of b-modular decompositions that can be performed need not be limited to the number of b-modules in the original network under consideration.

Provided the original network under consideration contains at least one b-module (and therefore two), there exists at least one sequence of $k \geq 1$ b-modular decompositions that reduces the original network to a revised network N containing no b-modules. Section 8 discusses the choice of such a sequence; for now, suppose it has made. Let n denote a node of N whose supply-demand r.v. is independent. The node that served as the cutnode for the immediately preceding b-modular decomposition is always one choice for n; for simplicity, it is assumed hereafter that n is always this node. After inserting an artificial intermediate node n' into the network at the node n, the node n' serves as the cutnode for one last b-modular decomposition. The resulting c-equivalent network consists only of the arc (n',n) and the two nodes n and n' having supply-demand r.v.'s Y_n and $Y_{n'}$, respectively; since this simple network behaves like a single random-source sink having supply-demand r.v. $Y_n + Y_{n'}$, its reliability R^* equals $\Pr[Y_n + Y_{n'} \geq 0]$ and is easily computed as $R^* = \Sigma \Pr[Y_{n'} = k]\Pr[Y_n \geq -k]$ where the summation is taken over all k for which $\Pr[Y_{n'} = k] > 0$. Having been reduced to a c-equivalent network consisting of a single node with reliability R^*, the original network has reliability $R = cR^*$.

It is interesting to consider the use of a sequence of b-modular decompositions to determine the feasibility of deterministic transportation networks; that is, the special class of stochastic transportation networks in which every r.v. equals a known constant with probability one. In such a network, the state space of a b-module consists of a single state vector x. If $x \varepsilon H_{-\infty}$, the entire network must be infeasible; however, if $x \varepsilon H_k$ for $k_{min} \leq k \leq k_{max}$, b-modular decomposition simply adds k to the cutnode's supply-demand constant and deletes from the network all other nodes and all arcs of the b-module.

5. PARTITIONING THE STATE SPACE OF A BLOCK-MODULE

Consider a b-module N having a cutnode n, a set S of sources with supplies a(i) for $i \in S$, a set T of sinks with demands b(i) for $i \in T$, a state vector $X = (X_1, X_2, \ldots, X_r)$, and state space Ω^+. This section develops an algorithm for constructing the partition $\{H_k\}$ of Ω^+ needed to perform a block-modular decomposition. The algorithm is based on a decomposition principle developed by Doulliez and Jamoulle in [6].

5.1 Overview of the Algorithm

Given vectors $m = (m_1, m_2, \ldots, m_r)$ and $M = (M_1, M_2, \ldots, M_r)$ having integer-valued components, the interval having lower endpoint m and upper endpoint M (denoted by [m,M]) is the set of vectors $x = (x_1, x_2, \ldots, x_r)$ having integer-valued components for which $m_j \leq x_j \leq M_j$ for $1 \leq j \leq r$. Let $\Omega = [L,U]$ denote the smallest interval containing Ω^+, where the size of an interval is measured by its cardinality. Such a Ω always exists since Ω^+ is a finite set of vectors all having integer-valued components. The algorithm in this section constructs a partition $\{G_k\}$ of Ω, where G_k is defined by replacing Ω^+ by Ω in the definition of H_k. Clearly, $H_k = G_k \cap \Omega^+$ and $Pr[H_k] = Pr[G_k]$ for all k.

It is helpful to think of the algorithm as a branching process that produces a rooted tree. The root of the tree corresponds to Ω and every other node of the tree corresponds to an interval contained in Ω. Branching from a node corresponds to partitioning the interval into several smaller intervals. At the end of each iteration of the algorithm, the leaves of the tree correspond to a partition of Ω into intervals.

Associated with each interval is a label k; if $k \geq 0$, the cutnode n is considered to have a demand of k units, and, if $k \leq 0$, n is considered to have a supply of -k units. An interval I with label k is either fathomed or unfathomed. If I has label k and is fathomed, $I \subseteq G_k$ and no further branching from I is necessary; that is, I will correspond to a leaf in the final tree produced by the algorithm. If I has label k and is unfathomed, although $I \cap G_j = \phi$ for $j > k$, further branching is necessary to determine if $I \cap G_k$ is non-empty.

Since $G_k = \phi$ for $k > a(S) - b(T)$, Ω is given the label $a(S) - b(T)$ at the algorithm's initialization; as indicated by the flow chart of Figure 2, each iteration of the algorithm consists of examining an interval I and, depending upon whether or not $I \cap G_k = \phi$, either reducing its label or partitioning it into one fathomed interval and several unfathomed intervals. Subsections 5.2-5.5 provide the details of the subroutines summarized in the flow chart; Subsection 5.6 discusses modifications to the algorithm and sensitivity analysis.

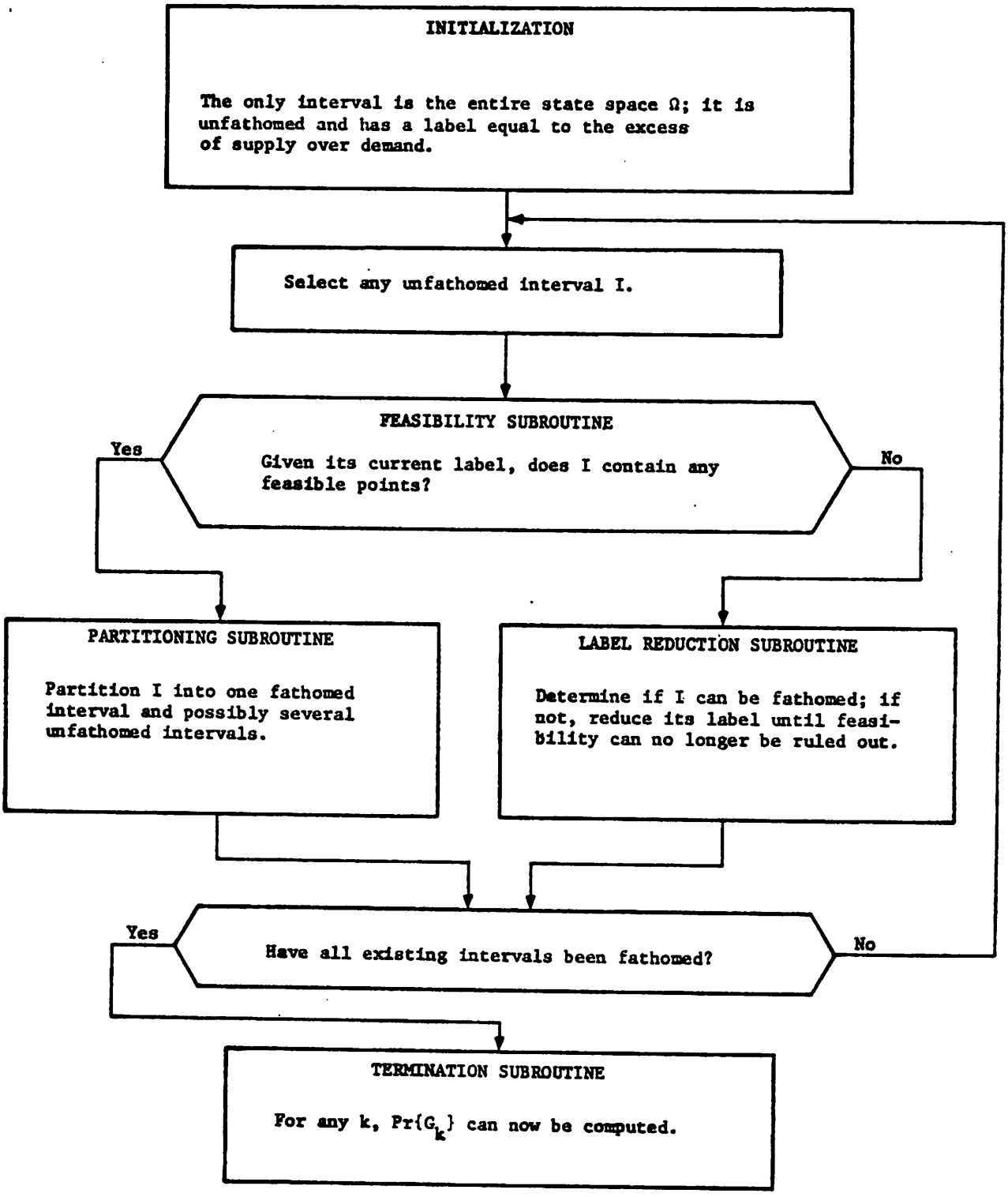

INITIALIZATION
The only interval is the entire state space Ω; it is unfathomed and has a label equal to the excess of supply over demand.
Select any unfathomed interval I.
FEASIBILITY SUBROUTINE
Given its current label, does I contain any feasible points?
Yes
No
PARTITIONING SUBROUTINE
Partition I into one fathomed interval and possibly several unfathomed intervals.
LABEL REDUCTION SUBROUTINE
Determine if I can be fathomed; if not, reduce its label until feasibility can no longer be ruled out.
Yes
Have all existing intervals been fathomed?
No
TERMINATION SUBROUTINE
For any k, Pr{G_k} can now be computed.

Figure 2

5.2 The Feasibility Subroutine

Given the interval chosen is

$$I = [(m_1, m_2, \ldots, m_r); (M_1, M_2, \ldots, M_r)]$$

having label k, execute the following steps:

(a) Augment the b-module by joining each source $i \in S$ to a common fictitious source s with an arc (s,i), by joining each sink $i \in T$ to a common fictitious sink t with arc (i,t), and by joining the cutnode n to both the fictitious source s and sink t with arcs (s,n) and (n,t).

(b) Define the arc capacities $c(\cdot)$ or $c(\cdot,\cdot)$ of the augmented network as follows:

$$c(j) = M_j \text{ for } 1 \leq j \leq r,$$

$$c(s,i) = a(i) \text{ for } i \in S,$$

$$c(i,t) = b(i) \text{ for } i \in T,$$

$$c(s,n) = \max(-k,0) \text{ and } c(n,t) = \max(k,0).$$

(c) Determine the max-flow from s to t in the augmented network using the Ford-Fulkerson labeling method [7, pp. 17-19] suitably modified to handle undirected arcs (cf. [7, p. 23] or [10, pp. 224-228]). For $1 \leq j \leq r$, denote the flow in arc j by f_j. Also, denote the min-cut by $(P,\overline{P})$; upon termination of the Ford-Fulkerson algorithm, P equals the set of labeled nodes, and $\overline{P}$ equals the set of unlabeled nodes.

(d) It is well-known (cf. [7, pp. 38-39]) that the deterministic transportation network having arc capacities M_j for $1 \leq j \leq r$ and having a supply of a(i) at $i \in S$, a demand of b(i) at $i \in T$, and either a supply of -k at n if $k < 0$ or a demand of k at n if $k > 0$ is feasible if and only if the max-flow in the augmented network equals $c(T,t) + c(n,t)$, or equivalently, $(T \cup n,t)$ is a min-cut. Hence, the interval I contains feasible points if and only if the max-flow in the augmented network equals $c(T,t) + c(n,t)$.

5.3 The Partitioning Subroutine

Given the max-flow of $c(T,t) + c(n,t)$ from s to t in the augmented network defined in the Feasibility Subroutine, execute the following steps:

(a) Let $v^o = (v_1^o, v_2^o, \ldots, v_r^o)$ where $v_j^o = \max(f_j, m_j)$ for $1 \leq j \leq r$.

(b) For $1 \leq j \leq r$, define F_j

as the minimum possible flow on arc j given that the max-flow flow from s to t in the augmented network must equal $c(T,t) + c(n,t)$. Then define $v^* = (v_1^*, v_2^*, \ldots v_r^*)$ where $v_j^* = \max(F_j, m_j)$. Note that $v_j^* = v_j^o = m_j$ if $f_j \leq m_j$ and that $v_j^* = v_j^o = M_j$ if arc j is a member of min-cut obtained in part (c) of the Feasibility Subroutine. Thus, if either $f_j < m_j$ or arc j belongs to the min-cut, v_j^* can be determined immediately. However,

if either of these two cases does not occur, F_j must be computed prior to determining v_j^*. The Ford-Fulkerson labeling method provides a convenient means for computing F_j. Suppose the direction of the flow in arc j is from node i to node k. Start the labeling process at node i and attempt to label node k; the only modification is that node k cannot be labeled directly from node i even if $f_j < M_j$. If a breakthrough of value ϵ occurs, increase by ϵ the flow in each forward arc of the flow augmenting path, decrease by ϵ the flow in each reverse arc of the flow augmenting path, and decrease by ϵ the flow in arc j. Thus, the value of the max-flow is unchanged but the flow in arc j has been reduced by ϵ. Repeat the labeling process until a breakthrough is is impossible. At termination, F_j is the current value of the flow in arc j. (Actually, since $v_j^* = \max(F_j, m_j)$, the algorithm can be terminated if the flow in arc j decreases to m_j or lower.)

(c) Partition the interval I into the intervals $A, B_1, B_2, \ldots, B_r$, $C_1, C_2, \ldots, C_r$ where

$$A = [(v_1^o, v_2^o, \ldots, v_r^o); (M_1, M_2, \ldots, M_r)]$$

and, for $1 \leq j \leq r$,

$$B_j = [(v_1^*, v_2^*, \ldots, v_{j-1}^*, m_j, m_{j+1}, \ldots, m_r); (M_1, M_2, \ldots, M_{j-1}, v_j^*-1, M_{j+1}, \ldots, M_r)]$$

$$C_j = [(v_1^o, v_2^o, \ldots, v_{j-1}^o, v_j^*, v_{j+1}^*, \ldots, v_r^*); (M_1, M_2, \ldots, M_{j-1}, v_j^o-1, M_{j+1}, \ldots, M_r)]$$

Note that $B_j = \phi$ if $v_j^* = m_j$ and $C_j = \phi$ if $v_j^* = v_j^o$. It is easy to show that these subsets are indeed a partition of I, especially upon noting that $\{B_j,\ 1 \leq j \leq r\}$ is a partition of

$$B \equiv \{x \in I | x_j < v_j^* \text{ for at least one } j\},$$

$\{C_j, 1 \leq j \leq r\}$ is a partition of

$$C \equiv \{x \in I | x_k \geq v_k^* \text{ for } 1 \leq k \leq r \text{ and } x_j < v_j^o \text{ for at least one } j\},$$

and $\{A,B,C\}$ is a partition of I. If additional details are needed, consult [6].

(d) Given the definition of v^o and the fact that $x \geq v^o$ for all $x \in A$, it is clear that $A \subseteq G_k$; hence, assign the label k to interval A and consider it fathomed. Given the definition of v_j^* and the fact that $x_j < v_j^*$ for all $x \in B_j$, it is clear that $B_j \cap G_k = \phi$; hence, assign the label k-1 to B_j and consider it unfathomed. Because $x \geq v^*$ and $x_j < v_j^o$ for all $x \in C_j$, assign the label k to interval C_j but consider it unfathomed.

5.4 The Label Reduction Subroutine

If the Feasibility Subroutine determines that the interval I contains no feasible points, execute the appropriate step below:

(a) If $n \in P$, reset the label of I to $-\infty$ and consider I fathomed.

(b) If $n \in \bar{P}$, reset the label of I to k-D where D is defined as the amount the max-flow in the augmented network falls short of $c(T,t) + c(n,t)$.

The justification for the above steps will now be provided.

It is well-known (cf. [7, pp. 38-39]) that $\bar{P} - t$ is a subset of nodes in the transportation network for which the difference between the net demand within the subset and the total capacity of all arcs entering the subset is the strictly positive amount D, thus violating a

necessary condition for feasibility in the theorem due to Gale [8]. In particular, one of the following must hold:

(1) If $n \in P$, then

$$b(T \cap \bar{P}) - a(S \cap \bar{P}) - c(P-s,\bar{P}-t) = D > 0.$$

In this case, step (a) is appropriate since the excess net demand in $\bar{P} - t$ can never be satisfied, regardless of the degree to which the boundary node n can serve as a supply.

(2) If $n \in \bar{P}$, then

$$b(T \cap \bar{P}) + k - a(S \cap \bar{P}) - c(P-s,\bar{P}-t) = D > 0.$$

In this case, step (b) is appropriate since if k were reduced by at least D, the interval I might then contain some feasible points. If $k > 0$ ($k<0$), reduction of k is equivalent to decreasing the demand (increasing the supply) at the cutnode.

5.5 The Termination Subroutine

When all existing intervals have been fathomed, execute the following steps:

(a) For each interval I, compute Pr{I}. If the arc capacities are independent,

$$Pr\{I\} = \prod_{j=1}^{r} \left(\sum_{x_j=m_j}^{M_j} Pr\{X_j=x_j\} \right);$$

if not, use the joint p.m.f. of X to compute

$$Pr\{I\} = \sum_{x_1=m_1}^{M_1} \sum_{x_2=m_2}^{M_2} \cdots \sum_{x_r=m_r}^{M_r} Pr\{X = (x_1,x_2,\ldots,x_r)\}.$$

(b) For $k = -\infty$ and $-b(T) \leq k \leq a(S) - b(T)$, G_k is simply the union of the non-overlapping intervals having label k. Hence, compute $Pr\{G_k\}$ by

$$Pr\{G_k\} = \sum_{\{I \mid I \text{ has label } k\}} Pr\{I\}.$$

5.6 Modifications to the Algorithm and Sensitivity Analysis

At the expense of additional and perhaps significant computation time, the algorithm may produce many intervals I for which $Pr\{I\} = 0$. Hence, in some cases (e.g., when $|\Omega^+|$ is significantly less than $|\Omega|$), it will probably be more beneficial to reduce computational effort by insuring inductively that $Pr\{I\} > 0$ for each interval I. More specifically, if $Pr\{X_j = v_j^o\} = 0$ ($Pr\{X_j = v_j^*\} = 0$) after defining v_j^o (v_j^*) in the Partitioning Subroutine, redefine v_j^o (v_j^*) by increasing it until a value v_j is reached for which $Pr\{X_j = v_j\} > 0$. Then, given v^o and v^*, if $Pr\{X_j = v_j^o - 1\} = 0$, ($Pr\{X_j = v_j^* - 1\} = 0$), instead of using $v_j^o - 1$ (v_j^*-1) in the Partitioning Subroutine as the j-th component of the upper limiting state space defining the interval C_j (B_j), use the largest value v_j less than $v_j^o - 1$ (v_j^*-1) for which $Pr\{X_j = v_j\} > 0$. Hereafter, this modification of the algorithm will be referred to as Modification A. Although the modified algorithm will no longer produce a partition of Ω, the subsets will still be non-overlapping and their union will include each $x \in \Omega^+$.

Other modifications to the algorithm, as well as a discussion of various forms of sensitivity analysis, are omitted here but may be found in [14].

6. AN EXAMPLE

The 10-node, 15-arc transportation network of Figure 3 serves as an example throughout this section. Nodes 1, 6, and 10 are sources, each having a constant supply of 20 units; nodes 2, 3, 5, 7, 8, and 9 are sinks, each having a constant demand of 5; node 4 is the only intermediate node. The random arc capacities are independent and denoted by X_j, X_j', and X_j''. for $1 \leq j \leq 5$, where each X_j takes on with equal probability one of the two values indicated in Figure 3 and where X_j, X_j', and X_j'' are identically distributed. Thus, the transportation network will be in one of 2^{15} equally likely states.

The subnetworks defined by the three sets of nodes {1,2,3,4}, {4,5,6,7}, and {4,8,9,10} are three identical minimal b-modules all having node 4 as a cutnode. One possible evaluation of the reliability of the network by a sequence of b-modular decomposition proceeds as follows:

(1) The subnetwork defined by the set of nodes {1,2,3,4} serves as the first b-module with node 4 serving as the cutnode. Figure 4 contains the tree produced by the partitioning algorithm of Section 5 when it is applied using Modification A to the b-module. Within a node of the tree, the second and third lines contain the interval's upper endpoint M and lower endpoint m, respecitively; the first line of node contains the interval's initial label and any subsequent changes (denoted by →). The remainder of Figure 4 is self-explanatory. Examination of the intervals corresponding to leaves of the tree confirms the following partition $\{H_k\}$:

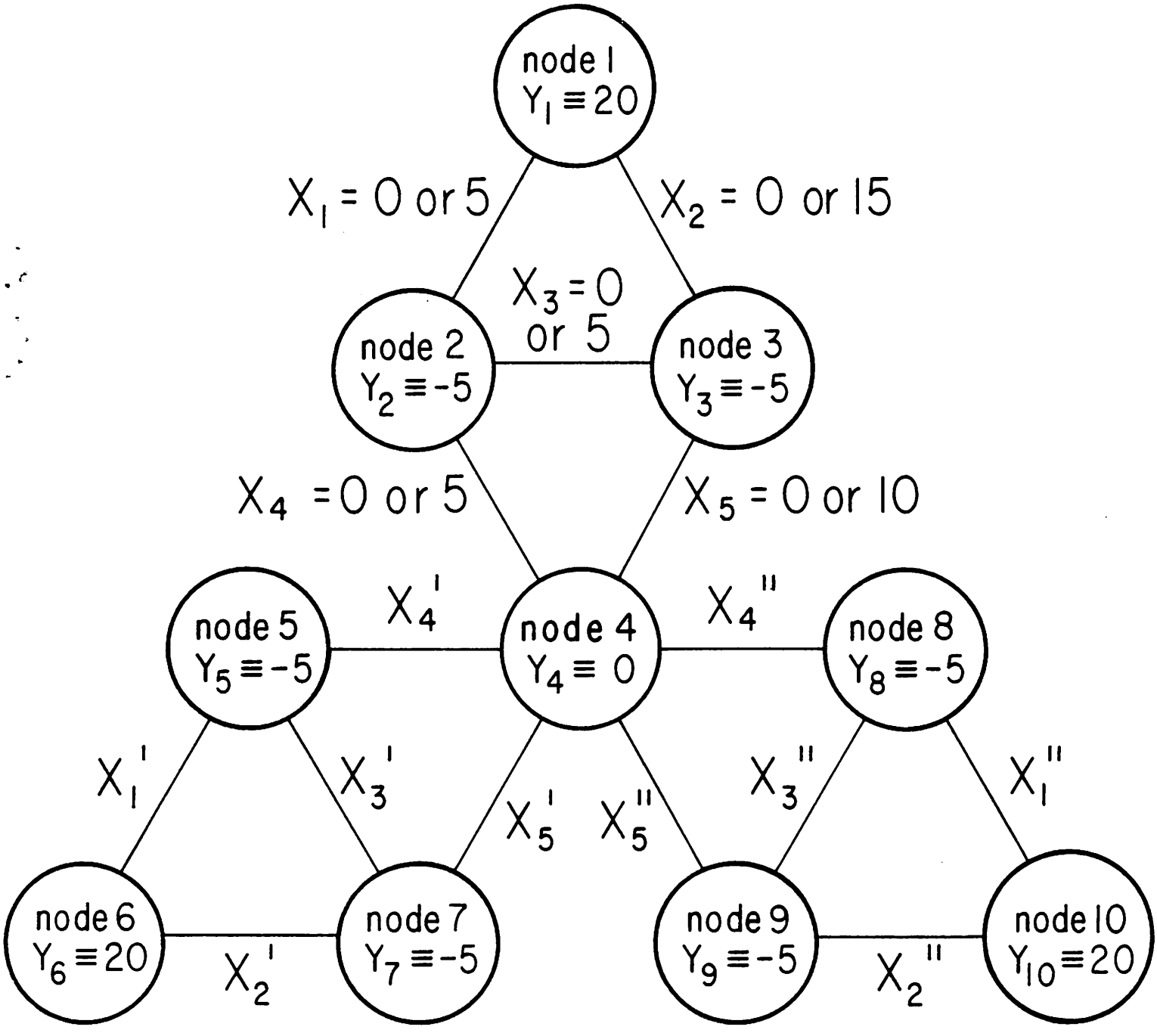

node 1
$Y_1 \equiv 20$
$X_1 = 0$ or 5
$X_2 = 0$ or 15
$X_3 = 0$ or 5
node 2
$Y_2 \equiv -5$
node 3
$Y_3 \equiv -5$
$X_4 = 0$ or 5
$X_5 = 0$ or 10
node 5
$Y_5 \equiv -5$
X_4'
node 4
$Y_4 \equiv 0$
X_4''
node 8
$Y_8 \equiv -5$
X_1'
X_3'
X_5'
X_5''
X_3''
X_1''
node 6
$Y_6 \equiv 20$
X_2'
node 7
$Y_7 \equiv -5$
node 9
$Y_9 \equiv -5$
X_2''
node 10
$Y_{10} \equiv 20$

Figure 3

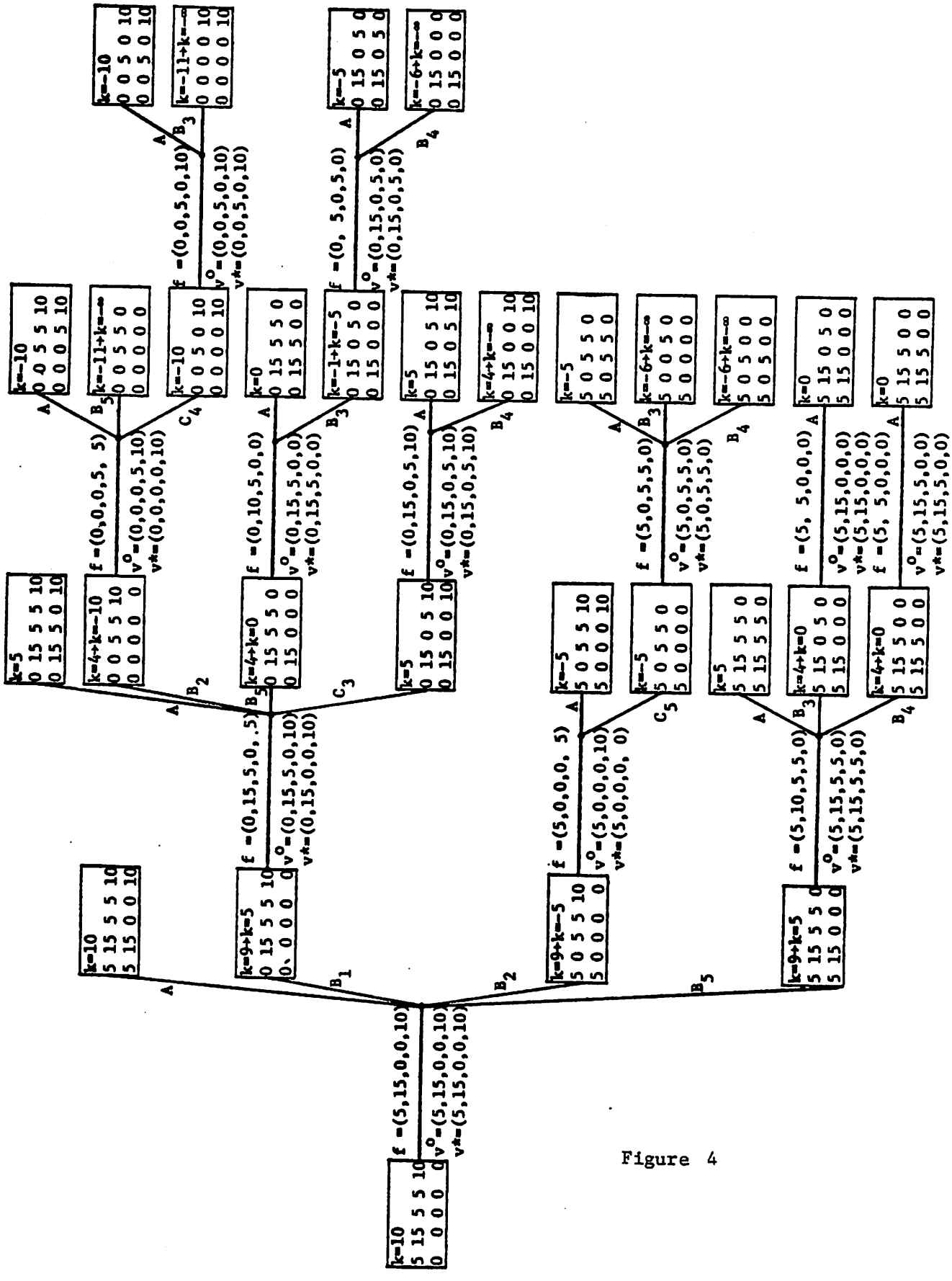


Figure 4

H_{10} = {(5,15,0,0,10);(5,15,0,5,10);(5,15,5,0,10);(5,15,5,5,10)}

H_5 = {(0,15,5,0,10);(0,15,5,5,10);(0,15,0,5,10);(5,15,5,5,0)}

H_0 = {(0,15,5,0,0);(0,15,5,5,0);(5,15,0,0,0);(5,15,0,5,0);
(5,15,5,0,0)}

H_{-5} = {(0,15,0,5,0);(5,0,0,0,10);(5,0,0,5,10);(5,0,5,0,10);
(5,0,5,5,10);(5,0,5,5,0)}

H_{-10}= {(0,0,0,5,10);(0,0,5,5,10);(0,0,5,0,10)}

$H_{-\infty}$ = {(0,0,0,0,0);(0,0,0,5,0);(0,0,5,0,0);(0,0,5,5,0);
(0,0,0,0,10);(0,15,0,0,0);(0,15,0,0,10);(5,0,0,0,0);
(5,0,0,5,0);(5,0,5,0,0)}.

Because each of the 32 possible state vectors is equally likely to occur, $Pr[H_{10}] = \frac{4}{32}$, $Pr[H_5] = \frac{4}{32}$, $Pr[H_0] = \frac{5}{32}$, $Pr[H_{-5}] = \frac{6}{32}$, $Pr[H_{-10}] = \frac{3}{32}$, and $Pr[H_{-\infty}] = \frac{10}{32}$. B-modular decomposition then deletes all nodes and arcs of the b-module except node 4 and results in the c-equivalent network of Figure 5(a) where node 4 is now a random source-sink having a supply-demand r.v. Y_4. From the values given above for the $Pr[H_k]$, $c = (1 - Pr[H_{-\infty}]) = \frac{22}{32}$ and $Pr[Y_4 = 10] = \frac{4}{22}$, $Pr[Y_4 = 5] = \frac{4}{22}$, $Pr[Y_4 = 0] = \frac{5}{22}$, $Pr[Y_4 = -5] = \frac{6}{22}$, and $Pr[Y_4 = -10] = \frac{3}{22}$.

(2) The next b-module consists of the subnetwork of Figure 5(a) defined by the set of nodes {4,5,6,7} with node 4 again serving as the cutnode. This time, the b-modular decomposition involves two stages since node 4 is now a random source-sink. In the first stage, an artificial intermediate node 4' is inserted into the b-module at node 4 (as illustrated in Figure 5(b)) in order to serve temporarily as the cutnode for a b-module identical to the one analyzed in step (1). Deleting all nodes and arcs of the b-module except node 4' results in the $\left(\frac{22}{32}\right)^2$-equivalent network of Figure 5(c) where node 4' is now a random source-sink having an independent supply-demand r.v. $Y_{4'}$, identically distributed to Y_4. The second stage of the b-modular decomposi-

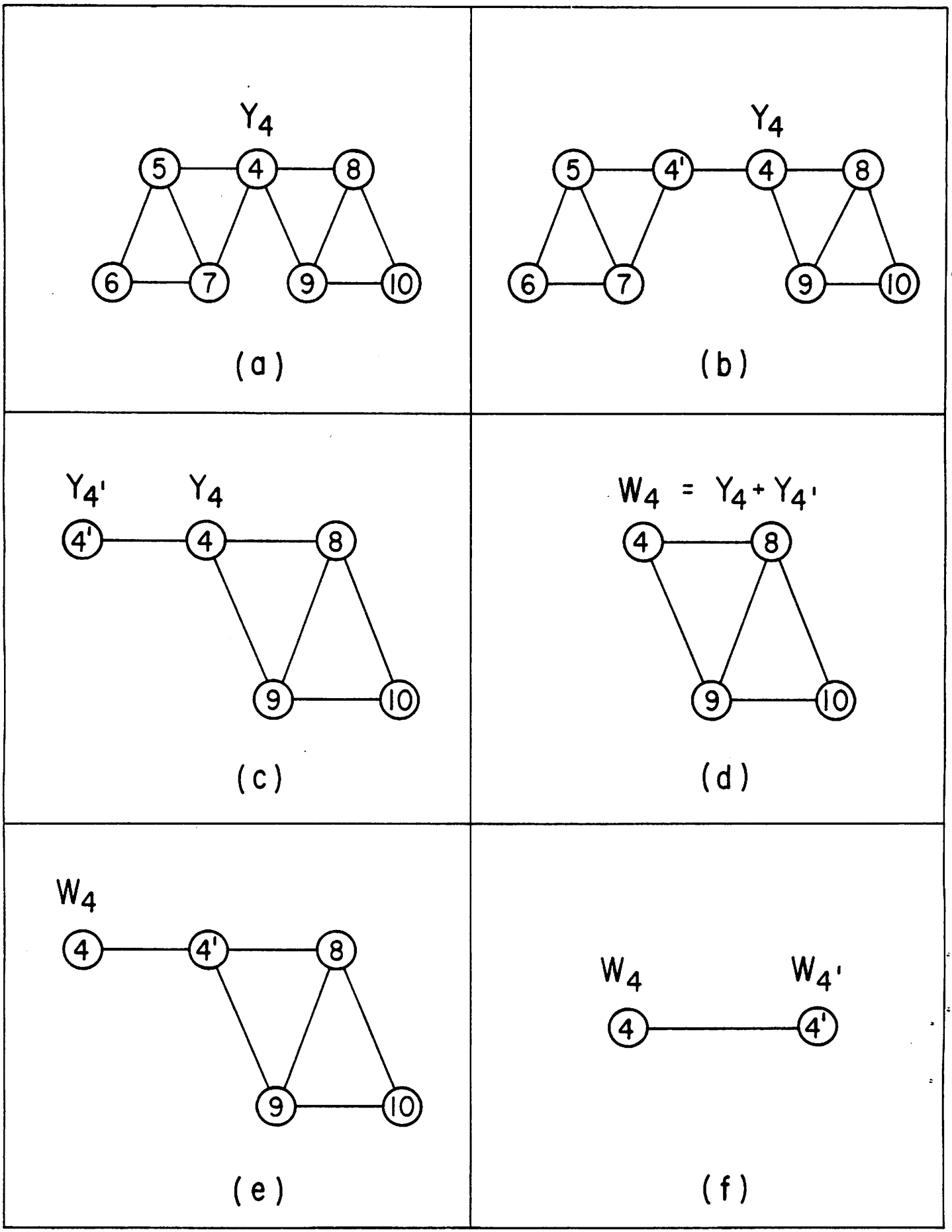

Y_4
5
4
8
6
7
9
10
(a)
Y_4
5
4'
4
8
6
7
9
10
(b)
$Y_{4'}$
Y_4
4'
4
8
9
10
(c)
$W_4 = Y_4 + Y_{4'}$
4
8
9
10
(d)
W_4
4
4'
8
9
10
(e)
W_4
$W_{4'}$
4
4'
(f)

Figure 5

tion combines nodes 4 and 4' into a single random source-sink having a supply-demand r.v. $W_4 = Y_4 + Y_4$, for which $\Pr[W_4 = 20] = \frac{16}{484}$, $\Pr[W_4 = 15] = \frac{32}{484}$, $\Pr[W_4 = 10] = \frac{56}{484}$, $\Pr[W_4 = 5] = \frac{88}{484}$, $\Pr[W_4 = 0] = \frac{97}{484}$, $\Pr[W_4 = -5] = \frac{84}{484}$, $\Pr[W_4 = -10] = \frac{66}{484}$, $\Pr[W_4 = -15] = \frac{36}{484}$, $\Pr[W_4 = -20] = \frac{9}{484}$; this results in the network of Figure 5(d).

(3) Since the network of Figure 5(d) contains no b-modules, the final b-modular decomposition also involves two stages. In the first stage, an artificial intermediate node 4' is again inserted into the network at node 4 (as illustrated in Figure 5(e)) in order to serve temporarily as the cutnode for a b-module identical to the one analyzed in step (1). Deleting all nodes and arcs of the b-module except node 4' results in the $\left(\frac{22}{32}\right)^3$-equivalent network of Figure 5(f) where node 4' is now a random source-sink having an independent supply-demand r.v. W_4, identically distributed to Y_4. The second stage of the b-modular decomposition combines nodes 4 and 4' into a single random source-sink having a supply-demand r.v. $W_4 + W_4$, and having a reliability $R^* = \Pr[W_4 + W_4 \geq 0]$; R^* is easily computed as

$$R^* = \Sigma_{k = -10,-5,0,5,10} \Pr[W_4 = k] \Pr[W_4 \geq -k] = \frac{6157}{10,648}.$$

Hence, the reliability R of the original network of Figure 3 is

$$R = cR^* = \frac{6157}{32,768}.$$

7. AN APPLICATION

Although the classic network flow of operations research is not a completely accurate model of the power flow in an electrical power network, the electrical power industry nevertheless regards a stochastic transportation network as a useful model for assessing the probability of meeting peak demand (cf. [15]). As an example, consider the network of Figure 6 that arises when modeling the major electrical power transmission network used by Pacific Gas and Electric (PG&E) to supply electrical power to northern California. The arcs represent PG&E's 500kV transmission system and each node represents a source and/or load demand of electricity. Whereas "s/d" within a node indicates it is a random source-sink of the most general type, "s" or "d" within a node indicates it is a random source or a random sink, respectively. Each random source represents either a major generation station or a 500kV substation where the total supply of power within the service area of the substation is always greater than or equal to the total demand by a random amount; a random sink represents a substation where the total load demand within the area is always greater than or equal to the supply by a random amount; a random source-sink of the most general type, however, represents an area that sometimes has excess power and at other times is short of power. The random sources include the node at the top of the figure representing potential power transfer of a random amount of PG&E from the Bonneville Power Administration in Oregon; the random sinks include the node at the bottom of the figure representing potential power transfer of a random amount from PG&E to Southern California Edison. The p.m.f.'s of the network's random arc capacities and random source-sinks are omitted here. The dotted lines in the figure outline independent subnetworks N_1, N_2, N_3, and N_4. A broad description of one possible sequence of b-modular decompositions that can be used to compute the network's reliability follows:

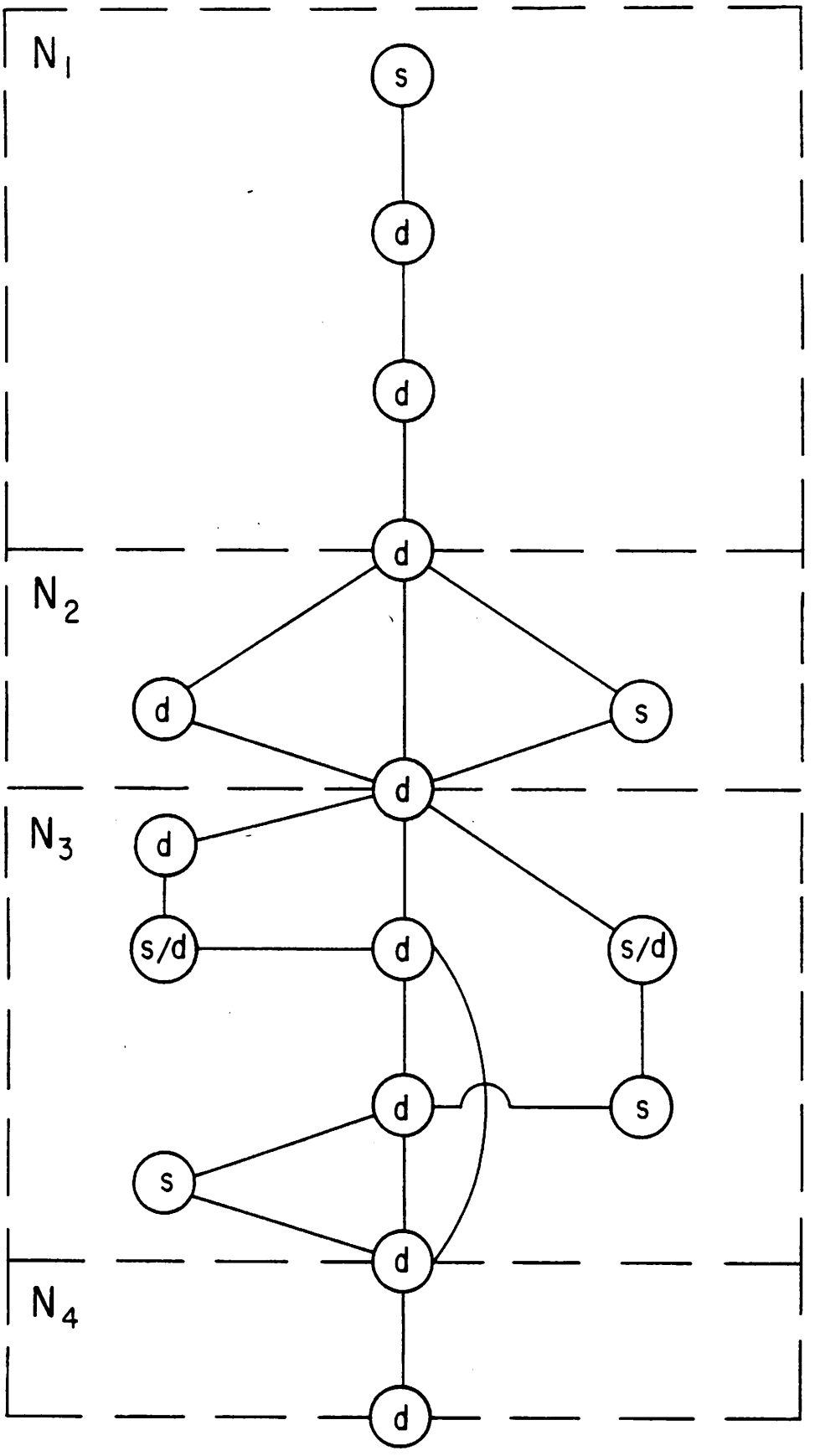


Figure 6

(1) N_1 and N_4 are both b-modules in the original network; two successive b-modular decompositions retain only the cutnodes N_1 and N_2 share with N_2 and N_3, respectively.

(2) The revised network consists only of the newly created b-modules N_2 and N_3 having a common cutnode; two successive b-modular decompositions collapse this revised network into a single random source-sink located at the common cutnode.

The existence of b-modules and cutnodes in PG&E's major (highest voltage) transmission network is due to the geography of California; that is, the Pacific Ocean on the west and the Sierra Nevada Mountains on the east dictated the linear nature of the network. B-modular decomposition will be of limited use in computing the reliability of a major transmission network located elsewhere in the United States (with the possible exception of Florida) since such a network will usually be highly redundant and, therefore, possess few (if any) b-modules and cutnodes. However, b-modular decomposition is more applicable to the lower voltage subtransmission and distribution networks (cf. [16]). More specifically, a subtransmission network connecting several substations usually has only a few cycles so that the existence of b-modules and cutnodes is likely, and a distribution network emanating from a substation is often a tree, in which case any node except a leaf of the tree may serve as a cutnode so that, for example, the reliability can be computed by a sequence of b-modular decompositions that successively "prune" branches of the tree.

8. CONCLUSION

Provided the stochastic transportation network under consideration contains at least one block-module, block-modular decomposition is an alternative to existing methods for computing the network's reliability. Instead of computing the reliability of one large network, it is possible to analyze a sequence of smaller subnetworks. Block-modular decomposition is particularly useful in the analysis of large electrical power networks.

A computer code implementing block-modular decomposition is under development. Upon its completion, experiments will provide an empirical answer to the question: "In performing a sequence of block-module decompositions that reduce the original network to a single node, are there heuristic rules for selecting the next block-module if the objective is to minimize the total computation time?". One possible answer is to always select the minimal block-module whose state space has the smallest cardinality. The experiments should also validate the conjecture that block-modular decomposition, where applicable, is computationally more efficient than existing methods; such a conjecture is based on the following:

(1) The concept of modular decomposition and other types of decomposition have proved quite effective in the analysis of binary reliability networks (cf. [2], [3], [12], [13], and references cited therein).

(2) Using modular decomposition, the reliability of the network of Figure 3 can be computed by hand in the manner discussed in Section 6; hand computation would be impossible using any other existing method.

(3) Block-modules are not only easily identified visually but also can be identified efficiently on a computer by a straight forward adaptation of an algorithm in Aho, Hopcroft, and Ullman [1] which computes a graph's articulation points and biconnected components (graph theoretic concepts similar to cutnodes and block-modules) in a number of steps proportional to the number of arcs in the network.

ACKNOWLEDGMENTS

This research was supported by the Department of Energy, Division of Electric Energy Systems, under Research Contract EC-77-S-01-5105. Also, I wish to thank Professor Felix F. Wu of the Department of Electrical Engineering and Computer Science of the University of California at Berkeley for his helpful comments and suggestions made during the preparation of this paper.

REFERENCES

[1] Aho, A. V., Hopcroft, J. E., and Ullman, J. D., The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass., 1974, pp. 179-187.

[2] Birnbaum, Z. W. and Esary, J. D., "Modules of Coherent Binary Systems," SIAM Journal of Applied Mathematics, Vol. 13, No. 2 (June 1965), pp. 444-462.

[3] Bodin, L. D., "Approximations to System Reliability Using a Modular Decomposition," Technometrics, Vol. 12 (May 1970), pp. 335-344.

[4] Bodin, L. D., "The Catalogue Ordering Problem," Technical Report ORC 67-39, Operations Research Center, University of California, Berkeley, Ca. (1967).

[5] Doulliez, P., "Probability Distribution Function for the Capacity of a Multiterminal Network," Revue Francaise d'Automatique, Informatique et Recherche Operationnelle, Vol. 1 (1971), pp. 39-49.

[6] Doulliez, P. and Jamoulle E.,"Transportation Networks with Random Arc Capacities," Revue Francaise d'Automatique, Informatique et Recherche Operationelle, Vol. 3 (November 1972), pp. 45-59.

[7] Ford, L. R. and Fulkerson, D. R., Flows in Networks, Princeton University Press, Princeton, New Jersey, 1962.

[8] Gale, D., "A Theorem on Flows in Networks," Pacific Journal of Mathematics, Vol. 7 (1957), pp. 1073-1082.

[9] Harary, F., Graph Theory, Addison-Wesley, Reading, Mass., 1969.

[10] Hillier, F. S. and Lieberman, G. J., Operations Research, Holden-Day, Inc., San Francisco, Ca., 1974.

[11] Pang, C. K. and Wood, A. J., "Multi-area Generation System Reliability Calculations," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-94, No. 2 (March/April 1975), pp. 508-517.

[12] Shogan, A. W., "Sequential Bounding of the Reliability of a Stochastic Network," Operations Research, Vol. 34, No. 6 (November/December 1976), pp. 1027-1044.

[13] Shogan, A. W., "A Decomposition Algorithm for Network Reliability Analysis," Networks, Vol. 8, No. 3 (1978), pp. 231-251.

[14] Shogan, A. W., "Modular Decomposition in Stochastic Transportation Networks," Technical Memorandum No. UCB/ERL M78/86, Electronics Research Laboratory, University of California, Berkeley, Ca. (September 1978).

[15] Sullivan, R. L., Power System Planning, McGraw-Hill Book Company, New York, 1977.

[16] Weedy, B. M., Electric Power Systems (second edition), John Wiley and Sons, New York, 1972.

Using Decomposition to Improve the Efficiency of Computing the Reliability of a Capacitated Flow Network

Jane Nichols Hagstrom

University of California, Berkeley

ABSTRACT

Reliability computations for an undirected transportation network with multiple sources and sinks are extremely time-consuming at best. It is here proposed that using the combinatorial technique of decomposing the underlying graph into its triconnected components is frequently a useful first step toward minimizing computation time. This is considered particularly for the case where capacity and demand constraints are independent, discrete random variables.

October 15, 1979

Using Decomposition to Improve the Efficiency of Computing the Reliability of a Capacitated Flow Network

Jane Nichols Hagstrom

University of California, Berkeley

1. Introduction

It is often desirable to calculate the probability of successful operation of the transportation network of some commmodity such as oil, water, electric power. Related problems for which much work has been done are binary coherent system reliability and communication network reliability. The introduction of capacities to the arcs of a network greatly increases the difficulty of doing an efficient computation.

The main thrust of this paper will be to introduce a general concept of decomposition of a transportation problem into smaller problems. Under many conditions, the problem will be most efficiently solved by solving the smaller prob-

lems and building up the solution from these smaller problems. Three papers which deal with the same problem are those of Doulliez and Jamoulle [1972], Shogan [1978], and Willie [1979]. Shogan's paper and this one are intimately related; in fact the main contribution of this one over that is the more general framework introduced. Like Shogan's, this paper will resort to a method based on that of Doulliez and Jamoulle when it comes to proposing a specific method for solving the subproblems. Willie's method is significantly different from the other papers' and is not readily adaptable to building up the solution from the solutions to subproblems since he restricts himself to situations in which all components of the system have only two possible states.

We will introduce at this point an example which will be followed throughout the rest of the paper. We are given the transportation network whose graphical representation is shown in Figure 1. There are three vertices of the graph which are special: one represents a point of supply (s), one a point of demand (d), and one a point of both supply and demand (s/d). All other vertices are simply points where there is a choice of route for shipment. The numbered edges represent transportation paths between vertices. The amount of supply, demand, or capacity of the shipping routes will be assumed to be independent, discrete random variables. We will indicate how to calculate the probability that the network is able to supply the demands.

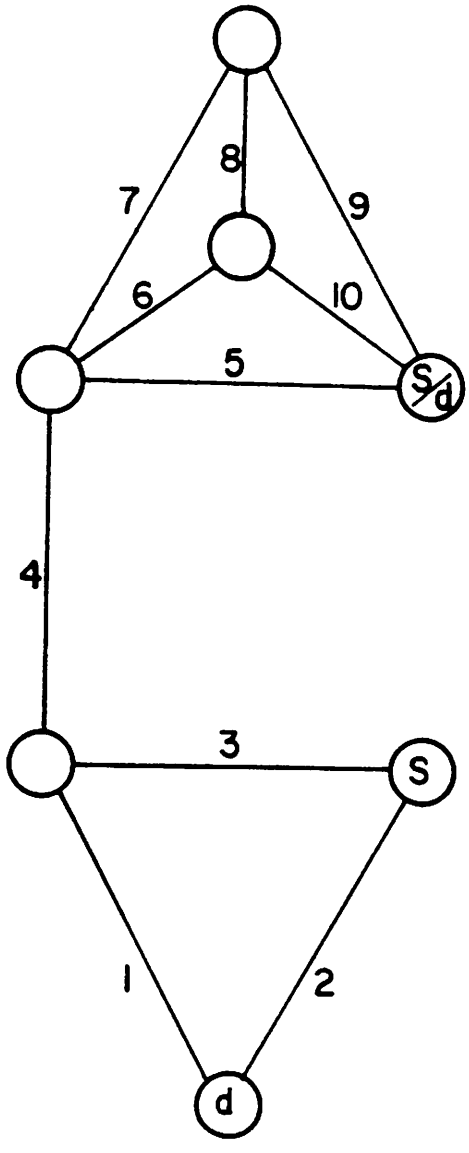


Figure 1

The organization of the rest of the paper is as follows. In the next section, we will clarify details of the problem and model. The following section will describe the decomposition concept and procedure for performing it. The last section will describe the general method of using the decomposition and specifically how it might be implemented.

2. Problem and Model

The deterministic version of this problem, sometimes known as the supply problem or transshipment problem, is studied in Gale's book [1960]. Given a transportation network with routes with fixed capacity, supply points with fixed capacity, and demand points with fixed requirements, he characterized the circumstances under which there is a feasible assignment of commodity flows to the possible routes so that all demands are satisfied. Unfortunately, this characterization does not in general suggest good algorithms, even for solving the deterministic case. We will be dealing with a stochastic version of this problem, in which all capacities and demands are independent, discrete random variables, and we will ask what the probability of the existence of a feasible flow is. As pointed out before, Figure 1 is an illustration of such a problem.

For this problem, it will be convenient to construct a slightly different graphical representation. To the vertices already described, we will add a new vertex, a super source-sink. To the super source-sink, we will draw edges

from every supply point and every demand point. If we require a flow on this new graph such that the flow from the super source-sink along an edge to a supply point does not exceed the capacity of that supply point and that the flow into the super source-sink along an edge from a demand point exceed the requirements at that point, we can consider all constraints of the problem to be associated with edges (requiring, of course, flow conservation at every vertex). This procedure has been performed in Figure 7.

At this point, let us introduce some terminology and notation. A graph (Often what is defined here is called a multigraph.) G = (V,E) consists of a set of vertices V and a set of edges E, such that each edge is incident with exactly two vertices. The edge is directed if one of those vertices is identified as the head of the edge and the other is the tail. The graph is directed if every edge is directed. If every edge is undirected, the graph is undirected. Except where stated otherwise, a reference to a graph will mean an undirected one. If a graph represents a flow network, we may identify certain vertices as sources, in which case, flows out of them may exceed flows into them, or sinks, where the opposite case may or must hold.

A subgraph of G is a graph whose vertices are a subset of V and edges are a subset of E such that its vertices and edges have the same incidence relation that they did in G. Let V_0 be a subset of V. Then $G(V_0)$, the subgraph induced

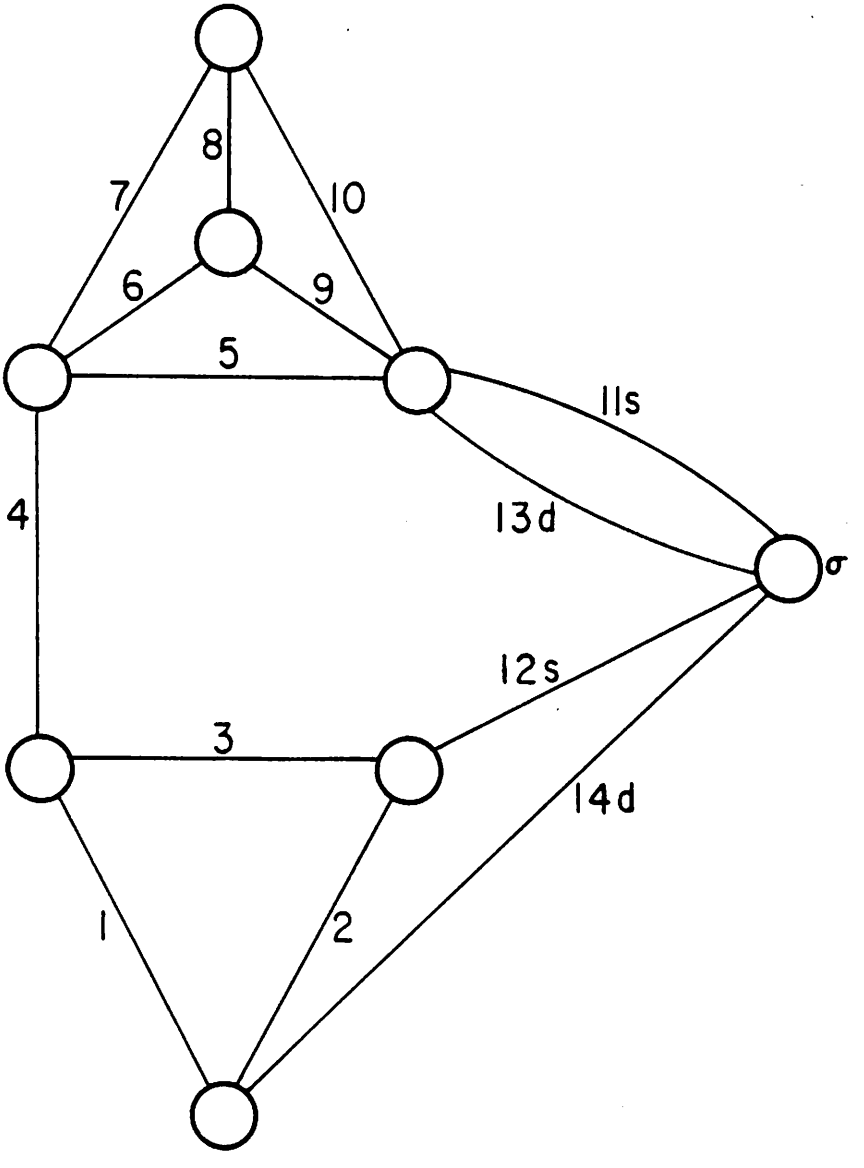


Figure 2

by $V_\emptyset$, is the subgraph of G whose vertices are $V_\emptyset$ and whose edges are those which are incident only to vertices of $V_\emptyset$. Let $E_\emptyset$ be a subset of E. Then similarly, the subgraph induced by $E_\emptyset$, $G(E_\emptyset)$, is the subgraph of G whose edge set is $E_\emptyset$, and whose vertices consist of those to which some member of $E_\emptyset$ is incident.

A path is an alternating list of vertices and edges $(v_\emptyset, e_1, v_1, e_2, \cdots, e_k, v_k)$ such that e_i and v_i are incident and v_i and e_{i+1} are incident. A graph is connected if there is a path between every pair of vertices.

Consider a graph on which we wish to define a flow. There may be certain vertices designated as sources and certain as sinks. If the graph is undirected, choose arbitrary directions for each edge. A flow on the graph is an assignment of a real number f_i to each edge i called the flow on that edge such that

1) For every vertex which is a source, the sum of the flows on edges into that vertex is no more than the sum of the flows out of that vertex.

2) For every vertex which is a sink, the sum of the flows on edges out of that vertex is no more than the sum of the flows into that vertex.

3) For every other vertex, flow conservation holds, that is, the sum of the flows on edges into that vertex equals the sum of the flows on edges out of that vertex.

A flow on a network is of no interest unless there are

constraints. Constraints may be of the form of an upper bound or lower bound on the flow on an edge or they may be vertex capacity constraints or they may be constraints associated with sources and sinks. We will consider only the first and the last. For a source, a constraint will be on the amount by which the flow out may exceed the flow in. For a sink, a constraint will be on the amount by which the flow in must exceed the flow out.

As was suggested above, we can alter the graph of a flow network so that there are only edge constraints and no source or sink constraints. Furthermore, sources and sinks lose their identity and flow conservation must hold at every vertex. Consider the constraints in our example problem. Since they are random, we do not know their actual value, but we do know their form. Let σ be the super source-sink we added to the graph. Our original edges, which were not incident with σ, were undirected. The capacity constraint applies whether we have flow in one direction or the other. Then if we arbitrarily choose a direction for such an edge i, the flow f_i will be positive or negative depending on whether it uses the edge in a forward or backwards direction, and will be constrained to lie between c_i and $-c_i$, where c_i is the capacity of that edge. Suppose i is an edge incident with σ and a source (supply point). Let us direct i away from σ. Then the flow on i is constrained to lie between 0 and c_i, where c_i is the supply capacity of the source. Actually, to maintain feasibility and maximize

excess supply, these flows will always be positive, so that we can consider the flow on such an edge to have an upper bound of c_i and no lower bound. If i is an edge incident with σ and a sink, it is natural to direct that edge towards σ and speak of the flow being constrained to be greater than the demand at the sink. For simplicity of later discussions, it turns out to be more convenient to direct the edge away from σ, and assign the flow a (negative) upper bound of c_i where $-c_i$ is the demand at the sink.

We now have just two types of edges, ones incident with σ which have an upper bound of c_i and no lower bound, and the rest which have an upper bound of c_i and a lower bound of $-c_i$. Soon we will introduce some fictional edges, some of which will have constraints on them. The same two cases for the constraints will hold.

Figure 3 shows all the edges for our example with directions. We will here explain the purpose of the added edge 15. We are going to solve a slightly more general problem than that of finding the probability that there exists a feasible flow on a network whose constraints (the c_i) are random. Two possible versions of this general problem exist. In the first version let us concentrate on a certain demand point. We might ask "What is the probability that all other demands in the network are met and the excess supply available at this demand point exceeds a certain number?" The other version allows us to look at a vertex

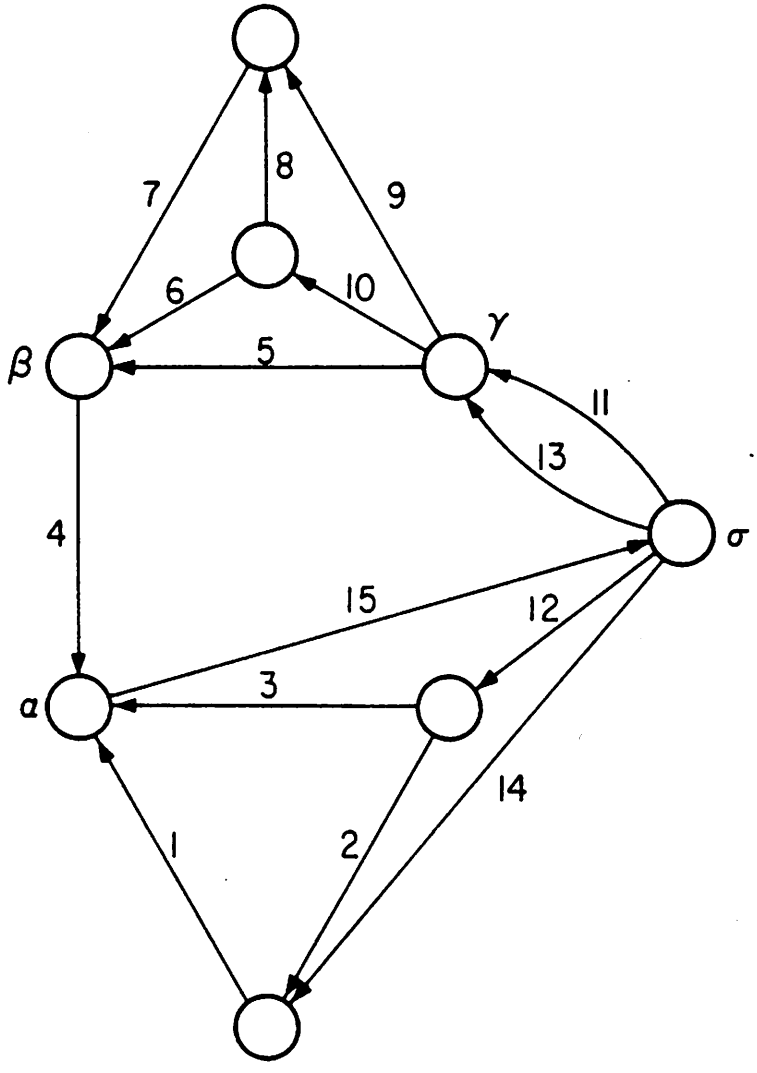


Figure 3

which is not a demand point and ask "What is the probability that all demands in the network are met and an excess supply greater than a certain number could be delivered to this vertex if it could be used?" This second version is the kind of question an expansion planner might ask.

Thus we are going to do the following. We are going to select a vertex of the graph which is for some reason of particular interest to us, and we are going to compute the (defective) distribution of the excess supply, which may be negative, available at that vertex when all demands are met. It is defective because for each value k, we will find

$$\text{Prob}\{V \leq k \text{ and all demands are met}\}$$

In general, for no value of k will this probability be 1. A slight variation in approach would give us for each k,

$$\text{Prob}\{V \leq k \text{ and all demands are met whenever possible}\}.$$

This distribution would not be defective.

By adding an edge from our chosen vertex to σ and finding the probability distribution of the maximum flow V on this edge when the flow is feasible on the network, we will have accomplished this task. In Figure 3, vertex α has been chosen and edge 15 has been added.

To review what we will get by doing this: For any realization of the $\tilde{c}_i$, one of three possible cases will hold. First, there may be no feasible way of supplying the demands associated with vertices other than our chosen one. Second,

there may be a feasible flow which yields a positive maximum flow of value k through our added edge, in which case it is possible to supply all demands and have an excess supply of k available at the chosen vertex. Third, there may be a feasible flow which yields a negative maximum flow -k through our added edge, in which case it is necessary to supply an added quantity k at our chosen vertex in order for there to be a feasible flow. If the chosen vertex is a demand point, we will have more information than just the feasibility or infeasibility of the network, we will also know by how much the demand at the chosen vertex is exceeded or undersupplied. If it is not a demand point, we get information on the feasibility and the desirability of making that vertex either a supply or demand point. In the probabilistic form of the problem, this information is in the form of a probability distribution.

From now on we will assume we have a graph of the form in Figure 3, with all sources and sinks joined by edges to a special vertex σ, and an added edge joining some selected vertex to σ. When we wish to consider the directed version of the graph, we will have our added edge directed into σ, all other edges incident to σ directed out of σ, and the directions on all other edges arbitrary. In the next section, we will forget the directions, and look again at them in the last.

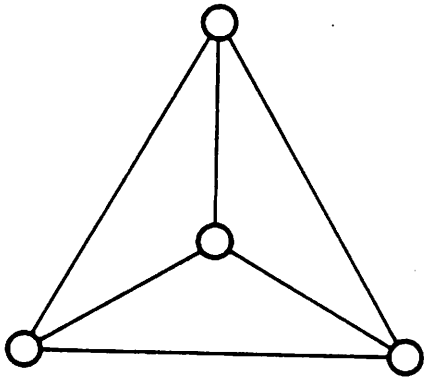
3. Decomposition

In this section, we will describe a combinatorial notion of decomposition of a graph. The same notion may be applied to matroids and blocking systems, as described by Cunningham and Edmonds [1979]. We will have to start with more definitions.

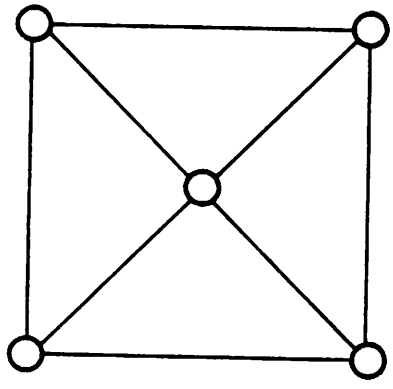
A subset V_0 of the vertices of a graph will be said to disconnect a graph G if the subgraph of G induced by the complement of V_0 is not connected, or if it consists of a single vertex. We say a graph is k-connected if the smallest set of vertices which disconnects the graph has more than k elements. As drawn in Figure 1, our graph is singly connected but not biconnected. In Figure 2, it is biconnected but not triconnected. The graphs in Figure 4 are triconnected. Graph (e) is also 4-connected.

If a flow network is well-designed, then its graph with super source-sink σ will be biconnected. Otherwise one vertex can be removed to disconnect it. If σ can be removed, the network is composed of two independent networks which can be treated separately. Otherwise every other vertex lies on a path from a source to a sink and thus there are two paths from it to σ. Removal of any single vertex will leave all other vertices connected to σ.

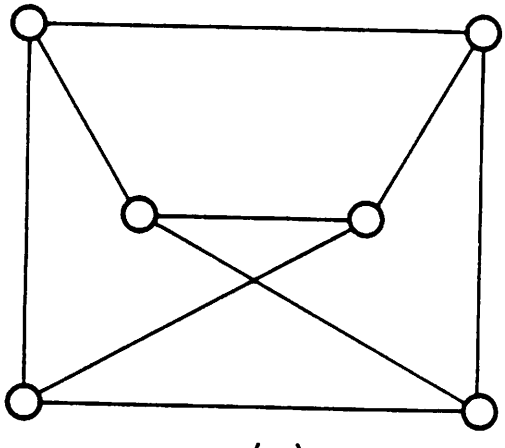
In order to decompose our biconnected graph, we want to partition the edges and associate them with smaller graphs



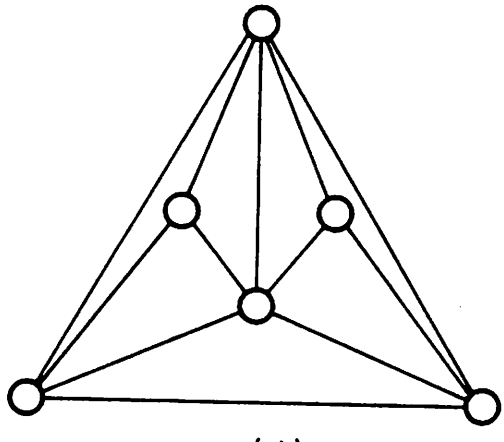
(a)



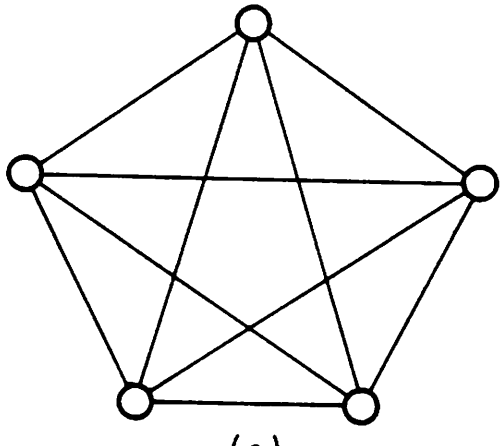
(b)



(c)



(d)



(e)

Figure 4

which will preserve information about sets of vertices which are triconnected. The conceptualization described here is due to Tutte [1966]. We look for a partition (E_1, E_2) of the edges of the graph G such that E_1 and E_2 each have at least two edges and the induced subgraphs $G(E_1)$, $G(E_2)$ have exactly two vertices in common. This pair of vertices is called a hinge, and we see that any path including a vertex from $G(E_1)$ and a vertex from $G(E_2)$ includes a hinge vertex. We call such a partition a split. In Figure 3, hinges are $[\sigma,\alpha]$, $[\beta,\gamma]$, $[\gamma,\sigma]$, $[\sigma,\beta]$, $[\alpha,\gamma]$. Two possible splits of the graph in Figure 3 at $[\alpha,\sigma]$ are shown in Figure 5. Splitting edge 15 away from the rest of the graph is conceptually possible but not productive and therefore disallowed.

We have lost information in this split. Consider a pair of vertices other than this hinge that lie in one of the generated subgraphs. They are connected by a path which passes through the hinge and along edges belonging to the other subgraph. This path contributes 1 to the degree of their connectivity. Then to the subgraph in which they lie, we will add a virtual edge joining the hinge to represent the lost path. The other subgraph will also need such a virtual edge. These two subgraphs with their virtual edges added are called split graphs of the original graph. In Figure 6, the split graphs generated by the split (a) in Figure 5 are shown. We wish to split our split graphs as long as any hinges remain. Looking at graph (b) in Figure 6, we see $[\sigma,\alpha]$ is again a hinge. Splitting at $[\sigma,\alpha]$ and

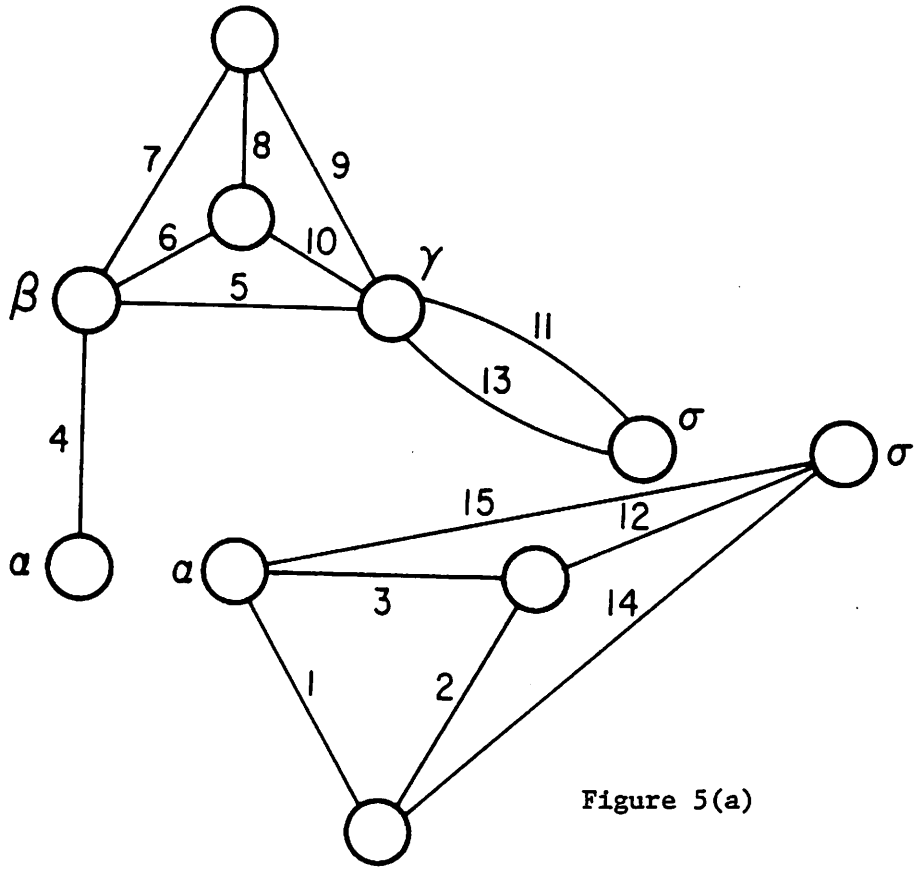


Figure 5(a)

(a)

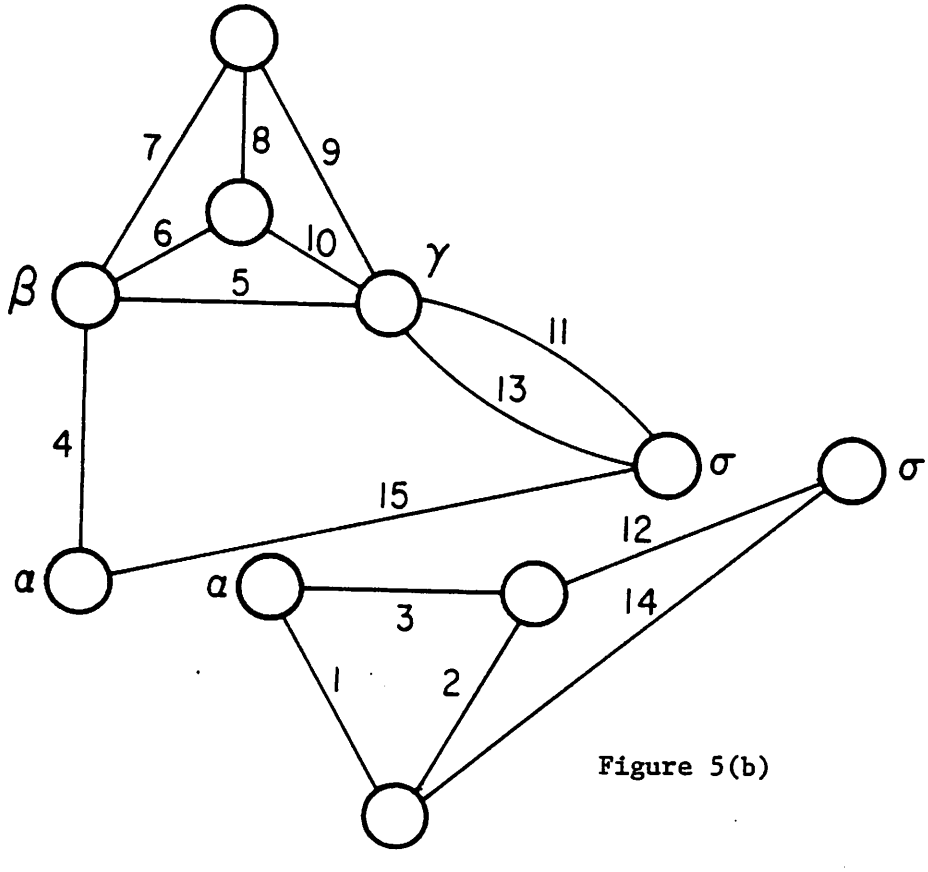
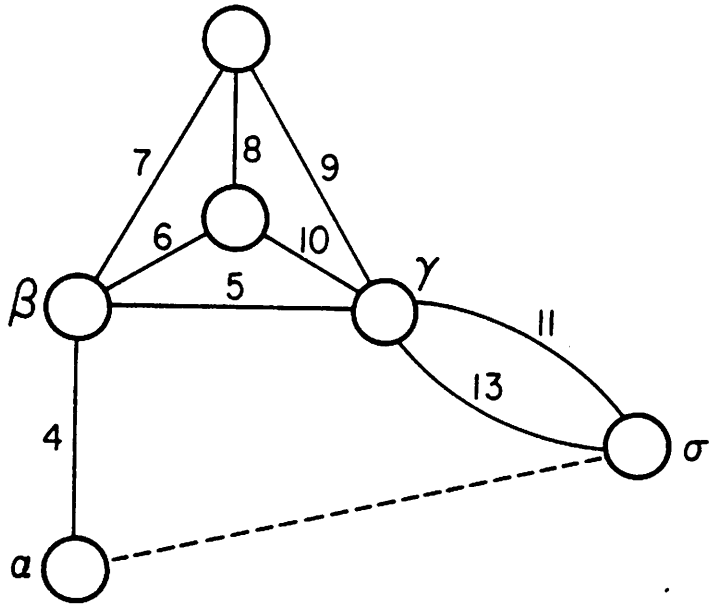
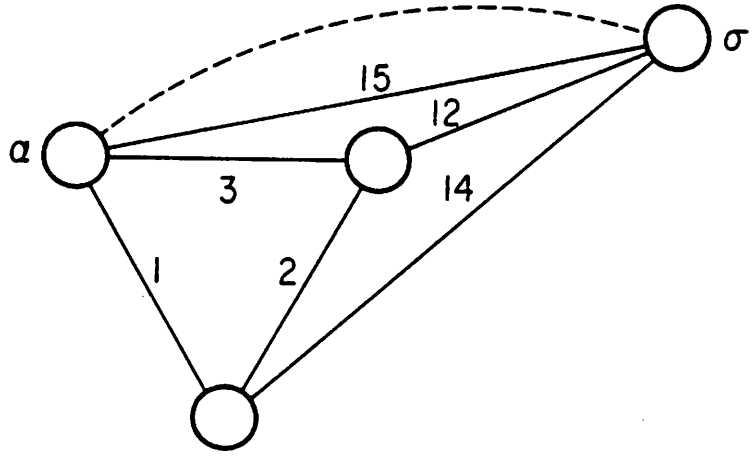


Figure 5(b)

(b)



(a)

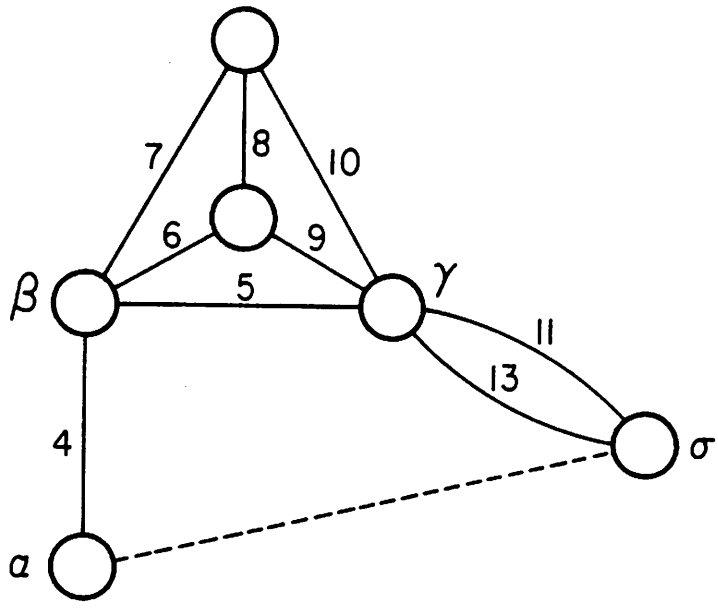


(b)

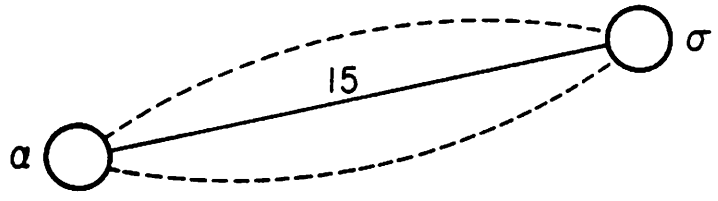
Figure 6

adding virtual edges corresponding to the split, we now have the three graphs illustrated in Figure 7. Note that we would have the same three graphs if we had started with split (b) shown in Figure 5 and then split again at $[\sigma,\alpha]$.

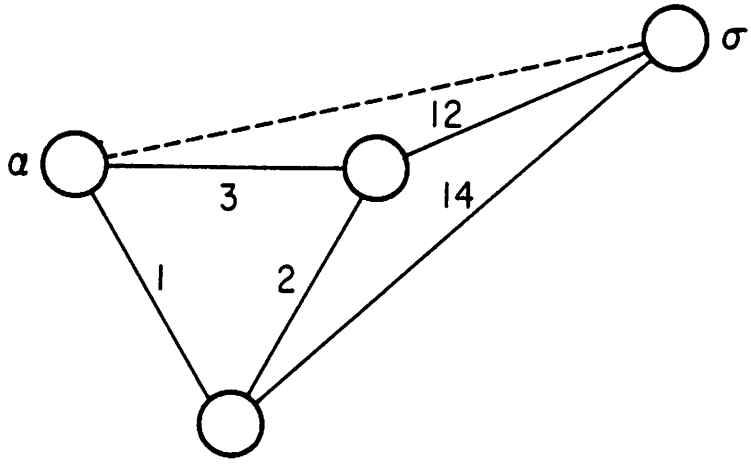
Now starting with the graphs shown in Figure 7, we can imagine splitting successively at hinges until no more splits on any of the graphs generated can be performed. As discussed in more detail in Cunningham and Edmonds [1979] and Hopcroft and Tarjan [1973], with two adjustments this decomposition of the graph is unique. First, at some point we split at either $[\alpha,\gamma]$ or $[\sigma,\beta]$ and in our final decomposition ended up either with a pair of triangles α,β,γ and γ,σ,α or the pair σ,α,β and β,γ,σ. In either case these triangles each have a virtual edge which was generated by the same split. Our rule will be: Merge any pair of polygons which have virtual edges generated by the same split back into a larger polygon by joining them at the hinge corresponding to the split and erasing the virtual edges, thereby undoing the split. Thus in our example, no matter which way we had done the split, we merge the triangles back into the polygon $\sigma,\alpha,\beta,\gamma$. A second problem that can arise is associated with bonds. A bond is a graph with two vertices and at least three edges. Graph (b) in Figure 7 is a bond. The problem did not arise in our example, but sometimes several bonds may be generated by successive splits at the same hinge. Our second rule is: Merge all bonds associated with the same hinge by joining them at the



(a)



(b)



(c)

Figure 7

hinge and erasing the virtual edges that arose when they were split from each other, thereby undoing that particular split.

After merger of any such graphs, we will have a unique collection of graphs called the set of triconnected components of our original graph. Each of the triconnected components is either a truly triconnected graph, a bond of three or more edges, or a polygon of three or more edges. Figure 8 contains the triconnected components of the example.

In performing this decomposition, we can form a map from which we can recreate the original graph. This map is in the form of a decomposition tree formed of nodes and arcs. Each node will correspond to a triconnected component of the graph. Two nodes will be joined by an arc if their corresponding components have virtual edges generated by the same split. This map is described as a tree since there is exactly one path of arcs between any pair of nodes.

At this point, it will be convenient to introduce more information into the tree. We have chosen some special vertex to concentrate on, and added a special edge to our original graph joining this vertex to σ. That special edge occurs in exactly one triconnected component of the graph. Call the corresponding node of the decomposition tree the root of the tree. Because the map is a tree, each node of the tree will have a distance from the root, defined by the

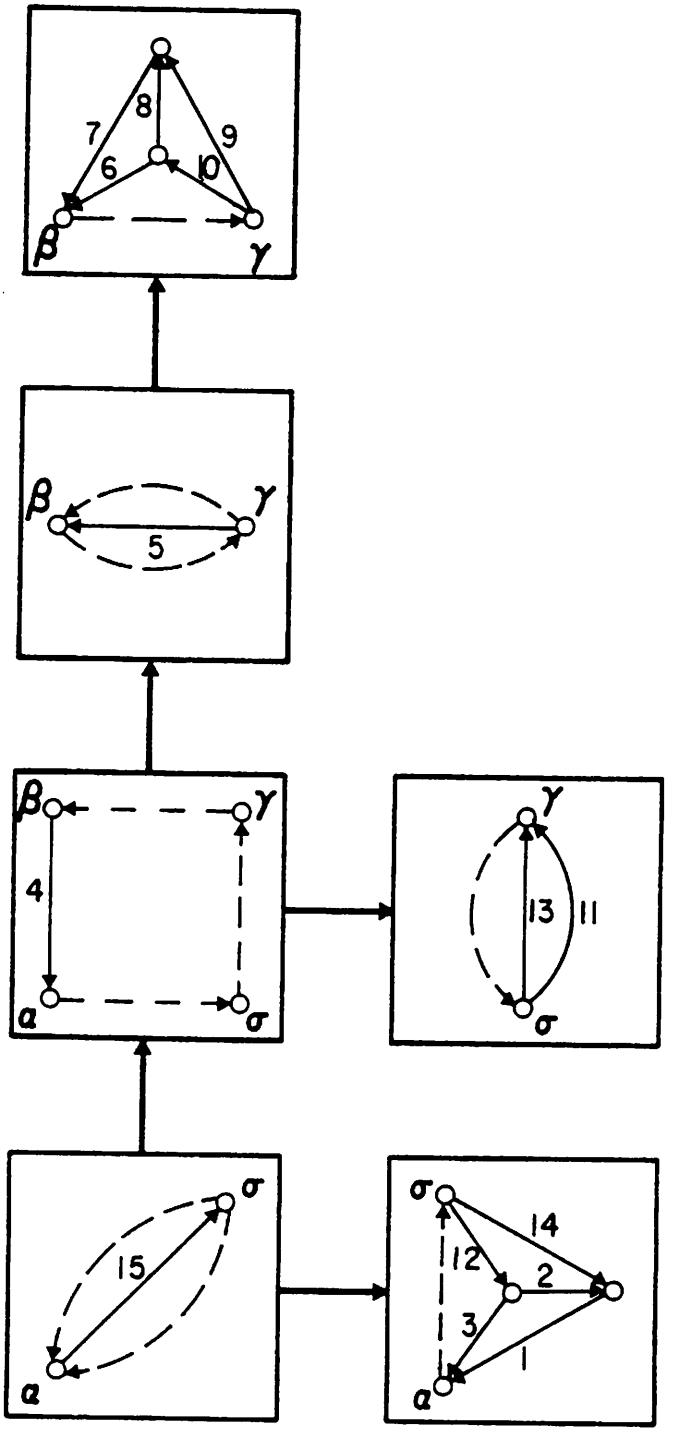


Figure 8

number of arcs in the (unique) path joining it to the root. Now we can define a direction for each arc, by calling the endpoint of the arc nearer to the root the tail and the endpoint farther from the root the head. The following terminology will apply to this directed tree. If v and w are nodes of the tree, such that there is an arc directed from v to w, then v is the father of w, and w is the child of v. Each node except the root has exactly one father. A node which has no children will be called a leaf. Figure 8 shows the decomposition tree with directed arcs for our example.

In the next section, we will show how this decomposition can be used to simplify reliability computations. It should be noted that Hopcroft and Tarjan [1973] have an algorithm which finds these triconnected components in time proportional to the total number of vertices and edges in the graph. Appropriate background for this material can be found in Tarjan [1972] or Aho, Hopcroft, and Ullman [1974].

4. Using the Decomposition

In this section we will describe how to use the decomposition tree to generate the probability distribution of the maximum flow possible to our chosen node when the flow is feasible. This approach is usable with any method of finding the probability distribution of the maximum flow possible through some chosen edge in a triconnected graph, but the problem remains of finding a good method for such a graph. After discussing the method in its generality, we

will discuss its use in conjunction with the method of Doulliez and Jamoulle applied to each triconnected component.

We have introduced directions on the edges of the graph so that we will be able to keep track of signs on flows and their constraints. We will need directions on the virtual edges of the triconnected components as well. In the discussion that follows, we will not attempt to distinguish between a node of the decomposition tree and the triconnected component that corresponds to it. A father and son pair in the decomposition tree will have virtual edges that were generated by the same split. Let us give these edges the same name, say i. In the son, we will often refer to i as the return edge of that component. We will also refer to the special edge in the root as the return edge of the root. To determine how to direct i, we look at it in the father. If i is incident with σ, direct i away from σ. Otherwise, assign it an arbitrary direction. In the son, we always give i the opposite direction.

Let us examine what we have now. We have a directed decomposition tree, in which our special edge appears in the root, and every other edge of our original graph appears in exactly one node. A father-child pair of nodes shares a virtual edge. Each node of the decomposition tree corresponds to a triconnected graph, a bond, or a polygon. Furthermore, every edge of one of these triconnected components has a direction. Whenever σ occurs in a tricon-

nected component, it is the head of the return edge.

The following algorithm may be used to process the decomposition tree. In essence, we will be "plucking" off the leaves, until we have nothing left but the root.

1) Choose a leaf ℓ of the tree. Let r be the return edge in ℓ. Find the probability distribution of $\hat{c}_r$, the maximum flow feasible through r, using the probability distributions of the constraint coefficients on the rest of the edges in ℓ.

2) If ℓ is the root, STOP; we are done. Otherwise, change r to a real edge in the father of ℓ, let $\hat{c}_r$ be its constraint, and erase ℓ from the tree. Go to step 1).

As was pointed out, the probability distribution computed could either be in the form of the probability that "there is a feasible flow in the rest of the graph and there is a maximum flow $\leq$ k in r", or the probability that "all demands in the rest of the graph have been met as far as possible and there is a maximum flow $\leq$ k in r."

The remaining problem is a serious one. How do we compute the probability distribution in the triconnected components? The computation is simple for a bond or a polygon, as we will show. In general, the computation will be laborious for a triconnected graph.

If the triconnected component being processed is a bond, the real edges are in parallel. The maximum flow

through the return edge will be the sum of the $\hat{c}_i$'s for the real edges i of the component. If only one realization of the $\hat{c}_i$'s yields a given level of maximum flow through the return edge, the probability of that level will just be the product of the probabilities associated with the c_i's.

If the triconnected component being processed is a polygon, the real edges will be in series. The maximum flow through the return edge will be the minimum of the $\hat{c}_i$'s over all real edges i in the triconnected component. The probability calculation is again straightforward.

The following proposed method for dealing with triconnected graphs will be sketched only briefly as it is for all intents and purposes equivalent to the methods described by Shogan [1978] and Doulliez and Jamoulle [1972]. To keep the exposition simple, we will ignore a subtlety introduced by Doulliez and Jamoulle which may be faster and allows finding upper and lower bounds when exact computation is prohibitively expensive.

We are given now a triconnected graph in which one edge is called the return edge r and all other edges i have a random constraint $\hat{c}_i$. If σ is in the graph, r is directed into σ and every other edge incident with σ is directed out of it. For every one of these latter edges, $\hat{c}_i$ is an upper bound on the flow and there is no lower bound. For all other edges i, $\hat{c}_i$ is an upper bound on the flow and $-\hat{c}_i$ is a lower bound.

In general, we will fix a value V of flow we want to pass through r. We will compute the probability that a flow greater than or equal to V is feasible in the graph. By performing this for all possible V's, we can compute the probability distribution for $\hat{c}_r$.

The algorithm works as follows. Let Ω be the event that V is feasible. Using as constraints the maximum values assumed by the $\hat{c}_i$'s, find a feasible flow of value V. Let A be the event that $\hat{c}_i \geq f_i$ for this assignment of flows. Then

$$\text{Prob}(\Omega) = \text{Prob}(A)+\text{Prob}(\Omega \text{ and not } A).$$

The next step is to find a subevent B to partition (Ω and not A), say

$$B = \Omega \text{ and } \{\hat{c}_1 < f_1\}$$

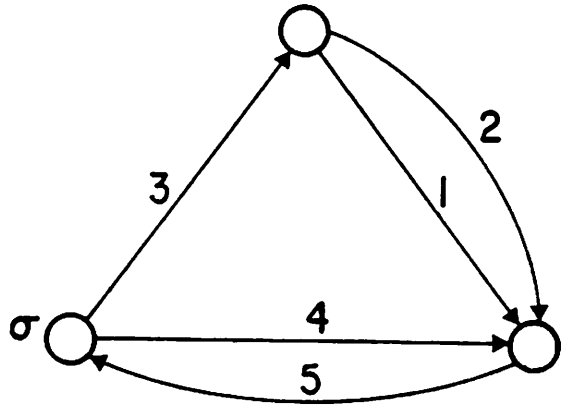
and calculate the probability of B and (Ω and not (A or B)) similarly to our attempt to calculate the probability of Ω.

Fortunately, a whole new maximum flow calculation does not have to be performed. To calculate the probability of B, we may take the solution and last augmentation graph from the Ford-Fulkerson determination of A, and search for augmenting cycles that include edge 1 (in reverse).

Having calculated the probability of Ω for a specific value of V, when we wish to proceed to V-1, again we do not have to start from scratch, but can adjust the solution and augmentation graph from the last calculation made for V.

Essentially, this algorithm just partitions the state vectors for the edges of the graph into sets that are known to be feasible for a given V. There is no reason to assume that we would not have to enumerate nearly all the states, so that we cannot expect this to be a particularly fast algorithm. However, if it is used in the recursive fashion suggested by the decomposition tree, the number of arithmetic operations will be reduced considerably whenever two states of the edges of a triconnected graph yield the same flow V. In the appendix, the number of operations are compared for the simple problem of Figure 9, with and without decomposition.

Thus decomposition into triconnected components will reduce the number of Doulliez and Jamoulle type computations required to find the reliability of a flow network. In applying any other algorithm to flow networks or communication networks, the possibility that decomposition will reduce computation time should be considered.



c_1=1 with probability p_1,
ϕ with probability $1-p_1$
c_2=1 with probability p_2,
ϕ with probability $1-p_2$
c_3=2 with probability p_3,
ϕ with probability $1-p_3$
c_4=ϕ with probability p_4,
-1 with probability $1-p_4$

Decomposition Tree

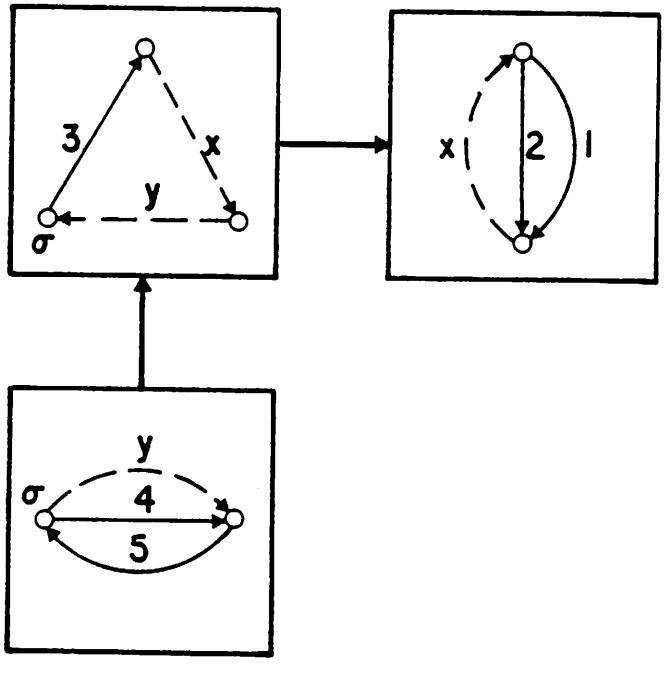


Figure 9

Appendix

Doulliez & Jamoulle-type Algorithm Applied to Original Graph of Figure 9

a) Prob$\{c_5 \geq 2\}$:

$c_1 \geq 1$	$c_2 \geq 1$	$c_3 \geq 2$	$c_4 \geq 0$	$p_1 \cdot p_2 \cdot p_3 \cdot p_4$

3 multiplications

b) Prob$\{c_5 \geq 1\}$:

$c_1 \geq 1$	$c_2 \geq 0$	$c_3 \geq 1$	$c_4 \geq 0$	$p_1 \cdot 1 \cdot p_3 \cdot p_4$
$c_1 = 0$	$c_2 \geq 1$	$c_3 \geq 1$	$c_4 \geq 0$	$(1-p_1) \cdot p_2 \cdot p_3 \cdot p_4$
$c_1 = 1$	$c_4 = -1$	$c_2 \geq 1$	$c_3 \geq 1$	$p_1 \cdot p_2 \cdot p_3 \cdot (1-p_4)$

9 multiplications

2 additions

c) Prob$\{c_5 \geq 0\}$

$c_1 \geq 0$	$c_2 \geq 0$	$c_3 \geq 0$	$c_4 \geq 0$	$1 \cdot 1 \cdot 1 \cdot p_4$
$c_4 = -1$	$c_1 \geq 1$	$c_2 \geq 0$	$c_3 \geq 1$	$(1-p_4) \cdot [p_1 \cdot 1 \cdot p_3$
	$c_1 = 0$	$c_2 \geq 1$	$c_3 \geq 1$	$+ (1-p_1) \cdot p_2 \cdot p_3]$

8 multiplications

2 additions

d) Prob$\{c_5 \geq -1\}$

$c_1 \geq 0$	$c_2 \geq 0$	$c_3 \geq 0$	$c_4 \geq -1$	$1 \cdot 1 \cdot 1 \cdot 1$

3 multiplications

total operations 27

Doulliez and Jamoulle-type Algorithm Applied Recursively to the Triconnected Components of the Graph in Figure 9

1.a) $Prob\{c_x \geq 2\}$

$c_1 \geq 1$ $c_2 \geq 1$ $p_1 \cdot p_2$

1 multiplication

b) $Prob\{c_x \geq 1\}$

$c_1 \geq 1$ $c_2 \geq 0$ $p_1 \cdot 1$

$c_1 = 0$ $c_2 \geq 1$ $(1-p_1) \cdot p_2$

2 multiplications

1 addition

c) $Prob\{c_x \geq 0\}$

$c_1 \geq 0$ $c_2 \geq 0$ $1 \cdot 1$

1 multiplication

2.a) $Prob\{c_y \geq 2\}$

$c_x \geq 2$ $c_3 \geq 2$ $p_3 \cdot Prob\{c_x \geq 2\}$

1 multiplication

b) $Prob\{c_y \geq 1\}$

$c_x \geq 1$ $c_3 \geq 1$ $p_3 \cdot Prob\{c_x \geq 1\}$

1 multiplication

c) $Prob\{c_y \geq 0\}$

$c_x \geq 0$ $c_3 \geq 0$ $1 \cdot 1$

1 multiplication

3.a) $Prob\{c_5 \geq 2\}$

$c_y \geq 2$ $c_4 \geq 0$ $p_4 \cdot Prob\{c_y \geq 2\}$

1 multiplication

b) $Prob\{c_5 \geq 1\}$

$c_y \geq 1$ $c_4 \geq 0$ $p_4 \cdot Prob\{c_y \geq 1\}$

$c_4=-1 \quad c_y \geq 2$	$(1-p_4) \cdot \text{Prob}\{c_y \geq 2\}$
	2 multiplications
	1 addition

c) $\text{Prob}\{c_5 \geq 0\}$

$c_y \geq 0 \quad c_4 \geq 0$	$p_4 \cdot 1$
$c_4=-1 \quad c_y \geq 1$	$(1-p_4) \cdot \text{Prob}\{c_y \geq 1\}$
	2 multiplications
	1 addition

d) $\text{Prob}\{c_5 \geq -1\}$

$c_2 \geq 0 \quad c_4 \geq -1$	$1 \cdot 1$
	1 multiplication

total operations	16

References

A.V. Aho, J.E. Hopcroft, J.D. Ullman

[1974] The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, Mass. 176-195.

W.H. Cunningham & J. Edmonds

[1979] A combinatorial decomposition theory. Canad. J. of Math., to appear, 57pp.

P. Doulliez & E. Jamoulle

[1972] Transportation networks with random arc capacities. RAIRO 3: 45-60.

L.R. Ford & D.R. Fulkerson

[1960] Flows in Networks. Princeton University Press, Princeton, N.J., 11-23.

David Gale

[1960] The Theory of Linear Economic Models. McGraw-Hill, New York, 148-155.

J.E. Hopcroft & R.E. Tarjan

[1973] Dividing a graph into triconnected components. SIAM J. Comput. 2: 135-158.

A.W. Shogan

[1978] Modular decomposition in stochastic transportation networks. Memorandum No. UCB/ERL M78/86, Electronics Research Laboratory, University of California, Berkeley, Ca., 29pp.

R. Tarjan

[1972] Depth-first search and linear graph algorithms. SIAM J. Comput. 1: 146-160.

W. Tutte

[1966] Connectivity in Graphs. University of Toronto Press, Toronto.

R. Willie

[1979] Computer-oriented Methods for Assessing the Reliability of Complex Systems. PhD. Thesis, University of California, Berkeley.

A DERIVATION OF THE TIME TO FAILURE DISTRIBUTION IN CONTINUOUS-TIME MARKOV CHAINS

Natarajan Narasimhamurthi

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California 94720

ABSTRACT

In this paper the expression for the 'time to failure distribution' for systems modelled as continuous-time finite state Markov Chains is derived using only elementary concepts of probability. This is used to obtain the expressions for expected time to failure and expected cycle time of the system. It is shown that under steady state assumptions the system can be modeled as if it were a two state Markov Chain for the computations of commonly used reliability indices.

Research sponsored by the Department of Energy Grant EC-77-S-01-5105.

1. Introduction

We consider a finite-state continuous-time Markov Chain. The state space is partitioned into two sets: U, the set of 'up-states' and D, the set of 'down-states.' Suppose that initially the system is in U with a given probability distribution. In this paper we derive the expression for the distribution of the time at which the system leaves U. Based on this general expression for the time to failure distribution, we present a simple derivation of the expected cycle time.

This problem arises in reliability studies [1]. Consider, for example, a complex repairable system with many independent components where each component has an exponentially distribured failure and rapair time. The system can then be modeled as a 2^n-state continuous-time Markov Chain, where n is the number of components in the system. Of these states, some correspond to the system being up (working) while the others correspond to system failure. In such models it is of great interest to find 'time to failure' distribution. Such models have been applied to large-scale power system studies [2] in the so called frequency and duration method.

Brown [3] has derived the time to failure distribution for a parallel system (system for which the set D consists of only one state, viz, where all components have failed). Kielson [4] has considered the structure of various failure time distributions of a general system and their inter-relationships. Ross [5] and Barlow and Proschan [6] have considered the case where initially all the components are up. They have derived some important properties of the time to failure distributions.

The derivation presented here makes use of only elementary concepts

of probability. We first note that the time to failure distribution function satisfies a linear first order differential equation with time varying coefficient. This coefficient is seen to depend on the vector of conditional probabilities. This vector satisfies a first order non-linear differential equation. We obtain the solution of this differential equation to derive the expression for the mean time to failure distribution. This expression is used to obtain the expression for expected cycle time and the expected time to failure of the system.

2. Preliminaries

To make the analysis manageable let us number the states 1 to N with the states in U numbered 1 to M. Let $\underline{x}(t)$ denote the state at time t. The assumptions inherent in the system model are given by:

Basic Assumptions.

1. For each time t, for each i,j such that $i \neq j$ there exists a $\lambda_{i,j}$ such that

$$\text{Prob}\{\underline{x}(t+\Delta t)=j \,|\, \underline{x}(t)=i\} = \lambda_{i,j}\Delta t + o(\Delta t);\ \Delta t \geq 0 \qquad (1)$$

2. The Markovian assumption that:

$$\text{Prob}\{\underline{x}(t_1)=j \,|\, \underline{x}(t)=i;S\} = \text{Prob}\{\underline{x}(t_1)=j \,|\, \underline{x}(t)=i\}$$

where $t_1 > t$ and S is any condition prior to time t.

3. $\lambda_{i,j}$ is independent of time t.

4. $\text{Prob}\{\underline{x}(t+\Delta t)=i;\underline{x}(t+\theta\Delta t)=j \text{ for some } \theta \in (0,1) \,|\, \underline{x}(t)=k\} = o(\Delta t)$ whenever $i \neq j$ and $j \neq k$.

In view of the above assumptions the system is time-invariant and without loss of generality let us assume that the initial time of interest is 0

(the result that we obtain can be easily translated to any other starting time).

Problem Formulation.

We are given that at time 0 the system is in the set of up-states and we are also given the initial probability distribution $\underline{p}_o$ where the i-th component of $\underline{p}_o$ is the Prob$\{\underline{x}(0)=i\}$. Let T represent the time at which the system leaves the set U. We are interested in obtaining the distribution of T viz.

$$F(t) = \text{Prob}\{T>t \,|\, \underline{x}(0) \in U; \underline{p}_o\}$$

$$= \text{Prob}\{\underline{x}([0,t]) \subset U \,|\, \underline{x}(0) \in U; \underline{p}\}$$

In order to obtain F(t) we shall obtain the differential equation governing the function F with $\Delta t > 0$

$$F(t+\Delta t) = \text{Prob}\{\underline{x}([0,t+\Delta t]) \subset U \,|\, \underline{x}(0) \in U; \underline{p}_o\}$$

$$= \text{Prob}\{\underline{x}([t,t+\Delta t]) \subset U \,|\, \underline{x}([0,t]) \subset U; \underline{p}_o\}$$

$$\text{Prob}\{\underline{x}([0,t]) \subset U \,|\, \underline{x}(0) \in U; \underline{p}_o\}$$

$$= (1-\text{Prob}\{\underline{x}(t') \text{ is not in U for some } t' \text{ in}$$

$$[t,t+\Delta t] \,|\, \underline{x}([0,t]) \subset U; \underline{p}_o\})\ F(t)$$

$$= [1-(\lambda(t)\ \Delta t + o(\Delta t))]F(t)$$

$$\text{where } \lambda(t) = \lim_{\substack{\Delta t \geq 0 \\ \Delta t \geq 0}} \frac{\text{Prob}\{\underline{x}(t+\Delta t) \in D \,|\, \underline{x}([0,t]) \subset U; \underline{p}_o\}}{\Delta t}$$

From this it follows that

$$\frac{dF}{dt}(t) = -\lambda(t)F(t) \tag{2}$$

Remarks. We can regard $\lambda(t)$ as the rate of departure from the set U. This would depend on the state of the system at time t. Thus in order to obtain $\lambda(t)$ we require the probability distribution of the state at time t. This as we shall see presently depends on t and satisfies a nonlinear differential equation. We are however given the initial probability distribution viz. $\underline{p}_o$ and the initial value problem has a closed form solution. Using this solution we obtain the expression for F(t).

3. Expression for F(t)

Notation: When we use $t_0, t_1, \ldots$, it is assumed that $t_0 < t_1, \ldots,$.

Definitions.

$$\lambda_{i,i} = -\sum_{\substack{j=1 \\ j\neq i}}^{N} \lambda_{i,j} \text{ and } \Lambda = \{\lambda_{i,j}\}_{i,j=1}^{N} = \begin{bmatrix} \Lambda_{UU} & \Lambda_{UD} \\ \Lambda_{DU} & \Lambda_{DD} \end{bmatrix}$$

$\underline{p}(t)$ a row vector of dimension M such that

$$\underline{p}^i(t) = \text{Prob}\{\underline{x}(t)=i \,|\, \underline{x}([0,t]) \subset U; \underline{p}_o\},$$

$\underline{s}(t)$ a row vector of dimension N such that

$$\underline{s}^i(t_2,t_1) = \text{Prob } \underline{x}(t_2)=i \,|\, \underline{x}([0,t_1]) \subset U; \underline{p}_o\};$$

clearly $\underline{s}^i(t,t) = \underline{p}^i(t) \quad i = 1,2,\ldots,M.$

Also we define $\underline{1}$ as a column vector of proper dimension with each

component equal to 1. This vector is useful for summation of components of row vectors.

3.1. The Differential Equation for $p(t)$

$$\underline{s}^i(t+\Delta t,t) = \sum_{j=1}^{M} \text{Prob}\{\underline{x}(t+\Delta t)=i \,|\, \underline{x}(t)=j\}$$

$$\text{Prob}\{\underline{x}(t)=j \,|\, \underline{x}([0,t]) \subset U;\underline{p}_o\}$$

$$= \sum_{\substack{j=1 \\ j\neq i}}^{M} \underline{s}^j(t,t)(\lambda_{j,i}\Delta t+o(\Delta t)) + \underline{s}^i(t,t)(1+\lambda_{i,i}\Delta t+o(\Delta t)) \qquad (3)$$

Now

$$\text{Prob}\{\underline{x}(t+\Delta t) \in U \,|\, \underline{x}([0,t]) \subset U;\underline{p}_o\} = \sum_{i=1}^{M} \underline{s}^i(t+\Delta t,t)$$

$$= \sum_{i=1}^{M} \{\underline{s}^i(t,t) + \sum_{j=1}^{M} \underline{s}^j(t,t)\{\lambda_{j,i}\Delta t+o(\Delta t)\}\}$$

$$= 1 + \sum_{i=1}^{M} \sum_{j=1}^{M} \underline{s}^j(t,t)\{\lambda_{j,i}\Delta t+o(\Delta t)\}$$

$$= 1 + \underline{p}(t)\Lambda_{UU}\underline{1}\Delta t+o(\Delta t)$$

Hence for $i = 1,2,\ldots M$

$$\underline{p}^i(t+\Delta t) = \frac{\underline{s}^i(t+\Delta t,t)}{1+\underline{p}(t)\Lambda_{UU}\underline{1}\ t+o(\Delta t)}$$

Using (3) we get

$$\underline{p}(t+\Delta t) = \frac{\underline{p}(t)+\underline{p}(t)\Lambda_{UU}\Delta t+o(\Delta t)}{1+\underline{p}(t)\Lambda_{UU}\underline{1}\Delta t+o(\Delta t)}$$

and

$$\underline{p}(t+\Delta t)-\underline{p}(t) = \frac{\underline{p}(t)\Lambda_{UU}\Delta t-\underline{p}(t)\Lambda_{UU}\underline{1}\underline{p}(t)\Delta t+o(\Delta t)}{1+\underline{p}(t)\Lambda_{UU}\underline{1}\Delta t+o(\Delta t)}$$

where

$$\frac{d\underline{p}}{dt} = \underline{p}\Lambda_{UU} - \underline{p}\Lambda_{UU}\underline{1}\underline{p}; \qquad \underline{p}(0) = \underline{p}_o \tag{4}$$

3.2. The Expression for F(t)

Also

$$\lambda(t) = \lim_{\substack{\Delta t->0 \\ \Delta t\geq 0}} \frac{\text{prob}\{\underline{x}(t+\Delta t) \in D \,|\, \underline{x}([0,t]) \subset U; \underline{p}_o\}}{\Delta t}$$

$$= \lim_{\substack{\Delta t->0 \\ t\geq 0}} \frac{\sum_{j=M+1}^{N} \text{prob}\{\underline{x}(t+\Delta t)=j \,|\, \underline{x}([0,t]) \subset U;\ \underline{p}_o\}}{\Delta t}$$

$$= \lim_{\substack{\Delta t->0 \\ \Delta t\geq 0}} \sum_{j=M+1}^{N} \sum_{i=1}^{M} \left[\frac{\text{Prob}\{\underline{x}(t+\Delta t)=j \,|\, \underline{x}(t)=i\}}{\Delta t}\right.$$

$$\left.\text{Prob}\{\underline{x}(t)=i \,|\, \underline{x}([0,t]) \subset U; \underline{p}_o\}\right]$$

$$= \lim_{\substack{\Delta t->0 \\ t\geq 0}} \frac{\sum_{j=m+1}^{N} \sum_{i=i}^{M} p^i(t)\lambda_{i,j}\Delta t+o(\Delta t)}{\Delta t}$$

$$= \sum_{j=m+1}^{N} \sum_{i=1}^{M} p^i(t)\lambda_{i,j} = \sum_{i=i}^{M} \sum_{j=m+1}^{N} p^i(t)\lambda_{i,j}$$

$$= \sum_{i=1}^{M} p^i(t) \sum_{j=m+1}^{N} \lambda_{i,j} = - \sum_{i=1}^{M} p^i(t) \sum_{j=1}^{M} \lambda_{i,j}$$

$$= - \sum_{i=1}^{M} \sum_{j=1}^{M} p^i(t) \lambda_{i,j}$$

$$= - \underline{p}(t) \Lambda_{UU} \underline{1} \tag{5}$$

It is easily verified that the solution for (4) is given by

$$\underline{p}(t) = \frac{\underline{p}_o e^{\Lambda_{UU} t}}{\underline{p}_o e^{\Lambda_{UU} t} \underline{1}} \tag{6}$$

Substituting (6) in (5) we get

$$\lambda(t) = - \frac{\underline{p}_o e^{\Lambda_{UU} t} \Lambda_{UU} \underline{1}}{\underline{p}_o e^{\Lambda_{UU} t} \underline{1}} \tag{7}$$

Substituting (7) in (2) and noting that the numerator in (7) is the defivative of the denominator we get

$$F(t) = \underline{p}_o e^{\Lambda_{UU} t} \underline{1} \tag{8}$$

Remarks. Suppose that the following holds:

Sum of the elements in each row of Λ_{UD} is the same (say λ_{out}). (9)

Then $\Lambda_{UU} \underline{1} = - \lambda_{out} \underline{1} \Rightarrow e^{\Lambda_{UU} t} = \underline{1} e^{-\lambda_{out} t}$. This with the initial condition that $\underline{p}_o \underline{1} = 1$ yields

$$F(t) = e^{-\lambda_{out} t} \tag{10}$$

Also it easily verified that F(t) is of the form (10) for any $\underline{p}_o$ (with $\underline{p}_o \underline{1} = 1$) if and only if (9) holds.

Professor R. E. Barlow, after seeing the priliminary version of this report, has suggested an alternative derivation based on the theory of Markov Chains. His derivationis based on considering a new Markov Chain with M+2 states with transition rate matrix Λ^* given by

$$\lambda^*_{i,j} = \lambda_{i,j} \text{ for } 1 \leq i, j \leq M$$

$$= -\sum_{k=1}^{M} \lambda_{i,k} \text{ for } 1 \leq i \leq M \text{ and } j = M+1$$

$$= -\lambda \quad \text{for } i = j \text{ and } i > M$$

$$= \lambda \quad \text{for } i \neq j \text{ and } i,j > M$$

$$= 0 \quad \text{for all other cases}$$

With this {M+1, M+2} becomes an absorbing set and F(t) is the same as the probability of finding the system in one of the first M states at time t. Using the standard expression for this probability one obtains Eq. (8).

4. Application to Reliability Evaluation

4.1. Preliminaries

In the previous section we derived an expression for the time spent in a group of states, given the initial probability vector. In this

section we shall apply the result to obtain the distribution of certain important random variables used in reliability analysis. Since the expression (8) requires that the initial probability vector be given we proceed under the assumption that the system is in steady state. This idea will be made precise after we define the random variables of interest.

Definition

We say that the time spent in the down (up) states at time t_o is T_D (T_U) if the following event takes place

$$\underline{x}(0^-) \in U;\ \underline{x}([t_o, t_o+T_d)) \subset D;\ \underline{x}(t_o+T_d) \in U.$$

Also we define the cycle time T_c at t_o as follows

$$\underline{x}(t_o^-) \in D;\ \underline{x}([t_o, t_o+t_1)) \subset U;\ \underline{x}(t_o+t_1, t_o+t_c)) \subset D;\ \underline{x}(t_p+t_c) \in U.$$

with $0 < t_1 < t_c$.

Also it is useful to define another random variable $y = t_c - t_1$. Throughout this analysis we take $t_o = 0$.

Underlying Assumptions

1. $\forall t\ \pi^i(t) = \text{Prob}\{x(t)=i\}$ exists.

Lemma: $\pi(t) = (\pi^1(t), \pi^2(t) \ldots \pi^N(t))$ is a constant and satisfies $\pi(t)\Lambda = 0$.

Proof: Using (5) we have

$$\underline{\pi}(t) = \frac{\underline{\pi}(t_o)e^{\Lambda(t-t_o)}}{\underline{\pi}(t_o)e^{\Lambda(t-t_o)}\underline{1}}$$

$$= \underline{\pi}(t_o)e^{\Lambda(t-t_o)}$$

since $\Lambda\underline{1} = 0$, $\underline{\pi}(t_o)\underline{1} = 1$

Hence $\underline{\pi}(\cdot)$ is bounded

$\Leftrightarrow \underline{\pi}(\cdot)$ is a constant

$\Leftrightarrow \underline{\pi}\Lambda = 0$

2. In view of the above lemma let $\underline{\pi} = \underline{\pi}(t)$.

We assume that $\forall i\ \underline{\pi}^i > 0$

4.2. Distributions

Statement of the problem:

Given the above two assumptions we require the following distributions

1. $F_U(t) = \text{Prob}\{T_U > t\}$ and similarly F_D
2. $F_c(t) = \text{Prob}\{T_c > t\}$

Since $T_c = T_U + y$ we have an equivalent formulation of 2 as

2'. $F_y(t,x) = \text{Prob}\{T_y > t | T_U = x\}$

Remark. If we have $F_y(t,x)$ then clearly

$$F_c(t) = -\int F_y(t-x,x)dF_U(x)$$

In order to use (8) we require the initial probability vector $\underline{p}_o$. Since the following problem comes up often in our work we solve it first.

Problem. Given a set of states A, given that at 0^-, $x(0^-) \in \overline{A}$, the complement of A, given $\underline{p}(0^-)$ the probability vector at time 0^-, find

$\underline{p}(0)$ the probability vector at time 0, given that $x(0) \in A$.

Clearly

$$\text{Prob}\{\underline{x}(0)=i \mid \underline{x}(-\delta)=j\} = \lambda_{j,i}\delta + o(\delta)$$

hence

$$\text{Prob}\{\underline{x}(0)=i\} = \sum_j \underline{p}_j(-\delta)\lambda_{j,i}\delta + o(\delta)$$

$$\Rightarrow \text{Prob}\{\underline{x}(0)=i \mid \underline{x}(0) \in A\}$$

$$= \frac{\sum_j \underline{p}_j(-\delta)\lambda_{j,i}\delta + o(\delta)}{\sum_{i\in A}\sum_j \underline{p}_j(-\delta)\lambda_{j,i}\delta + o(\delta)}$$

$$= \frac{\sum_j \underline{p}_j(0^-)\lambda_{j,i}}{\sum_{i\in A}\sum_j \underline{p}_j(0^-)\lambda_{j,i}}$$

Hence $\underline{p}_A(0)$ is a multiple of $\underline{p}_{\bar{A}}(0^-)\Lambda_{A\bar{A}}$

Let $\underline{p}_U(0)$ be a row vector such that

$$\underline{p}_u^i(0) = \text{Prob}\{\underline{x}(0)=i \mid \underline{x}(0^-) \in D; \underline{x}(0) \in U\}$$

Also let $\underline{p}_D(t)$ be a row vector such that

$$\underline{p}_D^i = \text{Prob}\{\underline{x}(t)=i+M \mid \underline{x}(t) \in D; \underline{x}[0,t) \subset U; \underline{x}(0^-) \in D\}$$

In view of the result just stated we have

$$\underline{p}_U(0) \text{ is a multiple of } \underline{\pi}_D\Lambda_{DU} = -\underline{\pi}_U\Lambda_{UU}$$

and

$\underline{p}_D(t)$ is a multiple of $\underline{p}_U(0)e^{\Lambda t}\Lambda_{UD}$

Direct application of (8) yields

$$F_U(t) = \underline{p}_U(0)e^{\Lambda_{UU}t}\underline{1}$$

$$= \frac{\underline{\pi}_U\Lambda_{UU}e^{\Lambda_{UU}t}\underline{1}}{\underline{\pi}_U\Lambda_{UU}\underline{1}}$$

$$= \frac{1}{\underline{\pi}_U\Lambda_{UU}\underline{1}} \frac{d}{dt}\underline{\pi}_U e^{\Lambda_{UU}t}\underline{1}$$

and

$$F_y(t,x) = \underline{p}_D(x)e^{\Lambda_{DD}t}\underline{1}$$

$$= \frac{\underline{\pi}_U\Lambda_{UU}e^{\Lambda_{UU}x}\Lambda_{UD}e^{\Lambda_{DD}t}\underline{1}}{\underline{\pi}_U\Lambda_{UU}e^{\Lambda_{UU}x}\Lambda_{UD}\underline{1}}$$

where

$$F_c(t) = -\int F_y(t-x,x)dF_U(x)$$

$$= -\int_0^t F_y(t-x,x)dF_U(x) - \int_t^\infty dF_U(x)$$

$$= F_U(t) - \int_0^t F_y(t-x,x)dF_U(x)$$

$$= F_U(t) + \int_0^t \frac{\underline{\pi}_U\Lambda_{UU}e^{\Lambda_{UU}x}\Lambda_{UD}e^{\Lambda_{DD}(t-x)}\underline{1}dx}{\underline{\pi}_U\Lambda_{UU}\underline{1}}$$

where we have made use of the fact that

$$\frac{de^{\Lambda_{UU}x}\underline{1}}{dx} = e^{\Lambda_{UU}x}\Lambda_{UU}\underline{1}$$

$$= -e^{\Lambda_{UU}x}\Lambda_{UD}\underline{1}$$

Remark: The integral is a standard convolution integral of the form $\int_0^t H(x)G(t-x)dx$. While a closed form for the integral is not possible we have the following easily verifiable result.

$$\int_0^\infty \int_0^t H(x)G(t-x)dxdt = \int_0^\infty H(x)dx \int_0^\infty G(t)dt$$

4.3. Expectations

While the distributions we obtained in Section 4.2 are of importance, in reliability studies one is more interested in the expected values of these random variables. These values are sometimes used for reliability indices. In this section we shall compute these reliability indices. While these computations are straightforward, the results we obtain can be given extremely simple physical interpretations.

1. $E(T_U)$ = Expected time spent in up-states

$$= \int_0^\infty F_u(t)dt$$

$$= \frac{1}{\underline{\pi}_U\Lambda_{UU}\underline{1}} - \underline{\pi}_U\underline{1} = \frac{\underline{\pi}_U\underline{1}}{\underline{\pi}_U\Lambda_{UD}\underline{1}}$$

Similarly

2. $E(T_D)$ = Expected time spent in down states

$$= \frac{\underline{\pi}_D \underline{1}}{\underline{\pi}_D \Lambda_{DU} \underline{1}}$$

Remark: The vector $\underline{\pi}_{UD}\underline{1}$ represents the vector of rates out of U, i.e.

$$(\underline{\pi}_{UD}\underline{1})^i = \lim_{\substack{\Delta t>0 \\ \Delta t->0}} \frac{\text{prob}\{\underline{x}(\Delta t) \in D | \underline{x}(0)=i\}}{\Delta t} = \lambda_{i,D}$$

and $\frac{\underline{\pi}_U}{\underline{\pi}_U \underline{1}}$ represents the vector of conditional steady state probability, given that the system is up. If we define the steady state failure rate of the system (λ) as

$$\lambda = \sum_{i \in U} \text{Prob}\{\underline{x}=i | x \in U\}\lambda_{i,D} = \lim_{\substack{\Delta t \to 0 \\ \Delta t>0}} \frac{\text{Prob}\{\underline{x}(\Delta t) \in D | \underline{x}(0) \in U\}}{\Delta t}$$

and similarly steady state repair rate of the system (μ) we have the following:

$$E(T_U) = \frac{1}{\mu} \quad \text{and} \quad E(T_D) = \frac{1}{\mu}$$

Also noting that $\underline{\pi}_D \Lambda_{DD} \underline{1} = \underline{\pi}_U \Lambda_{UU} \underline{1}$ we have

$$\text{Prob}\{x \in U\} = \frac{\underline{\pi}_U \underline{1}}{\underline{\pi}_U \underline{1} + \underline{\pi}_D \underline{1}} = \frac{\mu}{\lambda+\mu} \text{ and } \text{Prob}\{x \in D\} = \frac{\lambda}{\lambda+\mu}$$

3. $E(T_c)$ = Expected cycle time

$$= \int_0^\infty F_c(t)dt$$

$$= E(T_U) + \frac{\underline{\pi}_U \Lambda_{UD} \Lambda_{DD}^{-1} \underline{1}}{\underline{\pi}_U \Lambda_{UU} \underline{1}}$$

where we have made use of the fact following the remark on convolution integrals. Making use of the fact that

$$\underline{\pi}_U \Lambda_{UD} = -\underline{\pi}_D \Lambda_{DD} \quad \text{and} \quad \underline{\pi}_U \Lambda_{UU} = -\underline{\pi}_D \lambda_{DU}$$

we obtain

$$E(T_c) = \frac{1}{\lambda} + \frac{1}{\mu}$$

5. Conclusion

We collect the results in the previous section as the following two key observations.

1. Even under steady state assumptions the transitions between two groups of states in a time finite-state Markov chain' will not be a 'two-state Markov chain' and hence the residence time in any group of states will in general not be exponentially distributed.

2. However, for the calculations of the expected time spent in the up-states, the expected time spent in the set of down-states, the expected cycle time (or the mean time between failures), the probability of finding the system up or down in a system modelled as 'continuous-time finite-state Markov chain' we can model the system as if it were a 'two state Markov chain' the two states being the 'up-state' and the'down-state.'

6. Acknowledgements

I would like to thank Ms. Claudia Grief for all the discussions we had, and for her thought provoking counter examples to the various false starts I made prior to this work. Also I wish to thank Professor R. E. Barlow for his comments on this work.

References

[1] R. E. Barlow and F. Prochan, Statistical Theory of Reliability and Life Testing, Holt, Reinhart, and Winston, 1975.

[2] R. Billinton, R. J. Ringlee and A. J. Wood, Power System Reliability Calculations, The MIT Press, 1973.

[3] M. Brown, "The first passage time distribution for a parallel exponential system with repair," Reliability and Fault Tree Analysis, R. E. Barlow et al. (eds.), SIAM, 1975, pp. 365-398.

[4] J. Keilson, "Systems of independent Markov components and their transient behavior," Reliability and Fault Tree Analysis, R. E. Barlow et al. (eds.), SIAM, 1975, pp. 351-364.

[5] S. M. Ross, "On time to first failure in multicomponent expoenetial reliability systems," ORC Report 74-8, Operations Research Center, Univ. of California, Berkeley, 1974.

[6] R. E. Barlow and F. Prochan, "Theory of maintained systems: distribution of time to first system failure," Math. of Op. Res., vol. 1, 1976, pp. 32-42.

FOUNDATION OF THE
FREQUENCY AND DURATION MODEL
FOR POWER SYSTEM RELIABILITY EVALUATION

Claudia Greif

6/17/1978

I. Introduction

Power System Reliability is the study of system performance at various levels based on the individual component availability and the system configuration. In other words,given the life time and the repair time distribution of various components (generators,transformers, transmission lines etc.) the purpose is to find indices which characterize the probabilistic behavior of the system. The models currently used in Power System Reliability studies are:

1. LOLP model which gives the probability that certain demands cannot be satisfied and the expected amount of the demand not served. The time parameter is not included in this model.

2. FAD model which gives the expected frequency of outages of certain magnitudes along with the expected duration of these outages.

The idea of the FAD model seems to have been suggested by De Sieno and Stine [5]. Hall,Ringlee and Wood [6] later gave a recursive formula for computing the expected frequency and duration of outages thus making its application to real large-scale power systems more attractive. The FAD model in power systems assumes that component lifetime and repair time are exponential. The recursive formulas were derived, without proof, using intuitive arguments and generalization from simple two component -

four state systems.

The objective of this report is to provide a mathematical foundation for the FAD model. More general formulas than the ones currently used are derived. The approach in this report follows the paper by Ross [1].

After the introduction, the development of the work in this report is presented in three steps. In III the frequency of encountering any state is derived. In IV a formula for the frequency of system breakdown is obtained. A recursive formula for computing the frequency of breakdowns of a coherent system is derived in V.

II. Preliminaries

1. Renewal Model of Components

Consider a system of n independent components; at any instance of time each component can be up (functioning) or down (failed).

Define $x_i(t) = \begin{cases} 1 \text{ if component } i \text{ is up at time } t \\ 0 \text{ if component } i \text{ is down at time } t \end{cases}$

Let us assume that for every i, $\{x_i(t),\ t \geq 0\}$ is an alternating renewal process, i.e. the state of the i-th component alternates between up and down periods with up and down times independent and identically distributed. Specifically, if we let

$\mathcal{U}_j^i$ = the j-th up time of component i

$\mathcal{D}_j^i$ = the j-th down time of component i

then

$$x_i(t) = \begin{cases} 1 & t \leq \mathcal{U}_1^i \\ 0 & \mathcal{U}_1^i < t \leq \mathcal{U}_1^i + \mathcal{D}_1^i \\ 1 & \mathcal{U}_1^i + \mathcal{D}_1^i < t \leq \mathcal{U}_1^i + \mathcal{D}_1^i + \mathcal{U}_2^i \\ 0 & \text{etc.} \end{cases}$$

and we assume that, for every i, the pairs $(\mathcal{U}_j^i, \mathcal{D}_j^i)$, $j \geq 1$ are independent identically distributed random vectors. Let $F_i(t)$, $G_i(t)$ be the distribution functions of the life time $(\mathcal{U}_j^i)$ and repair time $(\mathcal{D}_j^i)$ of component i, and let $\frac{1}{\lambda_i}$ and $\frac{1}{\mu_i}$ be

their means ($0<\frac{1}{\lambda_i}<\infty, 0<\frac{1}{\mu_i}<\infty$). We assume that $F_i * G_i$ is non-lattice [3, p41].

The steady state probability that the i-th component is on is

$$p_i = \lim_{t\to\infty} P(t) = \frac{\frac{1}{\lambda_i}}{\frac{1}{\lambda_i}+\frac{1}{\mu_i}} = \frac{\mu_i}{\lambda_i+\mu_i} \tag{2.1}$$

and the steady state probability that the i-th component is down is

$$1-p_i = \lim_{t\to\infty} [1 - P(t)] = \frac{\lambda_i}{\lambda_i+\mu_i} \tag{2.2}$$

For a given state J of the form $(x_1,\ldots x_n)$, let

$$u_J=\{k|\ x_k= 1 \text{ in state } J\ \}$$

$$d_J=\{k|\ x_k= 0 \text{ in state } J\ \}$$

then we can write the probability of state J, using independence of components as

$$P\{\text{state } J\ \}=\text{Prob}\{(x_1,\ldots x_n)\}= \prod_{k\in u_J} p_k \prod_{k\in d_J} (1-p_k) \tag{2.3}$$

Define the <u>frequency of encountering the up or down state of component i</u> as

$$f_i = \lim_{k\to\infty} \frac{\text{number of up times of component i in t}}{t}$$

$$= \frac{1}{\frac{1}{\lambda_i}+\frac{1}{\mu_i}} = \frac{\lambda_i \mu_i}{\lambda_i+\mu_i} = \frac{1}{E\ (\text{cycle time})} \qquad (2.4)$$

It is also useful to introduce the following notation:

$$(1_i, \underline{x}) = (x_1, \ldots, x_{i-1}, 1_i, x_{i+1}, \ldots, x_n)$$
$$(0_i, \underline{x}) = (x_1, \ldots, x_{i-1}, 0_i, x_{i+1}, \ldots, x_n)$$

To simplify notation and check the results with previous works based on exponential distribution, let us call the reciprocal of the expected life time (repair time) of the i-th component , the <u>failure rate</u> (<u>repair rate</u>), i.e.

$$\text{failure rate} = \lambda_i = \frac{1}{\text{Expected life time of component } i}$$
$$\text{repair rate} = \mu_i = \frac{1}{\text{Expected repair time of component } i}$$

<u>Remark</u>: In reliability theory [2, p53], <u>the failure rate</u> r(t) of a lifetime distribution F(t) is

$$r(t) = \lim_{x \to 0} \frac{1}{x} \frac{F(t+x)-F(t)}{\overline{F}(t)}$$

This failure rate r(t) is a constant (does not depend on time) if and only if $F(t)=1-e^{-\lambda t}$. In this case,

$$r(t)=\lambda=\frac{1}{\text{Expected lifetime}}$$

For non exponential distributions r(t) is <u>not</u> a constant and $r(t) \neq \frac{1}{\text{Expected lifetime}}$. Therefore for non-exponential distributions, what we call as failure rate is <u>not</u> the same terminology used in reliability theory.

2. Regeneration Points

Even though $\{x_i(t), t \geq 0\}$ $i=1,2,\ldots n$ are independent alternating renewal processes and the system can alternate between functioning and failed states, the process may not have any regeneration points, i.e. there may not be a time at which the process starts all over again. Such regeneration points are only a consequence of particular distribution functions of the up and down times of each component. One distribution which would make the process regenerative is the exponential distribution; there are, however, weaker ways to ensure this property.

Suppose that for each $i=1,2,..n$ either the up or down time is a mixture of an exponential and an arbitrary distribution, i.e. either

$$F_i(t) \text{ is of the form } p_i(1-e^{-\lambda_i t})+(1-p_i)H_i(t)$$

or (2.5)

$$G_i(t) \text{ is of the form } p_i(1-e^{-\mu_i t})+(1-p_i)H_i(t)$$

with $H_i(t)$ arbitrary, with the same mean as the exponential ($\frac{1}{\lambda_i}$ or $\frac{1}{\mu_i}$ respectively), p_i arbitrarily small, $0<p_i<1$. This process will regenerate itself at those instances of time at which, for every i, the exponential part of the distribution is in effect; the mean regeneration cycle is finite since a finite state Markov chain has no null recurrent states [4, p392, th.4].

3. Limiting Frequency and Average Duration of System Breakdown

Let the limiting frequency of system breakdown be, by definition, the average number of system breakdowns per unit time, i.e.

$\lim_{t\to\infty} \frac{N(t)}{t}$ where $N(t)$ is the number of system breakdowns in $[0,t]$.

Under the assumption (2.5) (i.e. system has regeneration points), we can obtain the limiting average system up time $E(\mathcal{U}_\infty)$ and down time $E(\mathcal{D}_\infty)$ as

$$E(\mathcal{U}_\infty) = \frac{\text{Prob\{ system is up \}}}{\text{frequency of system breakdown}} \tag{2.6}$$

and

$$E(\mathcal{D}_\infty) = \frac{\text{Prob\{ system is down \}}}{\text{frequency of system breakdown}} \tag{2.7}$$

Remark: In [8], (2.6) and (2.7) are assumed to be valid for any type of distribution. These are actually limiting average results (not exact results) for any distribution under the assumption that the system has regeneration points.

III. Frequency and Duration for One State

Consider a system of n independent components and a certain state $K = (x_1,\ldots,x_n)$. Let $N_K(t)$ denote the number of times the system reaches the state K in [0,t] and define the (limiting) frequency of encountering the state K as

$$f_K = \lim_{t\to\infty} \frac{N_K(t)}{t} \tag{3.1}$$

Let K_i be the state $(x_1,\ldots,x_{i-1},1-x_i,x_{i+1},\ldots,x_n)$, i.e. the state which differs from K by the i-th component. Let $N_{K,i}(t)$ be the number of times system reaches K due to a transition of the i-th component in [0,t], i.e. from K_i. Then,

$$f_K = \lim_{t\to\infty} \frac{N_K(t)}{t} = \lim_{t\to\infty} \frac{E[N_K(t)]}{t} \quad \text{with probability 1.}$$

But $E(N_K(t)) = \sum_{i=1}^{n} E(N_{K,i}(t))+o(t)$ due to the non-lattice assumption; hence

$$f_K = \lim_{t\to\infty} \frac{\sum_{i=1}^{n} E(N_{K,i}(t))}{t} = \sum_{i=1}^{n} \lim_{t\to\infty} \frac{E(N_{K,i}(t))}{t}$$

$$= \sum_{i=1}^{n} \lim_{t\to\infty} \frac{N_{K,i}(t)}{t} \quad \text{with probability 1.} \tag{3.2}$$

Let $f_{K,i} = \lim_{t\to\infty} \frac{N_{K,i}(t)}{t}$

= frequency of encountering state K from K_i. (3.3)

To find $f_{K,i}$, we can distinguish two similar cases:

i). $x_i = 1$; to reach K from K_i component i has to be repaired.

ii). $x_i = 0$; to reach K from K_i component i has to fail.

i). Assume that $x_i = 1$. We use the renewal reward argument. Let

$$\phi = \begin{cases} 1 & \text{if system is in } K \\ 0 & \text{otherwise} \end{cases}$$

and

$$I_j = \begin{cases} 1 & \text{if the j-th repair of i takes system from } K_i \text{ to } K \\ 0 & \text{otherwise} \end{cases}$$

Then

$$\lim_{t\to\infty} \frac{N_{K,i}(t)}{t} = \lim_{j\to\infty} \frac{N_{K,i}(\text{time of j-th repair of i})}{\text{time of j-th repair of i}}$$

$$= \lim_{j\to\infty} \frac{I_1+\cdots+I_j}{j} \cdot \frac{j}{\text{time of j-th repair of i}}$$

$$= \lim_{j\to\infty} \frac{I_1+\cdots+I_j}{j} \cdot \frac{1}{\frac{1}{\lambda_i}+\frac{1}{\mu_i}} \qquad (3.4)$$

But $$\lim_{j\to\infty} \frac{I_1+\cdots+I_j}{j} = \lim_{j\to\infty} \frac{E(I_1+\cdots+I_j)}{j} = \lim_{j\to\infty} E(I_j) \qquad (3.5)$$

and since

$E(I_j)$ = Prob{j-th repair of i takes system from K_i to K}

we have

$$E(I_j) = \text{Prob}\ \{\phi(0_i,\underline{x})=0, \phi(1_i,\underline{x})=1\ \} \qquad (3.6a)$$

and

$$f_{K,i} = \frac{\lambda_i\ \mu_i}{\lambda_i+\mu_i}\ \text{Prob}\ \{\phi(0_i,\underline{x})=0, \phi(1_i,\underline{x})=1\ \} \qquad (3.6b)$$

But $\phi(0_i,\underline{x}) = 0$ for any $\underline{x}$ by assumption and $\phi(1_i,\underline{x}) = 1$ if $\underline{x}$ is such that $(1_i,\underline{x}) = K$. Also

Prob $\{\phi(1_i,\underline{x})=1\}$ = Prob$\{\phi(\underline{x})=1 | x_i=1\}$. So

$$E(I_j) = \frac{\text{Prob}\{K\}}{\text{Prob}\{x_i=1\}} \tag{3.7}$$

Finally

$$f_{K,i} = \frac{\text{Prob}\{K\}}{\text{Prob}\{x_i=1\}} \cdot \frac{\lambda_i \mu_i}{\lambda_i+\mu_i} = f_i \frac{\text{Prob}\{K\}}{\text{Prob}\{x_i=1\}} \tag{3.8}$$

Similarly if $x_i=0$ in K we have

$$f_{K,i} = f_i \frac{\text{Prob}\{K\}}{\text{Prob}\{x_i=0\}} \tag{3.9}$$

We then get

$$\begin{aligned} f_K &= \text{Prob}\{K\} \sum_{i<u_K} \frac{f_i}{\text{Prob}\{x_i=1\}} + \sum_{j<d_K} \frac{f_j}{\text{Prob}\{x_j=0\}} \\ &= \text{Prob}\{K\} \sum_{i<u_K} \frac{\lambda_i\mu_i}{\lambda_i+\mu_i} \frac{\lambda_i+\mu_i}{\mu_i} + \sum_{j<d_K} \frac{\lambda_j\mu_j}{\lambda_j+\mu_j} \frac{\lambda_j+\mu_j}{\lambda_j} \\ &= \text{Prob}\{K\} \left[\sum_{i<u_K} \lambda_i + \sum_{j<d_K} \mu_j \right] \end{aligned} \tag{3.10}$$

In other words:

Frequency of encountering state K

$$= (\text{ probability of state K }) * (\text{rates out of K}) \tag{3.11}$$

Also, the limiting average time spent in state K is

$$E(\text{state K}) = \frac{\text{Prob}\{K\}}{f_K} = \frac{1}{\text{rates out of K}} \tag{3.12}$$

Remark: (3.10) is used in [6] based on an intuitive reasoning. Here we give a rigorous proof for this limiting result, in the case of general distributions approximated by a mixture of

exponential and another distribution as stated in II-2. [8] contains a proof for (3.12) in the case of exponential distributions. It is interesting to note that the result holds for any distribution of the form (2.5) as a limiting average result.

IV. Frequency and Duration for a Group of States.

1. Frequency of System Breakdown for a General System

Consider a system of n independent components as before. There are altogether 2^n states which constitute the state space $\underline{X}$ of the system. Let U ,called the set of 'up' states, be an arbitrary subset of the state space. Let D be its complement with respect to $\underline{X}$, called the set of 'down' states. Let ϕ be the characteristic function of U i.e.

$$\phi(x) = \begin{cases} 1 & \text{if } \underline{x} \in U \\ 0 & \text{if } \underline{x} \in D \end{cases}$$

As time goes on we can visualize our system as a random process moving from one state to the other. We want to find the limiting number of transitions from U to D per unit time refered to as the limiting frequency of system breakdown i.e.

$$f_D = \lim_{t \to \infty} \frac{N(t)}{t} \tag{4.1}$$

where N(t) denotes the number of transitions from U to D in [0,t].

Let $N_i(t)$ be the number of transitions from U to D in [0,t] due to the i-th component. As proven in III,

$$f_D = \sum_{i=1}^{n} \lim_{t \to \infty} \frac{N_i(t)}{t}$$

Let

$$f_D{}^i = \lim_{t \to \infty} \frac{N_i(t)}{t} \tag{4.2}$$

be the limiting number of system breakdowns per unit time due to the i-th component. We can split $N_i(t)$ into two parts:

$N_i^f(t)$: transitions from U to D in [0,t]
due to failure of the i-th component.

$N_i^r(t)$: transitions from U to D in [0,t]
due to repair of i-th component.

Then $N_i(t) = N_i^r(t) + N_i^f(t)$ and consequently, dividing by t and taking $\lim_{t\to\infty}$ we have

$$f^i = f^{i,r} + f^{i,f} \tag{4.3}$$

To find $f^{i,r}$ we use the same arguments as in III. By defining

$$I_j^r = \begin{cases} 1 \text{ if i-th repair of i causes a transition from U to D} \\ 0 \text{ otherwise} \end{cases}$$

we obtain

$$f^{i,r} = \frac{\lambda_i \mu_i}{\lambda_i + \mu_i} \text{Prob}\{\phi(0_i, \underline{x})=1, \phi(1_i, \underline{x})=0\} \tag{4.4}$$

Similarly,

$$f^{i,f} = \frac{\lambda_i \mu_i}{\lambda_i + \mu_i} \text{Prob}\{\phi(1_i, \underline{x})=1, \phi(0_i, \underline{x})=0\} \tag{4.5}$$

Then

$$f_D = \sum_{i=1}^{n} (f^{i,r} + f^{i,f}) \tag{4.6}$$

Now define $\underline{x}=(x_1,\ldots,x_n) \in U$ to be a boundary state of U if:

a. $x_i = 1$, $\phi(1_i,\underline{x})=1, \phi(0_i,\underline{x})=0$ i.e. $x_i = 1$ and $(0_i,\underline{x}) \notin D$. Call this a boundary state of U by i up.

b. $x_i = 0$. $\phi(0_i,\underline{x})=1, \phi(1_i,\underline{x})=0$ i.e. $x_i=0$ and $(1_i,\underline{x}) \notin D$. Call this a boundary state of U by i down

Then Prob{boundary states of U by i up }

$$= \text{Prob}\{ x_i=1, \phi(1_i,\underline{x})=1, \phi(0_i,\underline{x})=0\}$$

$$= \text{Prob}\{ x_i=1\} \text{ Prob}\{ \phi(1_i,\underline{x})=1, \phi(0_i,\underline{x})=0\}$$

since $\{\phi(1_i,\underline{x}), \phi(0_i,\underline{x})\}$ is independent of x_i

Similarly Prob{ boundary states of U by i down }

$$= \text{Prob}\{x_i=0, \phi(0_i,\underline{x})=1, \phi(1_i,\underline{x})0\}$$

$$= \text{Prob}\{x_i=0\} \text{ Prob } \{\phi(0_i,\underline{x})=1, \phi(1_i,\underline{x})=0\}$$

Substituting into f_D we obtain

$$f_D = \sum_{i=1}^{n} f_i \left(\frac{\text{Prob\{ boundary states of U by i up \}}}{\text{Prob}\{x_i=1\}} + \frac{\text{Prob\{ boundary states of U by i down}\}}{\text{Prob}\{x_i=0\}} \right) \qquad (4.7)$$

Two variations of (4.7) are :

1. Since Prob{boundary states of U by i up } = p_i Prob{boundary states of U by i up, excluding component i }

and

Prob {boundary states of U by i down } = (1 - p_i) Prob { boundary states by i down, excluding component i }

we have

$$f_D = \sum_{i=0}^{n} f_i \text{ (Prob\{boundary states by i up, excluding component i\}}$$

+ Prob {boundary states by i down, excluding component i}) (4.8)

2. Substitute $f_i = \frac{\lambda_i \mu_i}{\lambda_i + \mu_i}$, Prob$\{ x_i = 1\} = \frac{\mu_i}{\lambda_i + \mu_i}$ and Prob$\{x_i = 0\} = \frac{\lambda_i}{\lambda_i + \mu_i}$ into (4.7) to get

$$f_D = \sum_{i=1}^{n} (\lambda_i \text{ Prob\{ boundary states of U by i up } \} +$$

$$+ \mu_i \text{ Prob\{ boundary states of U by i down\}}) \qquad (4.9)$$

Remarks: 1. Assume U has only one state K of the form $(x_1, x_2 \ldots . x_n)$. Then to find f_K, note that

K is boundary state of U by i up for any $i \in u_K$

and

K is boundary state of U by i down for any $i \in d_K$

By (4.9)

$$f_K = \sum_{i \in u_K} \lambda_i \text{ Prob\{K\}} + \sum_{i \in d_K} \mu_i \text{ Prob\{K\}}$$

$$= \text{Prob\{K\}} \{ \sum_{i \in u_K} \lambda_i + \sum_{i \in d_K} \mu_i \}$$

which agrees with equation (3.10).

2. Assume that we have a two component-four state system. The state space is $\underline{x} = \{(0,0),(0,1),(1,0),(1,1)\}$.

Let U be the set $\{(0,1),(1,0)\}$. We want the frequency of going from U to $D = \underline{X}-U$; then

$(0,1)$ is boundary state of U by 1 down and 2 up,

$(1,0)$ is boundary state of U by 1 up and 2 down.

By (4.8)

$$f_D = f_1\{(1-p_2)+p_2\} + f_2\{(1-p_1)+p_1\} = f_1 + f_2 \qquad (4.10)$$

But $f_{(0,1)} = \text{Prob}\{(0,1)\}(\mu_1+\lambda_2) = (1-p_1)p_2(\mu_1+\lambda_2)$

$$= \frac{\lambda_1\mu_2\mu_1 + \lambda_1\mu_2\lambda_2}{(\lambda_1+\mu_1)(\lambda_2+\mu_2)}$$

Similarly $f_{(1,0)} = \dfrac{\mu_1\lambda_2\lambda_1 + \mu_1\lambda_2\mu_2}{(\lambda_1+\mu_1)(\lambda_2+\mu_2)}$

From this we get that $f_{(1,0)}+f_{(0,1)} = f_1+f_2$. So (4.10) agrees with the intuitive result that, since states (0,1) and (1,0) are not linked by a direct path, the frequency of encountering the group $\{(0,1),(1,0)\}$ must be $f_{(0,1)}+f_{(1,0)}$.

2. Frequency of System Repair

Dually let us define $\underline{x} = (x_1 \ldots x_n) \notin D$ to be a boundary state of D if

a) $x_i = 0$, $\phi(0_i,\underline{x})=0$, $\phi(1_i,\underline{x})=1$

called a boundary state of D by i down

b) $x_i = 1$, $\phi(1_i,\underline{x})=0$, $\phi(0_i,\underline{x})=1$

called a boundary state of D by i up

By the same argument as in IV-1, we can find f_U, the limiting number of transitions from D to U per unit time :

$$f_U = \lim_{t\to\infty} \frac{N^*(t)}{t} \tag{4.11}$$

where $N^*(t)$ denotes the number of transitions from D to U in [0,t]. Then

$$f_U = \sum_{i=0}^{n} f_i \left(\frac{\text{Prob\{boundary states of D by i up }\}}{\text{Prob}\{x_i=1\}} + \frac{\text{Prob\{boundary states of D by i down }\}}{\text{Prob}\{x_i=0\}}\right) \tag{4.12}$$

or

$$f_U = \sum_{i=0}^{n} f_i (\text{Prob\{boundary states of D by i down, excluding component i}\} + \text{Prob\{ boundary states by i up, excluding component i}\}) \tag{4.13}$$

However, for any boundary state J of D by i up (down), J_i is a boundary state of U by i down (up). Hence equations (4.13) and (4.8) together imply that

$$f_D = f_U \tag{4.14}$$

3. Average Duration of System Uptime and System Downtime

Given f_D, we can find the average time the system spends in U ($\mathcal{U}_\infty$) or in D ($\mathcal{D}_\infty$) by

$$E(\mathcal{U}_\infty)=\frac{\text{Prob\{ system is in U \}}}{f_D} \tag{4.15}$$

$$E(\mathcal{D}_\infty)=\frac{\text{Prob\{ system is in D \}}}{f_D} \tag{4.16}$$

Note: f_D is determined solely based on the probabilities of the boundary states of U (fig 1); to find $E(\mathcal{U}_\infty)$ and $E(\mathcal{D}_\infty)$, we must include the probabilities of all states in U and D.

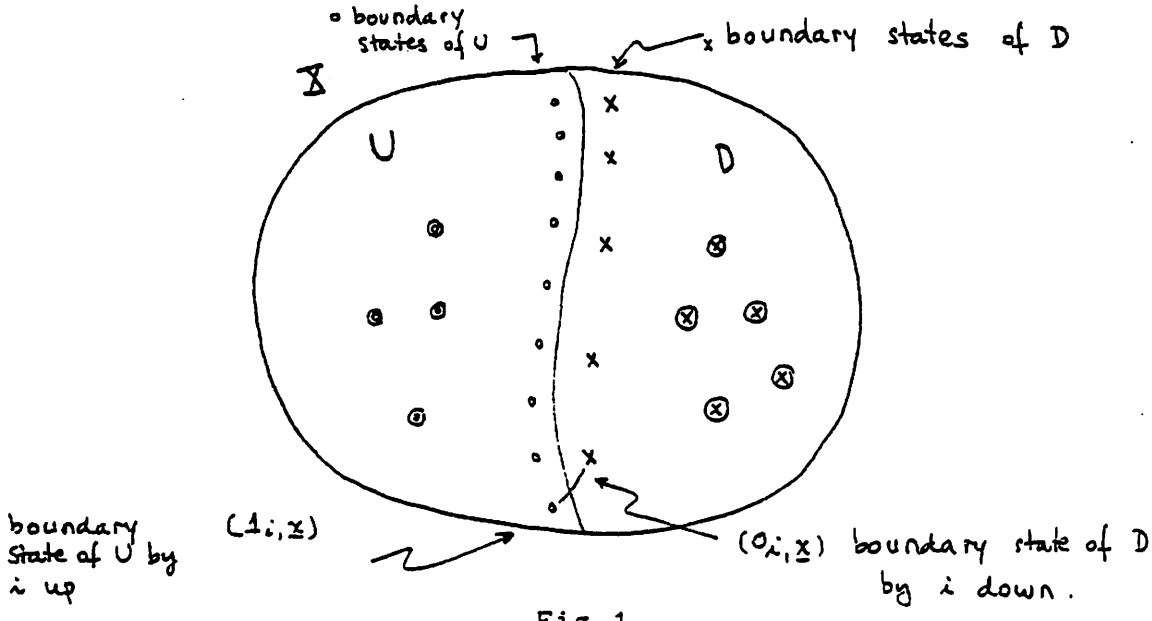


Fig 1

4. Frequency of System Breakdown for Coherent Systems

Suppose D has the following property :

If state K of the form $\underline{x} = (x_1 \cdots x_n) \lessdot D$

then, for any i , state $(0_i, \underline{x}) \lessdot D$; (4.17)

then the characteristic function ϕ is non-decreasing. To show this, note that

a) (4.17) implies

$$\phi(1_i, \underline{x}) = 0 \quad \Rightarrow \quad \phi(0_i, \underline{x}) = 0 \text{ for any } \underline{x} \tag{4.18}$$

b) Consider any state J of the form $\underline{x} = (x_1 \ldots x_n) \lessdot U$. For any $i \lessdot d_J$, $\phi(0_i, \underline{x}) = 1$. Then $\phi(1_i, \underline{x}) = 1$ (suppose not: then $\phi(1_i, \underline{x}) = 0$ and $\phi(0_i, \underline{x}) = 1$ which contradicts (4.18)).

So $\phi(0_i, \underline{x}) = 1 \Rightarrow \phi(1_i, \underline{x}) = 1$ for any $\underline{x}$

For convenience we shall call a system whose characteristic function ϕ is non-decreasing a coherent system. (Usually, a coherent system is characterized by ϕ non-decreasing and all components relevant [2,p6]. We will use the term coherency as defined above, not implying relevancy of components {which is not important for our purpose}) For such a system, the set of up states U has only boundary states by i up (for some components); then equations (4.7) to (4.9) become

$$f_D = \sum_{i=1}^{n} f_i \frac{\text{Prob\{boundary states of U by i up }\}}{\text{Prob}\{x_i = 1\}} \tag{4.19}$$

$$f_D = \sum_{i=1}^{n} f_i \text{Prob } \{\text{boundary states of U by i up, excluding component i }\} \quad (4.20)$$

$$f_D = \sum_{i=1}^{n} \lambda_i \text{ Prob}\{ \text{ boundary states of U by i up } \} \quad (4.21)$$

Remark. Reference [7] gives a heuristic development, based on renewal processes, for the frequency of breakdown for a two component - four state system. There they assume that for every component, the uptimes (downtimes) are independent identically distributed, non-lattice random variables. It is clear that the system does not renew itself for such general distributions. The formulas they provide, based on the assumption that the results for this simple case can be extended to large-scale systems, are true only as limiting frequencies or limiting average durations.

Examples

1. Consider a two component-four state system. The state space is $\underline{X} = \{(0,0),(0,1),(1,0),(1,1)\}$.

Let $U = \{(1,1),(0,1)\}$; $D = \{(1,0),(0,0)\}$.

The boundary states of U are

(1,1) by component 2 up

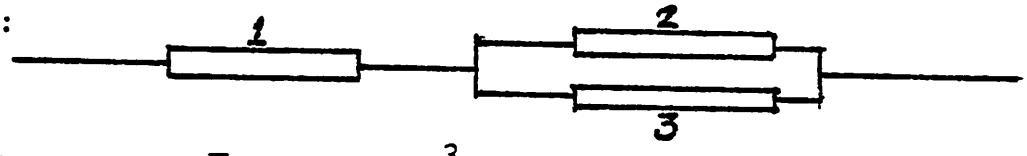
(0,1) by component 2 up

From equation (4.21) we get

$$f_D = \lambda_2(p_1p_2+(1-p_1)p_2) = \lambda_2 p_2 = f_2$$

Note : Here the first component is not relevant (hence $f_D = f_2$ agrees with the intuitive guess); yet formulas (4.19) through (4.21) can still be applied.

2. Consider a series - parallel system of 3 components as below:



The state space $\underline{X}$ contains $2^3=8$ states. The set of 'up' states U $=\{(1,1,1),(1,0,1),(1,1,0)\}$. Its boundary states are

(1,1,1) by 1

(1,0,1) by 1 and 3

(1,1,0) by 1 and 2

Then by equation (4.20)

$$f_D = f_1[p_2p_3+(1-p_2)p_3+(1-p_3)p_2] + f_2[p_1(1-p_3)]+f_3[p_1(1-p_2)] .$$

V Recursive Formula for Frequency Computation

1. Introduction

In IV we have derived a formula for the frequency of system breakdown. In this section we are going to describe a systematic and perhaps more efficient procedure of computing the frequency of breakdowns of a large coherent system for which the identification of boundary states are nontrivial. We will first show that there is a natural ordering of the down states of a coherent system. With this ordering we can construct a sequence $\{D_1, D_2, \cdots D_k\}$ of subsets of the state space $\underline{X}$, where D_j contains the first j elements of D and $D_k = D$ when k is equal to the number of elements in D. Then we are going to derive a recursive formula for computing the frequency of encountering the set D_j in terms of the frequency of encountering the set D_{j-1}. Let U_j be the complement of D_j with respect to $\underline{X}$.

2. State Ordering for Coherent Systems

$D \subseteq \underline{X}$ is a set of down states of a coherent system (i.e. it satisfies (4.17)) if and only if we can construct a sequence $\{D_1, D_2, \ldots, D_{|D|}\}$ (where the cardinality of D is $|D|$) of subsets of $\underline{X}$ such that

$$D_1 = \{ (0,0,..0) \}; \quad D_j = D_{j-1} \cup \{J\} \quad j=2,3,\ldots |D| \tag{5.1}$$

and

state $J = (x_1,\ldots,x_n)$ has the property:

for any $i \in u_J$, the state $(0_i,\underline{x}) \notin D_{j-1}$ (5.2)

To show the above, suppose we have such a sequence $\{D_1,D_2,\ldots,D_{|D|}\}$. Let $K = (y_1,y_2,\ldots y_n)$ be any state in $D_{|D|}$. Then, for any $i \in u_K$, $(0_i,\underline{y})$ belongs to $D_{|D|}$. So for any $y \notin D_{|D|}$ $\phi(1_i,\underline{y})=0 \Rightarrow \phi(0_i,\underline{y})=0$. So D is a set of down states of a coherent system.

Consider now $D = D_1 = \{ (0,0,\ldots 0) \}$ (D_1 satisfies (4.17)) and suppose that after j-1 steps we have D_{j-1}; we want to find a state J in U_{j-1} satisfying (5.2). We will show by contradiction that such a J exists.

Suppose there is no J satisfying (5.2) in U_{j-1}. Then for any state $K = (x_1,\ldots,x_n)$ in U_{j-1}, there exists an i in u_K such that $(0_i,\underline{x})$ belongs to U_{j-1}; but since

$K_i = (0_i,\underline{x})$ belongs to U_{j-1}, there exists an m belonging to u_{K_i} such that $(0_m,(0_i,\underline{x}))$ belongs to U_{j-1}. By finite induction, the state (0,0...0) belongs to U_{j-1} which is a contradiction. So we can find a state J in U_{j-1} to satisfy (5.2) and to build D_j recursively.

Remark: The sequence of subsets $\{D_1,\ldots D_{|D|}\}$ is not unique for a given set D.

3. Recursive Formula

To simplify notation, instead of using f_{D_j} let us use f(j) to denote the frequency of encountering the set D_j. We are going to derive a recursive formula for f(j).

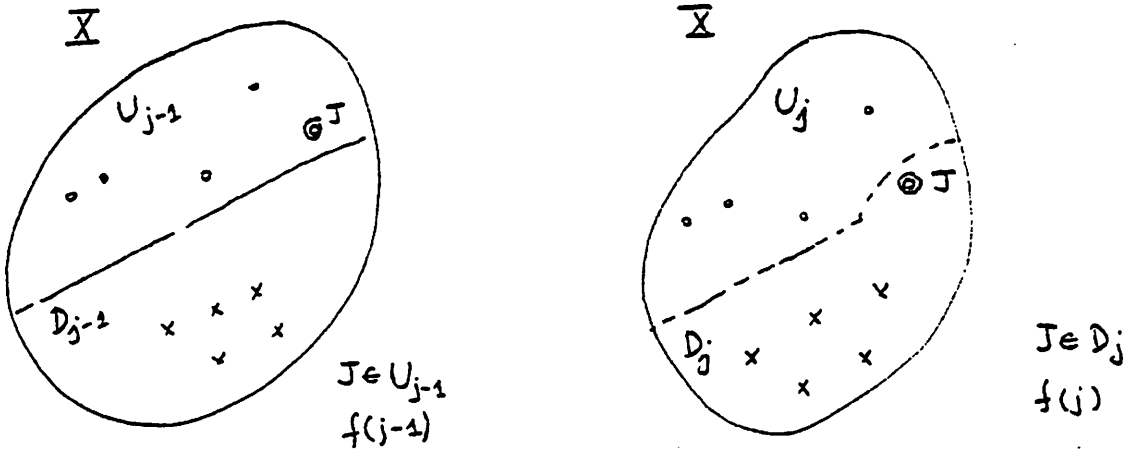


Fig 5.1

To derive the recursive formula we start with

$j = 1$ $\quad D_1 = \{ (0,\ldots,0) \}$. The boundary states of U_1 are

$(1,0,\ldots\ldots 0)$ by 1

$(0,1,\ldots\ldots 0)$ by 2

$\vdots$

$(0,0,\ldots\ldots 1)$ by n

Then we have by (4.21)

$$f(1) = \sum_{i=1}^{n} \lambda_i \quad \text{Prob}\{ \text{boundary state of } U_1 \text{ by } i\}$$

$$= \sum_{i=1}^{n} \lambda_i p_i \prod_{k\neq i} (1-p_k)$$

$$= \sum_{i=1}^{n} \lambda_i \frac{\mu_i}{\mu_i+\lambda_i} \prod_{k\neq i} \frac{\lambda_k}{\lambda_k+\mu_k}$$

$$= \sum_{i=1}^{n} \mu_i \prod_{k=1}^{n} \frac{\lambda_k}{\lambda_k+\mu_k}$$

$$= \text{Prob}\{(0,0,\ldots,0)\} \sum_{i=1}^{n} \mu_i \qquad (5.3)$$

$j = j+1$ We have $f(j-1)$ and want to derive $f(j)$. Look at D_{j-1} and U_{j-1}. Since $J = (x_1,x_2,\ldots,x_n)$ has the property that $(0_i,\underline{x}) \notin D_{j-1}$ for any $i \notin u_J$, the state J is a boundary state of U_{j-1} by all indices $i \notin u_J$. We can write $f(j-1)$ by (4.21) as

$$f(j-1) = \sum_{i=1}^{n} \lambda_i \text{ Prob\{ boundary states of } U_{j-1} \text{ by } i \}$$

$$= \sum_{i \lhd u_J} \lambda_i \text{ Prob\{ boundary states of } U_{j-1} \text{ excluding } J\}$$

$$+ \sum_{i \lhd d_J} \lambda_i \text{ Prob\{ boundary states of } U_{j-1} \text{ by } i\}$$

$$+ \sum_{i \lhd u_J} \lambda_i \text{ Prob\{ state } J \} \qquad (5.4)$$

Now we want to look at boundary states of U_j. Let J_i denote,as before, the state $(x_1,..,1-x_i,..x_n)$ where $(x_1,..,x_i,..,x_n)$ is state J i.e. J_i differs from J by the i-th component only. Then, for any $k \lhd d_J$, the states J_k are boundary states of U_j (in state J , for any $k \lhd d_J$, $x_k = 0$; so for J_k $x_k = 1$). Now

$$f(j) = \sum_{i=1}^{n} \lambda_i \text{ Prob\{boundary of } U_j \text{ by } i \}$$

$$= \sum_{k \lhd d_J} \lambda_k \text{ Prob\{boundary states of } U_j \text{ excluding } J_k\}$$

$$+ \sum_{k \lhd d_J} \lambda_k \text{ Prob}\{J_k\}$$

$$+ \sum_{k \lhd u_J} \lambda_k \text{ Prob\{ boundary states of } U_j \text{ by } k\} \qquad (5.5)$$

But

$$\sum_{i \lessdot u_J} \lambda_i \text{Prob}\{\text{boundary states of } U_{j-1} \text{excluding } J\}$$

$$+ \sum_{i \lessdot d_J} \lambda_i \text{Prob}\{\text{boundary states of } U_{j-1} \text{ by } i\}$$

$$= \sum_{k \lessdot d_J} \lambda_k \text{Prob}\{\text{boundary states of } U_j \text{excluding } J_k\}$$

$$+ \sum_{k \lessdot u_J} \lambda_k \text{Prob}\{\text{boundary states of } U_j \text{by } k\}$$

Hence we have

$$f(j) = f(j-1) - \sum_{i \lessdot u_J} \lambda_i \text{Prob}\{J\} + \sum_{k \lessdot d_J} \lambda_k \text{Prob}\{J_k\} \tag{5.6}$$

But $\text{Prob}\{J_k\} = \dfrac{\text{Prob}\{J\}}{\text{Prob}\{x_k = 0\}} \text{Prob}\{x_k=1\}$ for $k \lessdot d_J$. So

$$f(j)=f(j-1) - \sum_{i \lessdot u_J} \lambda_i \text{Prob}\{J\} +$$

$$\sum_{k \lessdot d_J} \lambda_k \frac{\mu_k}{\mu_k+\lambda_k} \frac{\mu_k+\lambda_k}{\lambda_k} \text{Prob}\{J\}$$

$$= f(j-1) - \sum_{i \lessdot u_J} \lambda_i \text{Prob}\{J\} + \sum_{k \lessdot d_J} \mu_k \text{Prob}\{J\} \tag{5.7}$$

Remarks 1. Assume we associate with each component k a certain capacity c_k. Then if state J is $\underline{x}=(x_1,\ldots,x_n)$, let its capacity be $\sum_{k=1}^{n} c_k x_k$. The increasing sequence of the state capacities gives a valid ordering of D. To see this suppose that the state J is to be added to D_{j-1}. Then for every $i \lessdot u_J$, the state$(0_i,\underline{x})$ has capacity $\sum_{k \neq i} c_k x_k = \sum_{k=1}^{n} c_k x_k - c_i < \sum_{k=1}^{n} c_k x_k$. This means that state

$(0_i,\underline{x})$ belongs to D_{j-1} due to its lower total capacity. This particular ordering is used in reference [6] to write (5.7) intuitively. It should be noted that (5.7) is not an exact result but rather a limiting one.

2. Another valid way of ordering is:

k = 0

D_1 includes the state with k (=0) components up i.e. state (0,0...0)

k=k+1

For this given k, append in any arbitrary order, all the states with k number of components up.

It is obvious that this is also a valid ordering since, if state J is $\underline{x}$ then for any $i<u_J$, $(0_i,\underline{x})$ has one less component up, so $(0_i,\underline{x})$ already belongs to D.

4. A Dual Recursive Formula

Dually we can order the set of up states of a coherent system. To be specific we have the following property:

$U \subseteq \overline{X}$ is a set of up states of a coherent system (with cardinality of $U = |U|$) if and only if we can construct a sequence $\{U_1,U_2,\ldots,U_{|U|}\}$ of subsets of $\overline{X}$ such that

$$U_1=\{(1,1,..1)\} \quad ; \quad U_j=U_{j-1}U\{J\} \qquad j=2,3...|U| \qquad (5.8)$$

where $J = (x_1, x_2, \ldots, x_n)$ has the property that for any $i \lessdot d_J$, the state $(1_i, \underline{x})$ belongs to U_{j-1}.

Similarly we can derive the recursive formula

$$f_{U_j} = f_{U_{j-1}} + \sum_{k \lessdot u_J} \lambda_k \text{ Prob}\{J\} - \sum_{k \lessdot d_J} \mu_k \text{ Prob}\{J\} \qquad (5.9)$$

Acknowledgements

I would like to thank Prof. Felix F. Wu for all the help and encouragement he gave me for this work. Also, I would like to take this opportunity to thank the Department of Energy, for the financial support they provided through the Electrical Energy Systems Division under the contract EC-77-S-01-5105.

References

1. Ross, S.M.
"On the Calculation of Asymptotic System Reliability Characteristics" Reliability and Fault Tree Analysis, Society for Industrial and Applied Mathematics 1975.
2. Barlow, R.E. and Proschan, F
"Statistical Theory of Reliability and Life Testing." Holt, Rinehart and Winston, Inc., 1975
3. Ross, S.M.
"Applied Probability Models with Optimization Applications" Holden - Day , 1970
4. Feller, W.
"An Introduction to Probability Theory and Its Applications" John Wiley & Sons, Inc., Vol I, Third Edition 1968.
5. DeSieno, C.F. and Stine, L.L
"A Probability Method for Determining the Reliability of Electric Power Systems", IEEE Transactions on Power Apparatus and Systems, Vol 83, pp 174-181, Feb 1964
6. Hall, J.D., Ringlee, R.J. and Wood, A.J.
"Frequency and Duration Methods for Power System Reliability Calculations :Generation System Model" IEEE Transactions on Power Apparatus and Systems Vol PAS-87 No 9, September 1968.
7. Ringlee, R.J. and Goode, S.D.
"On Procedure for Reliability Evaluation of Transmission Systems" , IEEE Transactions on Power Apparatus and Systems vol. PAS-89, no 4, April 1970.
8. Ayoub, A.K., Guy, J.D. and Patton, A.D.
"Evaluation of Some Methods for Calculating Generating System Reliability", IEEE Transactions on Power Apparatus and Systems vol PAS-89, no 4 , April 1970.

A DIRECT METHOD FOR POWER SYSTEM RELIABILITY EVALUATION

Khosrow Moslehi

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California 94720

ABSTRACT

The problem of reliability evaluation of a power system with DC model is formulated as the characterization and subsequent probability calculation of working states of that system. It is shown that power systems are generally not coherent. The concept of subminimal paths and subminimal cuts is introduced to develop bounds on the reliability function. A class of network topologies which guarantees coherency is also characterized. The concept of local coherency is introduced and its application to the identification of subminimal paths and subminimal cuts is described. A sufficient condition for local coherency is also developed. And finally a direct method for power system reliability evaluation is presented which does not require the solution of the load flow.

Research sponsored by the U. S. Department of Energy Contract EC-77-S-01-5105.

This manuscript was prepared on October 20, 1979.

TABLE OF CONTENTS

I. INTRODUCTION

The power system reliability problem is to evaluate the ability of a system to supply load demand while taking into account the random and planned outages of its equipment. The purpose of this study is to develop a methodology for bulk power system reliability evaluation using DC load flow model for the transmission system. Some methods have been proposed for bulk power system reliability evaluation, [1]. None of which have been satisfactory from both theoretical and computational points of view.

A simple and idealized problem of power system reliability evaluation is the identification and probability calculation of the set of working states of the system; a system is said to be in a working state if all load demands are satisfied and system equipment is operating within specified limits. We approach this problem by first analytically characterizing the set of working states.

The concept of coherent systems forms the foundation of the mathematical theory of reliability. A coherent system is one for which the performance of a component will not cause the system to deteriorate. For such a system, the set of working states can be characterized by minimal path sets or by minimal cut sets and reliability calculation as well as bounds can be obtained from the above characterization for coherent system. In this study we will examine the conditions under which a power system is a coherent system. The results in section II show that with the exception of power systems under generation improvement, power systems are generally not coherent; therefore, to characterize the set of working states of a noncoherent system the concept of subminimal paths and subminimal cuts is introduced. This concept will be used to develop upper and lower bounds on the reliability of the system.

For a fixed load demand, the noncoherency property is attributed to the effect of the transmission network on power flows. In section III we present a characterization of local coherency, i.e., coherency with respect to a subset of transmission lines. We also present a characterization of a class of network topologies for which coherency is guaranteed. This class of networks is quite limited. Later, in section III, we introduce an application of the concept of local coherency in the identification of subminimal paths and subminimal cuts of a power network. Finally, a sufficient condition for local coherency on the set of lines is established.

A method of reliability evaluation which is independent of the concept of coherency is suggested in section IV. A sequence of hyperboxes is constructed for this method and a subset of the working states is readily obtained for each hyperbox. The method is direct and does not require the solution of the load flow. Finally, a list of future work is included.

II. PROPERTIES OF POWER SYSTEM RELIABILITY MODEL

2.1. Problem Formulation

In the following study the power system $\mathbb{P}$ is considered to be a network with generators and load demands interconnected by transmission lines. We call both transmission lines and generators components of the system. Let n denote the number of components in the system and ℓ be the number of transmission lines. Thus, $(\ell-n)$ will be the number of generators. Let (N+1) be the number of nodes of the network. Each node corresponds to either a generator or a load demand or a combination of the two. Let $P_G \in \mathbb{R}_+^N$ represent the (real) power supply to the network from the generators, and let $P_D \in \mathbb{R}_-^N$ be the load demand of the system. Let us call the sum (P_G+P_D) node injection $P_N \in \mathbb{R}^N$

$$P_N = P_G + P_D \tag{2.1a}$$

Let A denote the reduced incidence matrix of the network. We assume that real power injection at the nodes P_N and the voltage phase angle at the nodes θ are related by the DC power flow equation

$$P_N = AYA^T\theta \tag{2.1b}$$

where Y is the branch admittance matrix, i.e., a diagonal matrix where $(Y)_{ii} = y_i$ is the admittance of line i. The line power flows P_L can be expressed as

$$P_L = YA^T\theta \tag{2.1c}$$

For steady-state operation, limits exist to the maximum power that can be supplied by individual generators and likewise on the maximum power that can be transferred by individual transmission lines.

Let C_G be the upper bounds on generators output

$$0 \leq P_G \leq C_G \tag{2.1d}$$

and let C_L be the vector of line capacities,

$$-C_L \leq P_L \leq C_L \tag{2.1e}$$

The power generation of the slack bus is equal to $(- P_{D_i} - (P_G)_i)$ where G_{sb} is the upper bound on it; hence, we have

$$\sum_{i=1}^{N} (P_D)_i \leq - \sum_{i=1}^{N} (P_G)_i \leq G_{sb} + \sum_{i=1}^{N} (P_D)_i \tag{2.1f}$$

Therefore, relations (2.1a-f) can be written in the matrix form as

$$Mz \leq b \tag{2.2}$$

where

$$M \triangleq \begin{bmatrix} AYA^t & -I \\ -AYA^t & I \\ 0 & I \\ 0 & -I \\ YA^t & 0 \\ -YA^t & 0 \\ -\Pi^t & 0 \\ \Pi^t & 0 \end{bmatrix} \qquad b \triangleq \begin{bmatrix} P_D \\ -P_D \\ C_G \\ 0 \\ C_L \\ C_L \\ C_{sb} \\ t_d \end{bmatrix} \qquad z \triangleq \begin{bmatrix} P_G \\ \theta \end{bmatrix}$$

and $t_d = \sum_{i=1}^{N} (P_D)_i$ total load demand

$$C_{sb} = G_{sb} + \sum_{i=1}^{N} (P_D)_i$$

$$\Pi^t = (1\ 1\ \ldots\ 1) \in \mathbb{R}^N$$

In reliability studies we are concerned mainly with the effect of component failure on the performance of the system. Let us suppose that each component may assume either a working state or a failed state. Let ξ_i be the binary indicator of the state of component i

$$\xi_i = \begin{cases} 0 & \text{component i failed} \\ 1 & \text{component i working} \end{cases}$$

The state of the system is defined by the states of its components, i.e., it is the collection of the component states.

$$\xi = \{\xi_1, \xi_2, \ldots, \xi_n\}$$

Thus, there are 2^n possible states for the entire system, and X_B will be used to denote the collection of these states (binary state space).

We would like to incorporate component failure directly into our initial formulation of the power system constraints (i.e., relation (2.2) so that one set of relations includes all these 2^n states. The following scheme can accomplish this purpose.

Let each diagonal element of Y, $(Y)_{ii}$, be a binary variable:

$$(Y)_{ii} = \begin{cases} 0 & \text{transmission line i is failed} \\ y_i & \text{transmission line i is working} \end{cases}$$

Let each element of C_G be a binary variable

$$(C_G)_i = \begin{cases} 0 & \text{generator i is failed} \\ g_i & \text{generator i is working} \end{cases}$$

The state of the system may thus be represented by a vector $x = (x_G, x_L) \in \mathbb{R}^{N+\ell}$ where the $\underline{i}$th component of x_G is $(C_G)_i$ and the $\underline{i}$th component of x_L is $(Y)_{ii}$. The space containing the 2^n possible states (state space) will be denoted by X.

For each system state, the values assumed by $(Y)_{ii}$ and $(C_G)_i$ are uniquely determined. Clearly, relation (2.2) represents the various constraints imposed on the values assumed by $(Y)_{ii}$ and $(C_G)_i$ for the corresponding system state.

For a system state $x \in X$, we say the system is working if all load demands are satisfied and system equipment is operating within specified limits, that is, all constraints in (2.2) are satisfied. Let us now define a set Ω_x for each $x \in X$ as

$$\Omega_x \triangleq \{z \in \mathbb{R}^{2N} | M_x z \leq b_x\} \tag{2.3}$$

where M_x and b_x are defined as M and b in (2.2) for state x. Then, a system state is said to be working iff $\Omega_x \neq \theta$. Let us denote the set of working states by W, i.e.,

$$W = \{x \in X | \Omega_x \neq \phi\}$$

Then, X can be partitioned as

$$X = W \cup F$$

where

$$F = \{x \in X | \Omega_x = \phi\}$$

In what follows, $\bar{x}_i$ and $\underline{x}_i$ will be used to indicate the values of the state variable i when component i is working or failed, respectively. If the state of every component except i is known, the relative system state will be denoted by $(\cdot i, x)$. The vectors $(\bar{x}_i, x)$ and $(\underline{x}_i, x)$ represent the system states when component i is working or is failed, respectively.

For ease of reference we may define structure function $\Phi(\ )$ to be

$$\Phi : X \rightarrow (0,1)$$

$$\Phi(x) = \begin{cases} 1 & x \in W \\ 0 & x \in F \end{cases} \qquad \forall x \in X$$

Alternatively, we may use the binary structure function $\phi(\ )$ which is defined as

$$\phi : X_B \rightarrow (0,1)$$

$$\phi(\xi) = \begin{cases} 1 & \text{is a working state} \\ 0 & \text{is a failed state} \end{cases} \qquad \forall \xi \in X_B$$

Similar to the notation for state space X, binary states $(0_i,\xi)$ and $(1_i,\xi)$ refer to states in X_B when component i is failed or working respectively.

We assume that all components of the system function independently. Furthermore, we assume that for each component the probability of working or failed is given, that is

$$\begin{cases} \text{Prob}\{x_i = \bar{x}_i\} = p_i \\ \text{Prob}\{x_i = \underline{x}_i\} = q_i = 1 - p_i \end{cases} \qquad \forall i \quad i \leq i \leq n$$

Consequently, for each system state $x \in X$, the probability of its occurring will simply be the product of the respective component state probabilities.

In evaluating the reliability of a power system $\mathbb{P}$ we are interested in the proability that the system is in a working state. Let us define the reliability of system $\mathbb{R}$ as

$$\begin{aligned} r(\mathbb{P}) &\triangleq \{\text{Probability that } \mathbb{P} \text{ is working}\} \\ &\triangleq \text{Prob}\{\Phi(x) = 1\} \\ &\triangleq E\{\Phi(x)\} \text{ i.e. expected value of } \Phi(x) \end{aligned}$$

As a result of the assumptions that components function independently, $r(\mathbb{P})$ will be the sum of the probabilities of working states

$$r(\mathbb{P}) = \sum_{x \in W} p(x)$$

Direct computation of reliability using the above formulation for a large system is prohibitive. Computation may be reduced by taking advantage of the properties of set W.

2.2. Reliability Coherency of Power Systems

The concept of coherent systems forms the basis of modern mathematical theory of reliability [2]. A vast amount of knowledge is available in reliability analysis of coherent systems. For such systems, the set of working states can easily be characterized and subsequently reliability bounds obtained. In this section we will examine the question as to whether the power system model (2.2) is indeed a coherent system.

First, let us state the definition of a coherent system.[†]

DEFINITION: A system $\mathbb{P}$ is a coherent system (or coherent structure) if the structural function $\Phi(\ )$ of that system is nondecreasing, that is,

$$\forall x, x' \in X \quad \text{if } x \geq x' \quad \text{then } \Phi(x) \geq \Phi(x')$$

Intuitively, the coherency property of a system implies that improving the state of any subset of components will not cause deterioration of the system.

PROPERTY II-1: Coherency under Generation Improvement

Consider the power system $\mathbb{P}$; for any fixed x_L, $\phi(\cdot, x_L)$ is a nondecreasing function.

[†] We assume all components in a system $\mathbb{P}$ are relevant.

For proof refer to Appendix (II-1).

Property (II-1) implies that for a fixed network, a power system is "coherent" under generation improvement. However, it should be pointed out that the results obtained in the Appendix (II-1) shows that if the lower capacity limits of the generators are not zero, generation improvement may cause system state to deteriorate (noncoherency property).

PROPERTY II-2: Noncoherency under Load Shedding

Throughout this work we have been assuming a constant set of load demands. However, in the following analysis we will examine the effect of load shedding (in a way this can be called load improvement) on the performance of the system with varying levels of load demand (i.e., different states of load demand).

Consider the power system $\mathbb{P}$ and any fixed state $x \in X$. Let P_D and P_D' be two possible load demand vectors. Suppose state x is a working state for load demand P_D. However given inequality $P_D \geq P_D'$, state x may not necessarily be a working state for load demand P_D'. For proof refer to Appendix (II-1).

Intuitively, for a fixed state $x \in X$, the state of the system may deteriorate, however, if load demand is decreased at certain nodes. As mentioned, we have assumed constant load demand and thereby have avoided the noncoherency property of load shedding.

PROPERTY II-3: Noncoherency under Transmission Network Improvement

Consider the power system $\mathbb{P}$ and a state $x = (x_G, x_L)$. Let x' be any state $x' = (x_G', x_L')$ such that

$$x'_G = x_G$$

$$x'_L = x_L$$

That is, for both states x and x' the state of genrators are the same and the state of transmission lines in x' are improved (or at least not worsened) with respect to x. If the system were coherent under transmission network improvement then we would have $\phi(x') \geq \phi(x)$. The following example shows that this may not always be true, though.

EXAMPLE (II-2-1)

Consider the power system in Fig. (II-1).

Let $P_D = \begin{bmatrix} 15 \\ 20 \end{bmatrix}$, $C_L = \begin{bmatrix} 15 \\ 20 \\ 10 \\ 35 \end{bmatrix}$ and $y = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix}$

CASE 1. Consider state $x^1 = (1,2,0,1)$. The unique solution to power flow P_L^1 in (2.1) will be

$$P_L^1 = \begin{bmatrix} 15 \\ 20 \\ 0 \\ 35 \end{bmatrix} \Rightarrow |P_L^1| \leq C_L \Rightarrow \Phi(x') = 1$$

CASE 2. Consider state $x^2 = (1,2,3,1)$. The unique solution to power flow P_L^2 will then be

$$P_L^2 = \begin{bmatrix} 12.3 \\ 22.7 \\ 2.7 \\ 3.5 \end{bmatrix} \Rightarrow |P_L^2| \not\leq C_L \Rightarrow \Phi(x^2) = 0$$

Hence, for $x^2 > x^1$ we have $\Phi(x^2) < \Phi(x^1)$.

Therefore, the structure of power system model of (2.2) is generally not coherent under the system transmission network improvement.

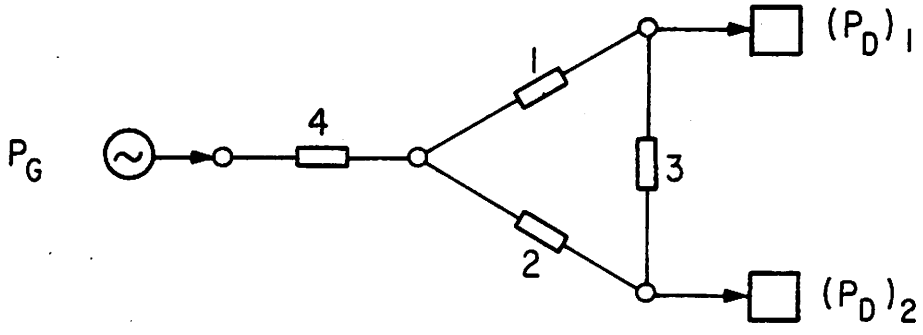


Fig. II-1.

2.3. Other Properties of Power System Reliability Model

In the following we present some alternative characterization of set W. These various characterizations will in some capacity influence the approaches to solving the problem of power systems reliability.

FACT II-1. Consider set Ω_x as defined in (2.3).

$$\Omega_x = \{z \in \mathbb{R}^{2N} | M_x z \leq b_x\}$$

A necessary and sufficient condition for Ω_x to be nonempty is:

$$\forall \lambda \geq 0 \quad \lambda \epsilon N(M^t) \quad \lambda^t b \geq 0 \tag{2.4}$$

where $N(M^t)$ denotes the null space of the transpose of matrix M.

Proof: Refer to Appendix (II-1).

Let $\lambda \triangleq \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_8, \lambda_7, \lambda_8)$ such that (2.4) can be written as (2.5)

$$\begin{cases} (\lambda_1-\lambda_2)^t P_D + \lambda_3^t C_G + (\lambda_5+\lambda_6)^t C_L + \lambda_7 G_{sb} - \lambda_8 t_D & \text{(a)} \\ (\lambda_1-\lambda_2)^t [AYA^t : -I] + [(\lambda_5-\lambda_6)^t YA^t + (\lambda_8-\lambda_7)^{\Pi^t} : (\lambda_3-\lambda_4)I] = 0 & \text{(b)} \\ \lambda_i \geq 0 \quad i = 1,2,\ldots,6 & \text{(c)} \end{cases} \tag{2.5}$$

relation (2.5b) reveals that $\lambda_1 - \lambda_2 = \lambda_3 - \lambda_4$; therefore, (2.4) is equivalent to:

$$\lambda'^t \eta \geq 0 \qquad \forall \lambda' \geq 0, \ \lambda' \in N(M'^t) \tag{2.6}$$

where

$$\lambda' = \begin{bmatrix} \lambda_3 \\ \lambda_4 \\ \lambda_5 \\ \lambda_6 \\ \lambda_7 \\ \lambda_8 \end{bmatrix}, \quad M' \triangleq \begin{bmatrix} AYA^t \\ -AYA^t \\ YA^t \\ -YA^t \\ -\Pi^t \\ \Pi^t \end{bmatrix} \quad \text{and} \quad \eta \triangleq \begin{bmatrix} P_D+C_G \\ -P_D \\ C_L \\ C_L \\ C_{sb} \\ -t_D \end{bmatrix}$$

Let us define set K as

$$K \triangleq \{\lambda' \in \mathbb{R}_+^{2(\ell+N+1)} \mid \lambda'^t M' = 0\}$$

where, by definition, K is a convex cone [3].

FACT II-2. From relation (2.6), $\Omega_x \neq \phi$ iff set $K^0 \neq \phi$ where

$$K^0 \triangleq \{\eta \in \mathbb{R}^{2(\ell+N+1)} \mid \eta^t \lambda' \leq 0 \quad \forall \lambda', \eta \text{ defined in (2.6)}\}$$

and, by definition K^0 is the polar of cone K, thus it is a convex cone [3].

FACT I-3. Let set K' be defined as

$$K' \triangleq \{(C_{sb}, P_D+C'_G, -P_D, C_L) \in \mathbb{R}^{2N+\ell+1} \mid (P_D+C_G, -P_D, C_L, C_L, C_{sb}, -t_D) \in K^0\}$$

Then

$$\Omega_x \neq \phi \Leftrightarrow K' \neq \phi$$

Using the definition of a convex cone and FACT (II-2), it can be shown that K' is also a convex cone.

Thus, given a fixed network configuration Y for all (P_D, C_G, C_L, C_{sb}) for which their respective states x are working states, vectors $(C_{sb}, P_D+P_G, -P_D, C_L)$ define a convex cone.

REMARK: Consider a power system model (2.2) and suppose node injection P_N is fixed. Then (2.2) is equivalent to:

$$P_N = AYA^t\theta$$
$$-C_L \leq P_L^Y = YA^t\theta \leq C_L \tag{2.7}$$

Let us define $Y \triangleq \{y_L \in \mathbb{R}^{\ell} \mid (Y)_{ii} = (y_L)_i,\ -C_L \leq P_L^y \leq C_L\}$

$$f : Y \to \mathbb{R}^{\ell}$$
$$f(y) = P_L^Y \tag{2.8}$$

It can easily be shown that the set of line admittances y for which the power flow P_L^Y is feasible is not a convex set, and that the function f(') is infact a nonlinear function. Thus, it is expected that the set W does not possess simple characterizations.

2.4. Reliability Theory of Noncoherent Systems

In this section we will examine the effect of noncoherency on some of the well-established relationship in reliability theory of coherent systems. We will, then, present the concept of subminimal paths and subminimal cuts so as to develop upper and lower bounds on the structure function of the system. Throughout this section, unless mentioned otherwise, a system is not necessarily coherent.

For the systems that we deal with we assume that every component is relevant, that is, for any component i

$$\Phi(\bar{x}_i, x) - \Phi(\underline{x}_i, x) \neq 0 \quad \text{for some } (\cdot, x) \in X$$

We also assume that none of the components have a destructive effect on the system performance, that is, there is no component i such that

$$\Phi(\bar{x}_i, x) = 0 \qquad \forall\ (\bar{x}_i, x) \in X$$
$$\Phi(\underline{x}_i, x) = 1 \qquad \text{for some } (\underline{x}_i, x) \in X$$

Furthermore, we assume

$$\Phi(\bar{x}) = 1 \quad \text{for } \bar{x} \in X \text{ where } (\bar{x})_i = \bar{x}_i \qquad i = 1,2,\ldots,n$$

and

$$\Phi(\underline{x}) = 0 \quad \text{for } \underline{x} \in X \text{ where } (\underline{x})_i = \underline{x}_i \qquad i = 1,2,\ldots,n$$

For the binary structure function $\phi(\ )$, not necessarily coherent, we have [2]

$$\coprod_{i=1}^{n} \xi_i \le \phi(\xi) \le \coprod_{i=1}^{n} \xi_i \qquad \forall \xi \in X_B \tag{2.9}$$

For a coherent system we have [2]

$$\forall \xi^1 \ \& \ \xi^2 \in X_B \begin{cases} \phi(\xi^1 \amalg \xi^2) \ge \phi(\xi^1) \amalg \phi(\xi^2) \\ \phi(\xi^1 \cdot \xi^2) \le \phi(\xi^1) \cdot \phi(\xi^2) \end{cases} \tag{2.10}$$

where

$$x \cdot y \triangleq \min(x,y)$$

$$x \amalg y \triangleq \max(x,y)$$

This property can be used to develop bounds on the reliability function. However, the relation (2.10) is not always valid for noncoherent structures.

Similarly, the following results can also be established for binary states space X_B and binary structure function $\phi(\ )$.

Let set $S = \{1,2,\ldots,n\}$ be the set of components in the system and let for given $x \in X$

$$\bar{S}_x = \{i \in S \mid (x)_i = \bar{x}_i\}$$

$$\underline{S}_x = \{i \in S \mid (x)_i = \underline{x}_i\}$$

A path vector is a state vector $x \in X$ such that $\Phi(x) = 1$. The corresponding $\bar{S}_x$ is a path set. A minimal path vector is a state vector $x \in X$ such that

$$\begin{cases} \Phi(x) = 1 \\ \Phi(x') = 0 \qquad \forall x' \leq x \quad x' \in X \end{cases}$$

The corresponding $\bar{S}_x$ is a minimal path set. A cut vector is a state vector $x \in X$ such that $\Phi(x) = 0$. The corresponding $\underline{S}_x$ is a cut set. A minimal cut vector is a state vector $x \in X$ such that

$$\begin{cases} \Phi(x) = 0 \\ \Phi(x') = 1 \qquad \forall x' \geq x \quad x' \in X \end{cases}$$

The corresponding $\underline{S}_x$ is a minimal cut set.

For a coherent system, set $W = \{x | \Phi(x) = 1\}$ can be characterized by minimal paths or minimal cuts. Furthermore, the minimal paths and minimal cuts can be used to obtain bounds on the reliability of the respective coherent system. Noncoherent systems, however, do not possess this property. Thus, for these system we introduce the concept of subminimal paths and subminimal cuts which characterize set W partially and can be used to obtain bounds on the reliability function

DEFINITION

A path vector $x \in X$ is called subminimal path vector if

$$\forall x' \in X \text{ for some } j \in \bar{S}_x \begin{cases} (x')_i = (x)_i & i = 1,\ldots,n \quad i \neq j \\ (x')_j = \underline{x}_j \end{cases}$$

Then x' is a minimal cut vector. Corresponding $\bar{S}_x$ is called a subminimal path set.

A cut vector $x \in X$ is called subminimal cut vector if

$$\forall x' \in X \quad \text{for some } j \in \underline{S}_x \begin{cases} (x')_i = (x)_i & i = 1,2,\ldots,n \; i \neq j \\ (x')_j = \bar{x}_j \end{cases}$$

Then x' is a minimal path vector. Corresponding $\underline{S}_x$ is a <u>subminimal cut set</u>.

Let the sets SP and SK represent the set of all subminimal path sets and subminimal cut sets of a system, respectively. For any subminimal path set $\bar{S} \in SP$ we define the binary function $\rho_{\bar{S}}(\ )$ such that

$$\rho_{\bar{S}}(x) = \prod_{i \in \bar{S}} \xi_i^x$$

And for any subminimal cut set $\underline{S} \in SK$ we define the binary function $\gamma_{\underline{S}}(\ )$ such that

$$\gamma_{\underline{S}}(x) = \prod_{i \in \underline{S}} \xi_i^x$$

where $\xi_i^x \triangleq (\xi^x)_i$ and $\xi_i^x = \begin{cases} 0 & (x)_i = \underline{x}_i \\ 1 & (x)_i = \bar{x}_i \end{cases}$

<u>THEOREM II-4-1</u>. For a system with structure function $\Phi(\ )$ we have $\forall x \in X$

$$\prod_{\underline{S} \in SK} \gamma_{\underline{S}}(x) \geq \Phi(x) \geq \prod_{\bar{S} \in SP} \rho_{\bar{S}}(x) \tag{2.11}$$

and

$$\prod_{\underline{S} \in SK} \gamma_{\underline{S}}(x) \geq \Phi(x) \geq \prod_{\bar{S} \in SP} \rho_{\bar{S}}(x) \tag{2.12}$$

<u>Proof</u>. Refer to Appendix (II-1)

<u>Bounds on Power System Reliability</u>

Let us recall that for any binary variable b, we have

$$E(b) = \text{Prob}\{b=1\}$$

and for a system $\mathbb{P}$,

$$r(\mathbb{P}) = E(\phi(\xi)) = \text{Prob}\{\phi(\xi) = 1\}$$

$$= E(\Phi(x)) = \text{Prob}\{\Phi(x) = 1\}$$

THEOREM II-4-2. If $\phi(\ )$ is the binary structure function of a system $\mathbb{P}$, then

$$\prod_{i=1}^{n} p_i \leq r(\mathbb{P}) \leq \coprod_{i=1}^{n} p_i$$

Proof. Refer to Appendix (II-1)

Let us use the result of Theorem (II-4-1) to develop bounds on system reliability. Taking the expected value of the expression (2.11) gives

$$E\Big(\prod_{\underline{S}\in SK} \gamma_{\underline{S}}(x)\Big) \geq r(\mathbb{P}) \geq E\Big(\coprod_{\bar{S}\in SP} \rho_{\bar{S}}(x)\Big) \tag{2.13}$$

For a large system, it is usually a formidable task to compute the exact values of the above lower and upper bounds. An efficient method for computing these bound is to apply a decomposition technique similar to the one used in [8]. To do so we must decompose the subset of working states which is characterized by the subminimal paths and subminimal cuts into disjoint subsets. The value for the bounds then, follows by properly adding the probabilities of all such subsets of states.

We may apply the inclusion-exclusion principle [9] to compute some bounds on the values of the bounds on system reliability given by (2.13). However, the bounds given by the following theorem can be easily computed.

THEOREM II-4-3. Suppose $\Phi(\ )$ is the structure function of a system. Let SK and SP be the set of all subminimal cuts and subminimal paths,

respectively. Then

$$\prod_{\underline{S}\in SK} \text{Prob}\{\gamma_{\underline{S}}(x) = 1\} \geq r(\mathbb{P}) \geq \prod_{\bar{S}\in SP} \text{Prob}\{\rho_{\bar{S}}(x) = 1\}$$

Proof. Refer to Appendix (II-1)

APPENDIX (II-1)

FACT II-1

Proof. Consider the problem of determining whether Ω_x is empty. We may transform the problem into a standard linear programming problem.

$$P\begin{cases} \min\limits_{z\in\mathbb{R}^{2N}} \langle \underline{0}, z\rangle \\ \text{s.t.} \\ \quad Mz \leq b \end{cases}$$

Let $m(z) = \max \lambda^t(Mx-b)$. Then,

$$m(z) = \begin{cases} 0 & x \in \Omega \\ \infty & x \in \mathbb{R}^{2N} \end{cases}$$

Hence, problem P may be written as

$$P\begin{cases} \min\limits_{z\in\mathbb{R}^{2N}} \max\limits_{\lambda\geq 0} \langle \underline{0}, z\rangle + \langle \lambda, (Mz-b)\rangle \end{cases}$$

The dual of this problem can, then, be written [5] as problem D:

$$D\{\max_{\lambda\geq 0} \min_{z\in\mathbb{R}^{2N}} \langle \lambda, (Mz-b)\rangle$$

Let

$$\mu(\lambda) = \min_{z\in\mathbb{R}^{2N}} \lambda^t Mz$$

$$\mu(\lambda) = \begin{cases} -\infty & \lambda^t M \neq 0 \\ 0 & \lambda^t M = 0 \end{cases}$$

Because of the boundedness of the solution, problem D is equivalent to

$$D \begin{cases} \max\limits_{\lambda \geq 0} -\lambda^t b \\ \text{s.t.} \\ \quad \lambda^t M = 0 \end{cases}$$

But, we have

solution to D = solution to P = 0

Hence, from the duality theorem, there is a bounded solution to problem P iff

$$\begin{cases} \min \lambda^t b \geq 0 \\ \text{s.t.} \\ \quad \lambda \geq 0, \quad \lambda^t M = 0 \end{cases}$$

Q.E.D.

PROPERTY II-1

Proof. For some fixed x_L, let $x = (x_G, x_L)$ be some state in X. Then consider any state $x' = (x'_G, x_L)$ such that $x'_G \geq x_G$.

CASE 1. if $\phi(x) = 0$ then either $\phi(x') = 0$ or 1.

CASE 2. if $\phi(x) = 1$ then from Fact (II-1) this is equivalent to

$$\Omega_x \neq \phi \Leftrightarrow \min_{\substack{\lambda \geq 0 \\ \lambda \in N(M_x^t)}} \lambda^t b_x \geq 0 \qquad \text{(A.1)}$$

From the definition of x and x' we have:

$$M_x = M_{x'} \text{ and } b_x = \begin{bmatrix} P_D \\ -P_D \\ C_G \\ \underline{0} \\ C_L \\ C_L \\ C_{sb} \end{bmatrix}, \quad b_{x'} = \begin{bmatrix} P_D \\ -P_D \\ C'_G \\ \underline{0} \\ C_L \\ C_L \\ C_{sb} \end{bmatrix} \quad \text{where } C'_G \geq C_G$$

Hence, from (A.1):

$$\min_{\substack{\lambda \geq 0 \\ \lambda \in N(M_x^t)}} \lambda^t b_{x'} \geq 0$$

or

$$\min_{\substack{\lambda \geq 0 \\ \lambda \in N(M_x^t)}} \lambda^t b_{x'} \geq 0 \qquad \text{(A.2)}$$

but, (A.2) is equivalent to $\Omega_{x'} \neq \phi \Rightarrow \Phi(x') = 1$. Q.E.D.

REMARK. if the lower capacity limits of the generators are $C_g \neq 0$,

i.e., $0 \neq \underline{C}_g \leq P_G \leq C_G$

then b_x and $b_{x'}$ will be

$$b_x = \begin{bmatrix} P_D \\ -P_D \\ C_G \\ -C_G \\ C_L \\ C_L \\ C_{sb} \end{bmatrix}, \quad b_{x'} = \begin{bmatrix} P_D \\ -P_D \\ C'_G \\ -C'_g \\ C_L \\ C_L \\ C_{sb} \end{bmatrix}$$

where $C'_G \geq C_G$ and $C'_g \geq C_g$. Then $\forall \lambda \geq 0$

$$\lambda^t b_x \geq 0 \not\Rightarrow \lambda^t b_{x'} \geq 0$$

which implies that the lower capacity limits of certain generators are nonzero; placing those generators into operation may in fact deteriorate the system performance for some states.

PROPERTY II-2

Proof. If state x is working for load demand P_D, from FACT (II-1) we have

$$\Phi(x) = 1 \Leftrightarrow \min_{\substack{\lambda \geq 0 \\ \lambda \in N(M_x^t)}} \lambda^t b_x \geq 0$$

where $b_x = (P_D, -P_D, C_G, \underline{0}, C_L, C_L, C_{sb})$.

Let $b'_x = (P'_D, -P'_D, C_G, \underline{0}, C_L, C_L, C_{SG})$

However, $-P'_D \geq -P_D$ does not imply that $b'_x \geq b_x$ or $-P'_D \geq -P'_D$ does not imply that $\lambda^t b'_x \geq \lambda^t b_x \geq 0 \ \forall \lambda \geq 0$ although we may have $\lambda^t b'_x < 0$. Hence, state x for load demand P'_D may be a failed state. Q.E.D.

THEOREM II-4-1

Proof. taking the expected value of expression (2.9), we obtain

$$E\left(\prod_{i=1}^{n} \xi_i\right) \leq E\{\phi(\xi)\} \leq E\left(\prod_{i=1}^{n} \xi_i\right)$$

Since variables ξ_i i = 1,...,n are independent, the result follows.

Q.E.D.

THEOREM II-4-2

Proof. (a) constraint (2.11)

(1) if $\exists$ some $\bar{S} \in SP \ni \rho_{\bar{S}}(x) = 1$ we have

$$\begin{cases} \Phi(x) = 1 \\ \prod_{\bar{S} \in SP} \rho_{\bar{S}}(x) = 1 \end{cases}$$

(2) if $\forall \bar{S} \in SP \quad \rho_{\bar{S}}(x) = 0$ then

$$\begin{cases} \Phi(x) = 0 \text{ or } 1 \\ \prod_{\bar{S} \in SP} \rho_{\bar{S}}(x) = 0 \end{cases}$$

$$(1)\&(2) \Rightarrow \Phi(x) \geq \prod_{\bar{S} \in SP} \rho_{\bar{S}}(x) \quad \forall x \in X$$

(3) if $\exists$ some $\underline{S} \in SK \ni \gamma_{\underline{S}}(x) = 0$ we have

$$\begin{cases} \Phi(x) = 0 \\ \prod_{\underline{S} \in SK} \gamma_{\underline{S}}(x) = 0 \end{cases}$$

(4) if $\forall \underline{S} \in SK \quad \gamma_j(x) = 1$ then

$$\begin{cases} \Phi(x) = 1 \text{ or } 0 \\ \prod_{\underline{S} \in SK} \gamma_{\underline{S}}(x) = 1 \end{cases}$$

$$(3)\&(4) \Rightarrow \Phi(x) \leq \prod_{\underline{S} \in SK} \gamma_{\underline{S}}(x) \quad \forall x \in X$$

(b) Constraint (2.12)

We have

$$\text{(i)} \quad \prod_{\underline{S} \in SK} \gamma_{\underline{S}}(x) \leq \prod_{\underline{S} \in SK} \gamma_{\underline{S}}(x)$$

(ii) $\prod_{\bar{S}\in SP} \rho_{\bar{S}}(x) \leq \prod_{\bar{S}\in SP} \rho_{\bar{S}}(x)$

The result follows by plugging (i) and (ii) in (2.11)

Q.E.D.

THEOREM II-4-3

Proof.

(a) $\forall \underline{S} \in SK$, $\gamma_{\underline{S}}(\ )$ is a nondecreasing function of x and x_i i = 1,2,...,n are independent r.v. Hence $\gamma_{\underline{S}}(x)$, $\underline{S} \in SK$ are associated random variables and we have [4]

$$\text{Prob}\{\prod_{\underline{S}\in SK} \gamma_{\underline{S}}(x) = 1\} \leq \prod_{\underline{S}\in SK} \text{Prob}\{\gamma_{\underline{S}}(x) = 1\} \qquad \text{(i)}$$

The relation below follows from (2.12)

$$\text{Prob}\{\prod_{\underline{S}\in SK} \gamma_{\underline{S}}(x) = 1\} \geq \text{Prob}\{\Phi(x) = 1\} \qquad \text{(ii)}$$

(i) and (ii) imply

$$\prod_{\underline{S}\in SK} \text{Prob}\{\gamma_{\underline{S}}(x) = 1\} \geq \text{Prob}\{\Phi(x) = 1\}$$

(b) $\forall \bar{S} \in SP$, $\rho_{\bar{S}}(\ )$ is a nondecreasing function of independent r.v. x_i i = 1,2,...,n. Hence $\rho_{\bar{S}}(x)$, $\bar{S} \in SP$ are associated random variables and we have [4]

$$\text{Prob}\{\prod_{\bar{S}\in SP} \rho_{\bar{S}}(x) = 1\} \geq \prod_{\bar{S}\in SP} \text{Prob}\{\rho_{\bar{S}}(x) = 1\} \qquad \text{(iii)}$$

From relation (2.12)we can write

$$\text{Prob}\{\Phi(x) = 1\} \geq \text{Prob}\{\prod_{\bar{S}\in SP} \rho_{\bar{S}}(x) = 1\} \qquad \text{(iv)}$$

The result follows from (iii) and (iv).

Q.E.D.

III. COHERENCE ANALYSIS OF POWER SYSTEM RELIABILITY MODEL

We have shown that power systems are generally not coherent structures. This noncoherency is attributed to the effect of transmission network on power flows. In this section we present (i) a characterization of local coherency, i.e., coherency with respect to a subset of lines, (ii) a characterization of a class of network topologies for which coherency is guaranteed, and (iii) some sufficient conditions for local coherency.

3.1. Equivalent Model for Change in Network State

For a power system $\mathbb{P}$, suppose the set of transmission lines is partitioned into subsets L1 and L2. We are going to derive conditions which guarantee that if in the absence of lines L1, the system is working, then, with lines L1, the system is still working.

The state space X can then be written as

$$X = X_G \times X_{L2} \times X_{L1}$$

where X_G, X_{L2} and X_{L1} are subspaces of generators, lines in L1 and lines in L2, respectively. Then any $x \in X$ can be written as

$$x = \begin{pmatrix} x_{LG} \\ x_{L1} \end{pmatrix} \quad \text{where} \quad \begin{matrix} x_{LG} \in X_G \times X_{L2} \\ x_{L1} \in X_{L1} \end{matrix}$$

Consequently we can write:

$C_L = \begin{pmatrix} C_{L1} \\ C_{L2} \end{pmatrix}$, $A = [A^1\ A^2]$ and $Y = \left[\begin{array}{c|} Y^1 \\ \hline \\ Y^2 \end{array}\right]$ where C_{L1}, A^1, Y^1 are respectively those portions of C_L, A, and Y related to lines in L1.

Suppose $\Phi(x_{LG}, x_{L1}) = 1$; we want to derive some conditions under which we have $\Phi(x_{LG}, x_{L1}) = 1$, too.

For $x = (x_{LG}, \underline{x}_{L1})$, let $\quad P_G = P_G^0$

$$P_L = \begin{pmatrix} 0 \\ P_L^0 \end{pmatrix}$$

$$\theta = \theta^0$$

and for $x' = (x_{LG}, \bar{x}_{L1})$, let $\quad P_G = P_G^0 + \Delta P_G$

$$P_L = \begin{pmatrix} P_{L1} \\ P_{L2}^0 + \Delta P_{L2} \end{pmatrix}$$

$$\theta = \theta^0 + \Delta\theta$$

we may write:

$\Phi(x) = 1 \Leftrightarrow$

$$\begin{cases} \underline{0} \le P_D + P_G^0 = A\left[\begin{array}{c|c} 0 & \\ \hline & Y^2 \end{array}\right] A^t \theta^0 \le C_G \\ -C_L \le \begin{pmatrix} 0 \\ P_{L2}^0 \end{pmatrix} = \left[\begin{array}{c|c} 0 & \\ \hline & Y^2 \end{array}\right] A^t \\ t_d \le \sum_{i=1}^{N} (P_G^0)_i \le C_{sb} \end{cases} \tag{3.1}$$

Hence, $x' = (x_{LG}, \bar{x}_{L1})$ will be a working state iff

$$\begin{cases} \underline{0} \le P_D + P_G^0 + \Delta P_G = A\left[\begin{array}{c|c} Y^1 & \\ \hline & Y^2 \end{array}\right] A^t(\theta^0 + \Delta\theta) \le C_G \\ -C_L \le \begin{bmatrix} \bar{P}_{L1} \\ P_{L2}^0 + \Delta P_{L2} \end{bmatrix} = \left[\begin{array}{c|c} Y^1 & \\ \hline & Y^2 \end{array}\right] A(\theta^0 + \Delta\theta) \le C_L \\ t_d \le \sum_{i=1}^{N} (P_G^0)_i + \sum_{i=1}^{N} (\Delta P_G)_i \le C_{sb} \end{cases} \tag{3.2}$$

From (3.1) and (3.2) we have $\Phi(x_{LG},\bar{x}_{L1}) = 1$ iff

$$Hw \leq d \tag{3.3}$$

$$H \triangleq \begin{bmatrix} A^1Y^1A^{1t}+A^2Y^2A^{2t} & -I \\ -(A^1Y^1A^{1t}+A^2Y^2A^{2t}) & I \\ 0 & I \\ 0 & -I \\ Y^1A^{1t} & 0 \\ Y^2A^{2t} & 0 \\ -Y^1A^{1t} & 0 \\ -Y^2A^{2t} & 0 \\ 0 & -\Pi^t \\ 0 & +\Pi^t \end{bmatrix}, \quad d \triangleq \begin{bmatrix} -\mu \\ +\mu \\ \bar{g} \\ -\underline{g} \\ \bar{\ell}1 \\ \bar{\ell}2 \\ -\underline{\ell}1 \\ -\underline{\ell}2 \\ \bar{g}_{sb} \\ \underline{g}_{sb} \end{bmatrix}, \quad w \triangleq \begin{bmatrix} \Delta\theta \\ \Delta P_G \end{bmatrix}$$

and $\mu \triangleq A^1Y^1A^{1t}\theta^0$

$\bar{g} \triangleq (C_G - P_G^0)$

$\underline{g} \triangleq -P_G^0$

$\bar{\ell}1 \triangleq C_{L1} - Y^1A^{1t}\theta^0$

$\underline{\ell}1 \triangleq -C_{L1} - Y^1A^{1t}\theta^0$

$\bar{\ell}2 \triangleq C_{L2} - P_L^0$

$\underline{\ell}2 \triangleq -C_{L2} - P_L^0$

$\bar{g}_{sb} \triangleq C_{sb} + \sum_{i=1}^{N} (P_G^0)_i$

$\underline{g}_{sb} \triangleq \sum_{i=1}^{N} (P_N)_i$

Let $\Omega \triangleq \{w \in \mathbb{R}^{2N} \mid Hw \leq d\}$

Thus $\Phi(x_{LG},\bar{x}_{L1}) = 1 \Leftrightarrow \Omega \neq \phi$

From Fact (II-1) we have

$$\Phi(x_{LG},\bar{x}_{L1}) = 1 \Leftrightarrow \forall \lambda \geq 0 \ni \lambda \in N(H^t) \ , \ \lambda^t d \geq 0 \tag{3.4}$$

To characterize all $\lambda \in N(H^t)$ where $\lambda \geq 0$ we construct a set of basis for $N(H^t)$ with positive terms. This can be done by utilizing the special

structure of matrix H. Such a set of basis is obtained and is listed below as the columns of the matrix Λ.

$$\Lambda \triangleq \left[\begin{array}{cccccc}
I & & -T^N & T^P & & \\
I & & T^P & -T^N & & \\
 & I & -T^N & T^P & & II \\
 & I & T^P & -T^N & & \\
 & & I & & & \\
 & & & I & & \\
 & & & & 1 & 1 \\
 & & & & 1 &
\end{array}\right] \tag{3.5}$$

where T^P and T^N are the positive and negative parts of the matrix T for T obtained from the following relation:

$$\begin{bmatrix} Y^1 A^{1t} \\ Y^2 A^{2t} \end{bmatrix} = T^t [A^1 Y^1 A^{1t} + A^2 Y^2 A^{2t}] \tag{3.6}$$

$$T = T^P + T^N \qquad T^P \in \mathbb{R}_+^N \times \mathbb{R}_+^\ell \,, \quad T^N \in \mathbb{R}_-^N \times \mathbb{R}_-^\ell$$

Let Λ_i be the $\underline{i}$th column of Λ where $\Lambda_i \geq 0$. Thus, because of special structure of matrix Λ, we have:

$$\forall \lambda \geq 0 \ni \lambda \in (H^t) \Leftrightarrow \lambda = \sum_{i=1}^{2(N+\ell+1)} a_i \Lambda_i \triangleq \Lambda a \tag{3.7}$$

where

$$a_i \geq 0 \quad \forall i \ni 2N \leq i \leq 2(N+\ell) + 1$$

Hence,

$$d^t\lambda = d^t \begin{bmatrix} -T^N & T^P \\ T^P & -T^N \\ & \\ I & \\ & \\ & I \end{bmatrix} \begin{bmatrix} a_{2N+1} \\ \\ \\ \\ a_{2(N+\ell)} \end{bmatrix} + d^t \left[\begin{array}{ccc|c|c} & & & & \\ \hline I & -T^N & T^P & & II \\ I & T^P & -T^N & & \\ \hline & & & & \\ & & & & \\ & & & 1 & 1 \\ & & & 1 & \end{array}\right] \begin{bmatrix} a_{N+1} \\ \\ \\ \\ a_{2(N+\ell+1)} \end{bmatrix}$$

$$+ d^t \left[\begin{array}{c} I \\ I \\ \hline \\ \\ \\ \end{array}\right] \begin{bmatrix} a_1 \\ \vdots \\ a_N \end{bmatrix}$$

Because of the special structure of vector d, the last term in the above expression is equivalent to zero. And since $\lambda \geq 0$ and considering the definition for d the second term is always positive. Therefore,

$$d^t\lambda = d^t \begin{bmatrix} -T^N & T^P \\ T^P & -T^N \\ & \\ I & \\ & \\ & I \end{bmatrix} \begin{bmatrix} a_{2N+1} \\ \\ \\ \\ a_{2(N+\ell)} \end{bmatrix} \qquad + \text{positive term}$$

Note that this term represents solely the effect of available power generation on the line power flows. If we substitute this term by zero in (3.8) (which actually is equivalent to setting $\Delta P_G = 0$ in (3.3) since $a_{N+1},\ldots,a_{2N}$ are the dual variables corresponding to constraints

on ΔP_G) we obtain the following sufficient condition for $\Omega \neq \phi$:

$$d^t \begin{bmatrix} -T^N & T^P \\ T^P & -T^N \\ & \\ & \\ I & \\ & I \end{bmatrix} \geq 0$$

or equivalently

$$\begin{cases} (-T^N)^t(-\mu) + (T^P)^t(\mu) + \begin{bmatrix} \bar{\ell}1 \\ \bar{\ell}2 \end{bmatrix} \geq 0 \\ (T^P)^t(-\mu) + (-T^N)^t(\mu) + \begin{bmatrix} -\underline{\ell}1 \\ -\underline{\ell}2 \end{bmatrix} \geq 0 \end{cases}$$

which implies that

$$\begin{cases} T^t\mu + \begin{bmatrix} \bar{\ell}1 \\ \bar{\ell}2 \end{bmatrix} \geq 0 \\ T^t\mu + \begin{bmatrix} \underline{\ell}2 \\ \underline{\ell}2 \end{bmatrix} \leq 0 \end{cases} \tag{3.9}$$

We define $P^0_{L1} = Y^1 A^{1t} \theta^0$ as the fake initial power flow in lines L1. Then, from (3.9) we have

$$\left| -T^t\mu + \begin{bmatrix} P^0_{L1} \\ P^0_{L2} \end{bmatrix} \right| \leq C_L \Rightarrow \Phi(x^1) = 1 \tag{3.10}$$

Physical Interpretation of Condition (3.10)

Suppose line $m \in L1$ lies between nodes i_m and j_m, i.e.,

$$A^1 = [A_1|A_2|\ldots|A_m|\ldots A_{\ell 1}] \text{ and } A_m = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \begin{matrix} \leftarrow i_m \\ \leftarrow j_m \end{matrix}$$

If $\mu_m = y_m(\theta^0_{i_m} - \theta^0_{j_m})A_m$, then $\mu = \sum_{m=1}^{\ell 1} \mu_m$ where $\ell 1$ = cardinality of L1.

Let $$\Delta P_L \triangleq -T^t \mu$$

$$= -\sum_{m=1}^{\ell 1} T^t \mu_m$$

Then, based on the definition of matrix T, ΔP_L will be equivalent to the power flow in the network of system $\mathbb{P}$ in the state $x = (\underline{0}, x_{L2}, \bar{x}_{L1})$ and $P_N = \sum_{m=1}^{\ell 1} -\mu_m$ for external power flow sources $p_m = -y_m(\theta_{i_m} - \theta_{j_m})$ between nodes i_m and j_m for $1 \leq m \leq \ell 1$ (Fig III-1).

State $x^1 = (x_G, x_{L2}, \bar{x}_{L1})$ will, thus be a working state if power flow $(\Delta P_L + (P^0_{L1}, P^0_{L2}))$ does not exceed the capacity of the lines where ΔP_L is as defined previously. Such an equivalent model for the effect of lines L1 on the power flow will be referred to as the equivalent line model.

REMARK Sufficient condition (3.10) can be obtained directly from (2.3) by substitutions. However, the approach presented here, using the duality of linear programming for characterization of constraint set, is more general.

3.2. Coherent Power Network Topologies

Power systems are generally not coherent systems. However, for a given network and node injections, the system may be coherent. If, for a given system, the improvement of components guarantees that the line flows will decrease (or at least will not increase), then clearly it is coherent. In this part we will characterize network topologies that possess such a property.

Consider the set of states x for which $AYA^t\theta = P_N$ has a unique solution; let p(x) denote the corresponding line flows.

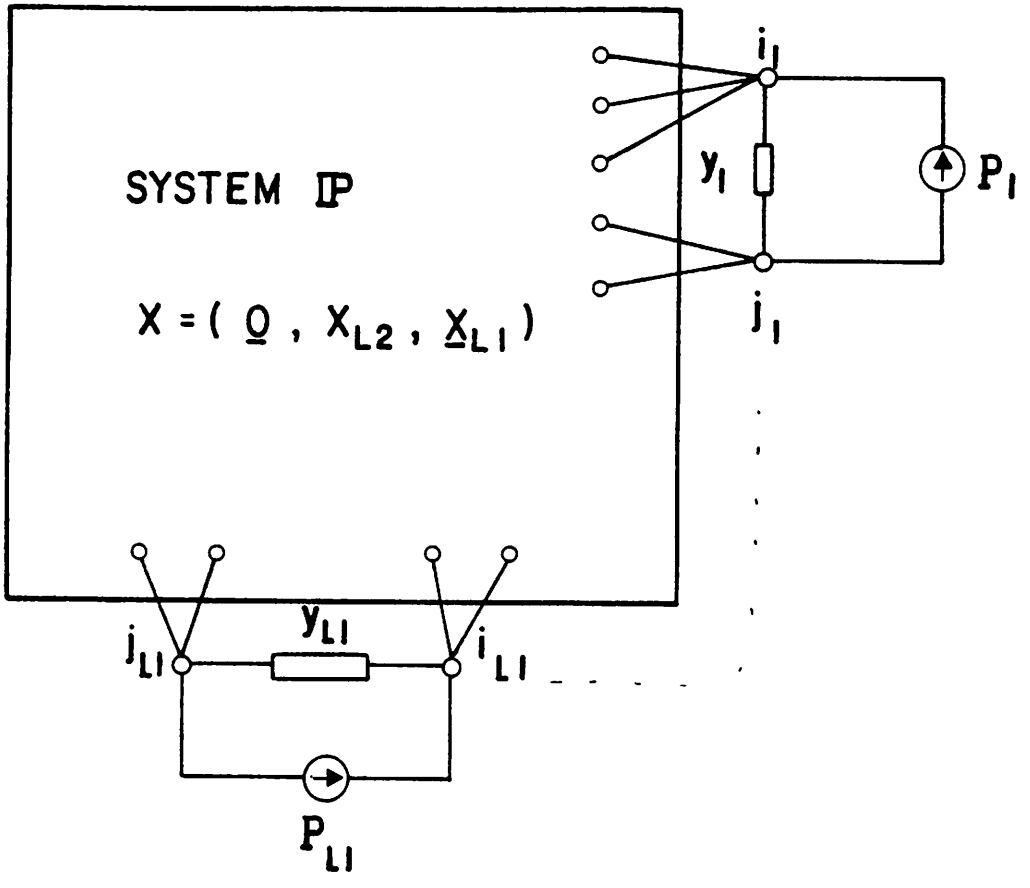


Fig. III-1.

DEFINITION. Given vector P_G such that $\underline{0} \leq P_G \leq C_G$ if

$$x' \geq x \Rightarrow |p_i(x)| \geq |p_i(x')| \quad i \in L_x$$

where L_x is the set of lines which is working for state x, then we say that the corresponding network is a coherent network topology or simply a coherent topology.

Clearly, a parallel system which consists of a load demand and a generator connected by a set of parallel line is a coherent topology. In fact, any network which consists of a series of connecting parallel lines is a coherent topology. Let us call such systems parallel structure systems.

THEOREM (III-1). Parallel structure power systems are the only topologically coherent power systems.

Proof. Consider a power system IP with any topology. Let $(\underline{x}_i,x)$ and $(\bar{x}_i,x) \in X$. If $\Delta P = p(\bar{x}_i-x) - p(\underline{x}_i-x)$, then $(\Delta P)_j$ $j \neq i$ can be obtained from the equivalent line model of part (III-1) (Fig. III-2). Let line i lie between nodes k and m. If θ_k and θ_m are phase angles at nodes k and m for state $(\underline{x}_i,x)$, then $\frac{\mu_i}{y_i} = (\theta_m-\theta_k)$. Let

$\Delta\theta \triangleq$ vector of node phase angles

$\Delta P_{st} \triangleq$ power flow in line (s,t) flowing from node s to node t.

N_j : set of nodes connected to node j by one transmission line.

With no loss of generality, let us assume $\Delta\theta_m \geq \Delta\theta_k$. We then claim that

$$\begin{cases} \text{(a)} & \Delta P_{mj} \geq 0 \quad \forall j \in N_m \\ \text{(b)} & \Delta P_{jk} \leq 0 \quad \forall j \in N_k \end{cases} \tag{3.11}$$

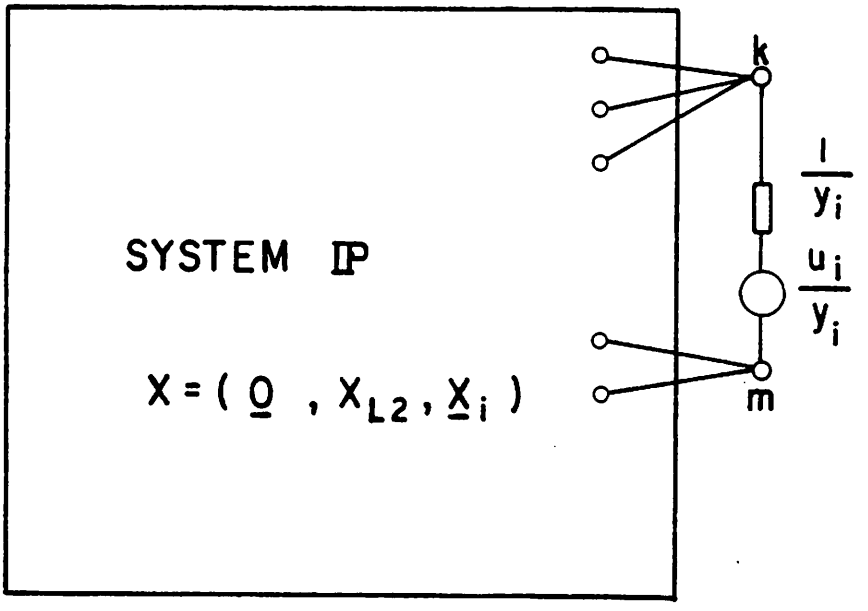


Fig. III-2.

Suppose now (3.11) is false, then $\exists$ some $j \in N_m$

$$\ni \quad \Delta P_{mj} < 0$$

$$\Delta\theta_m < \Delta\theta_j$$

This is not possible since the only source in the network is in branch (k,m), and according to the no gain theorem [6], the phase angles at every node must be smaller than the phase angle at node m. This proves that (3.11a) is true. Similarly, (3.11b) must be true.

On the other hand, if $|p_j(\underline{x}_i,x)| \geq |p_j(\bar{x}_i,x)| \quad j \in L_x$

Then we must have

$$|p_j(\underline{x}_i,x) + (\nabla P)_j| \geq |p_j(\bar{x}_i,x)| \qquad \begin{matrix} j = i,2,\ldots, \\ j \neq i \end{matrix} \tag{3.12}$$

And this is possible iff $p_j(\underline{x}_i,x)$ and $(\Delta P)_j$ are of the opposite sign. Specifically, because of relations (3.11a-b) we must have:

$$\begin{cases} p_{mj} \leq 0 & \forall j \in N_m \\ p_{jk} \geq 0 & \forall j \in N_k \end{cases} \tag{3.13}$$

we claim that relation (3.13) is true only if

$$\begin{cases} \theta_m = \max\limits_{q=1}^{N} \theta_q & \text{(a)} \\ \theta_k = \min\limits_{q=1}^{N} \theta_q & \text{(b)} \end{cases} \tag{3.14}$$

To show this, suppose that (3.13a) is not satisfied, then $\exists$ some node s $\ni$

$$\theta_s = \max_{q=1}^{N} \theta_q \quad s \neq m$$

which implies,

$$p_{st} \geq 0 \qquad \forall t \in N_s$$

Hence, from (3.12) we have

$$\Delta p_{st} \leq 0 \quad \forall t \in N_s$$

or

$$\sum_{t \in N_s} \Delta p_{st} \leq 0$$

This contradicts the Kirchhoff Current Law for power flows. Similarly, (3.14b) must be is true.

Therefore, relation (3.14) implies that only if the improved line lies between nodes with maximum and minimum phase angle can we expect power flow to decrease absolutely. This requires that there be only two nodes in the network. The only network with this property is a parallel system.

The argument can be repeated for each line if there is more than one line to be added to $\mathbb{P}$. Q.E.D.

3.3. Local Coherency

For a fixed demand, the noncoherency property of a power system is a consequence of the effect of line admittances on the distribution of power flows. Let us consider the problem of the effect of all subsets of a specified set of lines, say L1, on the power flow of the system. Let L2 be the set of remaining transmission lines.

DEFINITION. For a subset of lines L1 and a fixed x_G and x_{L2}, we say that system $\mathbb{P}$ is a locally coherent structure if $\forall x = (x_G, x_{L2}, x_{L1})$ and $x_{L1} \in X_{L1}$

either $\Phi(x) = 1$

or $\Phi(x) = 0$

We will refer to the corresponding set $X_{L1}^{GL} = \{x = (x_G, x_{L2}, x_{L1} | x_{L1} \in X_{L1}\}$ as a locally coherent subspace.

Application of Local Coherency

In the following we present an application of the concept of local coherency in identifying subminimal paths and subminimal cuts of the system $\mathbb{P}$.

A: Identification of subminimal paths

Consider a power system $\mathbb{P}$. For any fixed $x_G \ni \Phi(x_G, \bar{x}_L) = 1$, let L1 be a maximal subset of lines such that x_{L1}^{GL} is a locally coherent subspace. If $x_{G'} \leq x_G$ is a minimal vector in X_G such that $\Phi(x_{G'}, \bar{x}_{L2}, \underline{x}_{L1}) = 1$, then $x = (x_{G'L}, \underline{x}_{L1})$ will be a subminimal path vector. The corresponding $\bar{s}_x$ will be a subminimal path set. This, of course, follows from the definition of locally coherent subspaces and the property that power systems are coherent under generation improvement.

Therefore, to identify a subminimal path vector we may take the following steps:

1) pick $x_G \ni \Phi(x_G, \bar{x}_L) = 1$
2) find a maximal L1 $\ni x_{L1}^{GL}$ is a locally coherent subspace
3) find a minimal $x_{G'} \leq x_G \ni \Phi(x_{G'}, \bar{x}_{L2}, \underline{x}_{L1}) = 1$
4) $(x_{G'L}, \underline{x}_{L1})$ is a minimal path vector

B: Identification of subminimal cuts

For a power system $\mathbb{P}$, let $x = (x_G, x_L)$ be a cut vector such that

$$\Phi(x') = 0 \quad \forall x' \in X \ni x' \leq x \tag{a}$$

and

$$\sum_{i=1}^{N} (x_G)_i + C_{sb} \geq 0 \quad \sum_{i=1}^{N} (P_D)_i \tag{b}$$

where condition (b) excludes the trivial subminimal cuts for which total generation capacity is smaller than total load demand. Note that such a state x can be obtained by either connectivity analysis of the graph

of system $\mathbb{P}$ [7], or through network flow analysis [8].

Let $\underline{L1}$ be the set of all transmission lines such that

$$(x_L)_i = \underline{x}_i \qquad i \in \underline{L1}$$

Now, suppose L1 is a maximal subset of $\underline{L1}$ such that X^{GL}_{L1} is a locally coherent subspace of $\mathbb{P}$, i.e.,

$$\forall x = (x_G, \bar{x}_{L2}, x_{L1}) \in X^{GL}_{L1} \qquad \Phi(x) = 0$$

If $x_{G'} \geq x_G$ is a maximal vector in X_G such that $X^{GL}_{x_{L1}}$ is a locally coherent subspaces, then $x = (x_{G'}, \bar{x}_{L2}, \underline{x}_{L1})$ will be a subminimal cut vector and the corresponding $\underline{S}_x$ will be a subminimal cut set. This result follows from the definition of locally coherent subspaces and the property that power systems are coherent under generation improvement.

Therefore, to identify a subminimal cut vector we may take the following steps

1) pick some $x \in X \ni \Phi(x') = 0 \quad \forall x' \leq x$

   this will give the set of failed line, $\underline{L1}$ for state x

2) find L1 a maximal subset of $\underline{L1} \ni X^{GL}_{L1}$ is a L.C. subspace

3) find a maximal $x_{G'} \geq x_G \ni \Phi(x_{G'}, \bar{x}_{L2}, x_{\underline{L1}}) = 0$

4) $(x_{G'}, \bar{x}_{L2}, x_{\underline{L1}})$ is a subminimal cut vector.

Identification of Locally Coherent Subspaces

In order to devise a technique with which to identify locally coherent subspaces of a power system $\mathbb{P}$, we consider the following problem. For a fixed $x_{GL} = (x_G, x_{L2})$, let $x = (x_{GL}, x_{L1}) \in X$ be a working state. Suppose the twofold problem is to determine whether $\Phi(x) = 1$ $\forall x \in X^{GL}_{L1}$ and if not, to find a subset of L1, say L1', such that

$$\Phi(x) = 1 \qquad \forall x = (x_{GL'}, x_{L1'}) \in X_{L1'}^{GL'}$$

where

$$x_{GL'} \triangleq (x_G, x_{L2}, x_{L'})$$

$$L' = L1 - L1'$$

To solve this problem we use the line equivalent model of part (III-1). Thus, the network of Fig. (III-1) will be used to represent the change in the power flow of the existing lines ΔP_L due to the addition of a subset L1' of lines in L1. In this model, for each line $m \in L1'$ we add the equivalent line model between nodes i_m and j_m. Using the Superposition Theorem we may write

$$\Delta P_L = \sum_{m \in L1'} \Delta P_L^m$$

where ΔP_L^m is the power flow change due to the phase angle source $\frac{\mu_m}{y_m}$. Note that ΔP_L^m $m = 1,2,\ldots,\ell 1'$ is obtained by setting all sources equal to zero, with the exception of source $\frac{\mu_m}{y_m}$. An obstacle to solving this problem is that a different network topology is obtained for each subset of L1.

In the following section we will introduce the concept of supply loop so as to find a sufficient condition for locally coherency of a subspace X_{L1}^{GL} and thereby overcome our obstacle.

Let $G(v,k)$ denote the graph of a network with $(v+1)$ nodes and k branches. Thus, the graph of system IP in a state in which all lines except those in set L1 are working will be denoted by $G(N,\ell-\ell 1)$.

THEOREM III-2. Let $G(v,k)$ be a connected unhinged graph. If b_s denotes a branch of this graph (i.e. $b_s \in G(v,k)$). Then there is always a set of $(k-v)$ independent loops such that every loop includes branch b_s.

Proof. Refer to Appendix (III-1).

DEFINITION. Given a connected unhinged graph $G(\nu,k)$ and a branch $b_s \in G(\nu,k)$, any loop which includes branch b_s is called a supply loop of branch b_s. The set of $(k-\nu)$ independent such loops is called an independent set of supply loops of branch b_s.

COROLLARY (III-1). Let $G(\nu,k-k')$ be a connected unhinged graph. Suppose we add the set of branches K' to this graph so as to obtain a connected unhinged graph $G(\nu,k)$. If $b_s \in K'$, then an independent set of supply loops of b_s can be selected such that any supply loop in this set contains only b_s and no other branches in K'.

Proof. Refer to Appendix II-1.

REMARK. Consider a graph $G(\nu,k-k')$, a set of branches K' and the graph $G(\nu,k)$. If $b_s \in K'$, then in what follows if $G(\nu,k-k')$ is a connected unhinged graph we will then always select an independent set of supply loops of b_s as the set of supply loops in Corollary (III-1).

Let us now apply the results obtained above in determining whether X_{L1}^{GL} is a locally coherent subspace. However, note that if the network of system $\mathbb{P}$ for states $(\cdot, x_{L2}, \bar{x}_{L1})$ is not a connected graph, we may treat each subnetwork separately. In addition, the graph must not be unhinged for if it is, we can partition the graph into unhinged subgraphs and again solve the problem for each subgraph separately.

Suppose $G_d(N,\ell-\ell 1)$ is the directed graph of system $\mathbb{P}$ for state $(x_{GL}, \underline{x}_{L1})$ where the direction of each branch agrees with the power flow in that branch. Let B^m be the supply loop matrix of a particular set of supply loops of branch $m \in L1$, and let P^m be the respective supply loop power flows. The total power flow change due to all lines in $L1 \subseteq L1$,

therefore, will be

$$\Delta P_L \triangleq \begin{pmatrix} \Delta P_{L1} \\ \Delta P_{L2} \end{pmatrix} = \sum_{m \in L1'} \Delta P_L^m = \sum_{m \in L1'} (B^m)^t p^m \tag{3.15}$$

Note that p^m assumes different values for various subsets of L1.

For state $(x_{GL}, \underline{x}_{L1})$, let the power flow be

$$P_L^0 = \begin{pmatrix} \underline{0} \\ P_{L2}^0 \end{pmatrix} = \begin{pmatrix} \underline{0} \\ P_{L2}^{0+} \end{pmatrix} + \begin{pmatrix} \underline{0} \\ P_{L2}^{0-} \end{pmatrix}$$

where P_{L2}^{0+} and P_{L2}^{0-} denote the positive and negative parts of P_{L2}^0 respectively. As before, let P_{L1}^0 be

$$P_{L1}^0 = Y^1 A^1 t_\theta{}^0 = P_{L1}^{0+} + P_{L1}^{0-}$$

Let us also select the directions of branches in L1 to be the same as their respective power flows in P_{L1}^0.

Hence, $\Phi(x) = 1$ for some $x \in X_{L1}^{GL}$ if

$$\left| (P_{L1}^0 + \Delta P_{L1}(x))_i \right| \leq (C_{L1})_i \qquad \forall i \in L1'$$

$$\left| (P_{L2}^0 + \Delta P_{L2}(x))_j \right| \leq (C_{L2})_j \qquad \forall j \in L2$$

Then, $\Phi(x) = 1 \quad \forall x \in X_{L1}^{GL}$ if

$$\left\{ \begin{array}{l} \begin{bmatrix} P_{L1}^{0+} + \max(\Delta P_{L1}^+) \\ P_{L2}^{0+} + \max(\Delta P_{L2}^+) \end{bmatrix} \leq \begin{bmatrix} C_{L1} \\ C_{L2} \end{bmatrix} \\ \text{and} \\ - \begin{bmatrix} P_{L1}^{0-} + \min(\Delta P_{L1}^-) \\ P_{L2}^{0-} + \min(\Delta P_{L2}^-) \end{bmatrix} \leq \begin{bmatrix} C_{L1} \\ C_{L2} \end{bmatrix} \end{array} \right. \tag{3.16}$$

where $\max(\Delta P_L^+)_i = \max_{x \in X_{L1}^{GL}} (\Delta P_L^+(x))_i \qquad i = 1,2,\ldots,\ell$

abd $\min(\Delta P_L^-)_i = \min_{x \in X_{L1}^{GL}} (\Delta P_L^-(x))_i \qquad i = 1,2,\ldots,\ell$

Condition (3.16) is a sufficient condition for X_{L1}^{GL} to be a locally cohernet subspace. We may write

$$\begin{cases} B^m = B^{m+} + B^{m-} \\ p^m = p^{m+} + p^{m-} \end{cases} \tag{3.17}$$

Hence,

$$\Delta P_L^m = [(B^{m+})^t p^{m+} + (B^{m-})^t p^{m-}] + [(B^{m-})^t p^{m+} + (B^{m+})^t p^{m-}]$$

or

$$\begin{cases} \Delta P_L^{m+} = (B^{m+})^t p^{m+} + (B^{m-})^t p^{m-} \\ \Delta P_L^{m-} = (B^{m-})^t p^{m+} + (B^{m+})^t p^{m-} \end{cases} \tag{3.18}$$

Let p_{max}^{m+} and p_{min}^{m-} be such that $\forall i = 1,2,\ldots,\ell$

$$\begin{cases} (p_{max}^{m+})_i = \max_{x \in X_{L1}^{G'L}} (p^{m+}(x))_i \\ (p_{min}^{m-})_i = \min_{x \in X_{L1}^{G'L}} (p^{m-}(x))_i \end{cases} \tag{3.19}$$

$$X_{L1}^{G'L} = \{x | x = (x'_G, x_{L2}, x_{L1}),\ x_{L1} \in X_{L1} \text{ and } x'_G \text{ is the sources of } x_{L1}\}$$

Then, from (3.18) it follows

$$\begin{cases} \Delta P_L^{m+} \leq (B^{m+})^t p_{max}^{m+} + (B^{m-})^t p_{min}^{m-} \\ \Delta P_L^{m-} \geq (B^{m-})^t p_{max}^{m+} + (B^{m+})^t p_{min}^{m-} \end{cases} \tag{3.20}$$

or from (3.15) it follows

$$\begin{cases} \max(\Delta P^{+}_{L1}) \leq \sum\limits_{m\in L1} ((B^{m+})^{t} p^{m+}_{max} + (B^{m-})^{t} p^{m-}_{min}) \\ \min(\Delta P^{-}_{L1}) \geq \sum\limits_{m\in L1} ((B^{m-})^{t} p^{m+}_{max} + (B^{m+})^{t} p^{m-}_{min}) \end{cases} \tag{3.21}$$

Inequality (3.21) and constraint (3.16) provide the following sufficient conditions for locally coherency of subspace X^{GL}_{L1}:

$$\begin{cases} P^{max}_{L} \triangleq [P^{0+}_{L} + \sum\limits_{m\in L1} ((B^{m+})^{t} p^{m+}_{max} + (B^{m-})^{t} p^{m-}_{min})] \leq C_L \\ P^{min}_{L} \triangleq -[P^{0-}_{L} + \sum\limits_{m\in L1} ((B^{m-})^{t} p^{m+}_{max} + (B^{m+})^{t} p^{m-}_{min})] \leq C_L \end{cases} \tag{3.22}$$

Note that because of the choice of the branch directions in (3.22), we have $P^{0-}_{L} = 0$. Therefore, to verify the validity of constraint (3.22), we need to evaluate p^{m+}_{max} and p^{m-}_{min} $\forall m \in L1$. Before we proceed to solve the problem of evaluating p^{m+}_{max} and p^{m-}_{min}, we will examine the second part of the original problem. That is, if condition (3.22) does not hold, then a set $L1' \subset L1$ must be found such that $X^{GL'}_{L1'}$ is locally coherent subspace.

Suppose now that (3.22) is not satisfied. Let i and j be lines in the network of state $(\cdot, x_{L2}, \bar{x}_{L1})$ such that

$$(P^{max}_{L} - C_L)_i = \max_{k\in L} (P^{max}_{L} - C_L)_k \geq 0$$

$$(P^{min}_{L} - C_L)_j = \min_{k\in L} (P^{min}_{L} - C_L)_k \geq 0$$

Note that if (3.22) is not satisfied, at least one of such i and j will exist. Intuitively, i and j are the lines which are expected to be saturated the most. If i or j $\in$ L1 then we should remove these lines from set L1 to obtain L1'. If i or j share a node with some line in L1 we must remove that line in L1 from the network. And finally, if there are not such i and j in L1, then we may remove line q $\in$ L1 with maximum

related source i.e.,

$$\left(\frac{\mu_q}{y_q}\right) = \max_{j \in L1} \left(\frac{\mu_j}{y_j}\right)$$

Now if for the new subset of lines L1', the conditions of Corollary (III-1) hold, then there is no need for additional computation except to substitute L1' for L1 in (3.22). However, if these conditions are not satisfied, removal of lines i and j, if they belong to L1, may destroy some of the supply loops of the other branches in L1. In this case, the respective maximum and minimum power flows of these loops must be set equal to zero. We may repeat the selection of subsets of L1 as explained above until (3.22) is satified for some $L1' \subset L1$. Hence, $X^G_{L1'}$ will be a locally coherent subspace.

Determinization of p^{m+}_{max} and p^{m-}_{min}

We present some preliminary results on the computation of p^{m+}_{max} and p^{m-}_{min} $m = 1,2,\ldots,\ell 1$. For a system P with any topology, a crude upper bound on p^{m+}_{max} and $(-p^{m-}_{min})$ $m = 1,2,\ldots,\ell 1$ can easily be obtained as done below.

For a supply loop i in the set of independent supply loops of line $m \in L1$, suppose L^m_i denotes the set of lines in this loop. Thus, we will have: for $i = 1,2,\ldots,\ell-N$

$$\begin{cases} (p^{m+}_{max})_i \leq \left| \dfrac{\frac{\mu_m}{y_m}}{\sum_{j \in L^m_i} \frac{1}{y_j}} \right| \\ (-p^{m-}_{min})_i \leq \left| \dfrac{\frac{\mu_m}{y_m}}{\sum_{j \in L^m_i} \frac{1}{y_j}} \right| \end{cases} \qquad (3.23)$$

More study is being done to develop a method for determining exact values of p^{m+}_{max} and p^{m-}_{min} although some results have been reached on computing bounds on these quantities other than those given by (3.9). However, for a power system $\mathbb{P}$ whose network topology in state $(\cdot, x_{L2}, \bar{x}_{L1})$ is of ladder shaped, we can easily evaluate p^{m+}_{max} and p^{m-}_{min}. For such systems, if we select the supply loop directions the same as the flow in the line m, then $p^{m-}_{min} = \underline{0}$. For any $i \in L1$, $(p^{m+}_{max})_i$ can be obtained by removing all lines in L1 except for those in L^m_i.

APPENDIX (III-1)

THEOREM III-2

Proof. Let T be a tree of $G(\nu,k)$ such that $b_s \in T$. Let L_t denote the set of links associated with T. For $G(\nu,k)$ the following holds.

(i) It follows from the definition of connected unhinged graph that there are at least two independent paths between any pair of nodes.

(ii) Let $b^t_{ij} \in T$ be a branch that lies between nodes i and j. Let C_{ij} be the corresponding fundamental cut set. If $\exists$ k links in C_{ij}, then we can find k independent loops associated with these links such that all of these loops include b^t_{ij}. This is so because $\forall$ links $b_{mn} \in C_{ij}$, $\exists$ two independent tree paths between (m,i), and (n,j), respectively.

We use the above facts to prove the existence of the $(k-\nu)$ independent loops by constructing them. Let L be the set of such loop which is initially empty. Suppose $\exists$ m links in C_{b_s} where from (i), $m \geq 1$. Hence, from (ii) we can define m independent loops all of which include b_s. If $m = k-\nu$, we are done. Otherwise there are $m < k-\nu$ independent loops

in L and there are $(k-\nu-m)$ links for which no loop is defined. Let the graph G' be a subgraph of $G(\nu,k)$ which is defined by all the m independent loops in L. Clearly, G' is connected and unhinged. By definition, $\exists$ a tree path between any pair of nodes inside of G' and this path includes b_s. Let $(G-G')$ be the subgraph of G which results from the removal of branches in G'. Hence, $(G-G')$ must consist of a family of connected subgraphs $G \triangleq \{ G_1, G_2, \ldots, G_p \}$, for some finite p where each subgraph in G has at least two nodes in common with G'. Consider $G_n \in G$, and let (e,f) be a pair of nodes shared by G' and G_n. There is a tree path p^t_{ef} between nodes e and f in G', and since $e,f \in G_n$, $\exists$ a path p^n_{ef} between e and f in G_n. Hence, there must be a link in path p^n_{ef} in G_n. Let b^t_{ij} be a branch of p^t_{ef} and let C'_{ij} be that subset of links in C_{ij} which does not belong to G'. As a result no independent loop has been defined which includes any of the links in C'_{ij} yet. Let $b^{\ell}_{km} \in C'_{ij}$. There is a tree path p^t_{ik} between (i,k) and a tree path p^t_{jm} between (j,m). In addition there is a path p_{ij} in G' which includes b_s. Hence, a loop which includes b^t_{ij}, p^t_{ik}, p^t_{jm} and p_{ij} can be defined. Then there will be $(m+1)$ loops in L and $(k-\nu-m-1)$ links will be left which are not in any of the loops in L. Of course, p_{ij} may include some links in L_t. However, we have already defined an independent loop for each of them since they belong to G'. Add the new loop to G' and remove it from G_n to define new graphs G' and G_n. Then G' and graphs in $G = \{G_1, \ldots, G_n, \ldots, G_p\}$ have the same property as before. Repeat the above procedure until each subgraph in G vanishes. By construction each loop corresponds to one link in L_t, i.e., each loop in L is defined for a specific link in L_t. Hence, when all subgraphs in G vanish there will be $(k-\nu)$ loops in L such that each of them, inlcudes the branch b_s. These loops are independent because, by construction, no two of them share the same subset

of links in L_t. Q.E.D.

COROLLARY (III-1)

Proof. Consider graph $(\nu,k-k'+1)$ which is the union of $G(\nu,k-k')$ and $b_s \in K'$. By Theorem (III-2) we can define the set of $(k-k'-\nu)$ independent supply loops of b_s. The union of these loops gives $G(\nu,k-k'+1)$ and none of these loops include any branch in K' except b_s. Now, for any branch $b \in K'$ we must construct a supply loop of b_s such that except branch b all other branches of this loop belong to $G(\nu,k-k'+1)$. Such a loop can be constructed by the following precedure; for any $b \in K'$, consider the graph obtained from the union of $G(\nu,k-k'+1)$ and b. By Theorem (III-2) we can define a supply loop of b_s which includes b. Such a loop includes only b and b_s of the set K'. Hence, by repeating the procedure for all branches in K', we obtain the set of independent loops we are looking for. Q.E.D.

IV. A NEW METHOD FOR POWER SYSTEM RELIABILITY EVALUATION

Because of the noncoherency and nonlinearity of our model as explained in sections II and III, the set of working states W does not yield a simple analytic characterization. In this section we present the basic idea of a new scheme for identifying the set W by successively obtaining subsets of W.

4.1. Successive Method for Power System Reliability Evaluation

Consider a power system IP modelled by relations (2.1a/f). These relations represent the KCL on real power, KVL on voltage phase angles and the Ohm's Law between real power and phase angles together with component capacity constraints. Let each of these constraints be expressed separately. The component capacity constraint and KCL can be written as:

$$\begin{cases} -C_L \leq P_L \leq C_L \\ \underline{C} \leq A' P_L \leq \bar{C} \end{cases} \tag{4.1}$$

where

$$A' = \begin{bmatrix} A \\ -\Pi\, {}^{t}A \end{bmatrix}, \quad \bar{C} = \begin{bmatrix} C_G' + P_D \\ G_{sb} \end{bmatrix}, \quad \underline{C} = \begin{bmatrix} P_D \\ 0 \end{bmatrix}$$

After eliminating the redundant constraints. The reduced form of (4.1) may be expressed as follows

$$\begin{cases} (-C_L)_i \leq (P_L)_i \leq (C_L)_i & i \in R_L \\ (\underline{C})_j \leq (A')_j P_L \leq (\bar{C})_j & j \in R_N \end{cases} \tag{4.2}$$

where R_L is the set of lines and R_N is the set of nodes for which respectively, their capacity and power flow constraints together form a nonredundant set of constraints.

In order to present the basic idea of the successive method, let us make the following assumption.

ASSUMPTION (IV-1). The generators in system $\mathbb{P}$ are such that there is a working generator at every node in set R_N. As a result of this assumption, the following holds

$$(\underline{C})_j \neq (\bar{C})_j \quad j \in J$$

Let us define set C as follows

$$C \triangleq \{P_L \in \mathbb{R}^{\ell} | P_L \text{ satisfies (4.2)}\}$$

Hence, set C is a nonempty convex polyhedral (there is at least one working state in X).

Let us define the vector of phase angles across transmission lines δ as

$$\delta \triangleq A^t\theta \tag{4.3}$$

Then, for system $\mathbb{P}$, KVL can be expressed by the following linear manifold:

$$B\delta = 0 \tag{4.4}$$

where $B \in \mathbb{R}^{\ell-N} \times \mathbb{R}^{\ell}$ is a fundamental loop matrix. Finally, Ohm's Law can be written as

$$P_L = Y\delta \tag{4.5}$$

Let us now assume that there is at least one line which is always working. Let L2 denote the set of lines that are always working. Hence, L2 is not empty by assumption. We claim that it is possible to select

the fundamental loops corresponding to B such that every loop includes at least one line of set L2. This is so because we can choose the independent supply loops of one line in L2. Let L1 be the remaining set of lines in the system; B and δ can be partitioned accordingly:

$$B = [B^1 \; B^2]$$

$$\delta = \begin{bmatrix} \delta^1 \\ \delta^2 \end{bmatrix}$$

Hence, (4.4) can be written as

$$B^1\delta^1 = -B^2\delta^2 \tag{4.6}$$

Note that as a result of the assumption we just made, every row of B^2 has a nonzero element.

Relations (4.2,5,6) represent system IP for all possible states. The problem of identifying the set W, then, can be stated as: given a set $C \subseteq \mathbb{R}^{\ell}$ and the set of lines L2, find all network state for which there is a $P_L \in C$ that satisfies (4.5) for some δ in the linear manifold of (4.6). Let us refer to this formulation as problem (4.7).

In order to solve this problem we present the following method which involves successive approximation of a set in $\mathbb{R}^m$ m = 1,2,..., by a sequence of hyperboxes.

Let us define disjoint hyperboxes H_c^i inside C as

$$H_c^i = \{P_L \in C | \underline{u}_i \leq (P_L)_j \leq \bar{u}_j \quad j = 1,\ldots,\ell, \; \underline{u}_j \neq \bar{u}_i\}$$

Let HC be the set of such hyperboxes. The set HC is to be constructed so that in the limit it covers C. Let us consider the special problem of (4.7) when C is substituted by a hyperbox $H_c^i \in$ HC. Let

$$H_c^i = \{P_L \in C | \underline{u}_j \leq (P_L)_j \leq \bar{u}_j \quad j = 1,2,\ldots,\ell\}$$

we may write the power flow as

$$P_L = \begin{bmatrix} P_L^1 \\ P_L^2 \end{bmatrix} = \left[\begin{array}{c|c} Y^1 & \\ \hline & Y^2 \end{array}\right] \begin{bmatrix} \delta^1 \\ \delta^2 \end{bmatrix}$$

For any $P_L \in H_c^i$ we have

$$\underline{u}^2 \leq P_L^2 = Y^2\delta^2 \leq \bar{u}^2$$

or

$$(Y^2)^{-1}\underline{u}^2 \leq \delta^2 \leq (Y^2)^{-1}\bar{u}^2 \tag{4.8}$$

It follows from (4.6)

$$\underline{d} \leq B^1\delta^1 \leq \bar{d} \tag{4.9}$$

where $\underline{d} = \min(B^2\delta^2)$ and $\bar{d} = \max(B^2\delta^2)$. In Appendix (IV.1) it is shown that $\underline{d}_k \neq \bar{d}_k \quad k = 1,2,\ldots,N$. Let set D^i be defined as

$$D^i = \{\delta^1 \in \mathbb{R}^{\ell 1} | \underline{d} \leq B^1\delta^1 \leq \bar{d}\}$$

By definition, for any $\delta^1 \in D^i$ there is a $\delta^2 \in \mathbb{R}^{\ell-\ell 1}$ such that (δ^1,δ^2) is in the linear manifold of (4.6) and also $P_L^2 = Y^2\delta^2$ satisfies the line capacity constraint. Then, the problem (4.7) will be reduced to finding all values of Y^1 for which there is a $\delta^1 \in D^i$ and a $P_L' \in \mathbb{R}^{\ell 1}$ such that

$$\underline{u}_i^1 \leq (P_L^1)_i \leq \bar{u}_i^1 \tag{4.10}$$

and

$$P_L^1 = Y^1\delta^1$$

Equation (4.10) may be tested for disjoint hyperboxes in D^i as follows.

Let hyperboxes H_d^j in D^i be defined as

$$H_d^j = \{\delta^1 \in D^i | \underline{h} \leq \delta^1 \leq \bar{h}\} \tag{4.11}$$

The set of hyperboxes H_d^j $j = 1,2,\ldots$ is to be constructed so that in the limit it covers D^i. Consider the following problem for some $H_d^j \in HD$:

Given H_c^i and H_d^j as above, what are the values of Y^1 for which $\exists$ some $\begin{cases}\delta^1 \in H_d^j \\ P_L \in H_c^i\end{cases}$ such that

$$(P_L^1)_i = (Y^1)_i(\delta^1)_i \qquad \forall i \in L1$$

If neither $(Y^1)_i = 0$ nor $(Y^1)_i = y_n$ satisfies the above relation for some $i \in L1$, then there is not any Y^1 for which we can find some $P_L \in H_c^i$ and $\delta^1 \in H_d^j$ which together satisfy network constraints. However, if any of values $(Y^1)_i = 0$ and $(Y^1)_i = y_i$ or both satisfy the above conditions for all $i \in L1$, then the respective values for Y^1 together with Y^2 give a subset of working states. If we repeat this successively for all $H_c^i \in HC$ and $H_d^j \in H_D$ we obtain all network configurations Y for which the respective state is in W.

To summarize, the procedures of identifying W are as follows. Given system $\mathbb{P}$,

1) determine set C
2) select set of hyperboxes HC
3) Given $H_c^i \in HC$ determine set D^i
4) select set of hyperboxes HD^i
5) Given $H_d^j \in HD^i$ find all values of Y^1 for which
   $\exists$ $P_L \in H_c^i$ and $\delta^1 \in H_d^i \ni P_L^1 = Y^1\delta^1$

6) repeat 5 for all $H_d^j \in HD$

7) repeat 3 to 6 for all $H_c^i \in HC$

EXAMPLE III-1

Consider the power network in Fig. (IV-1).

Suppose Generator 1 and 2 are always working and

$$\begin{pmatrix}0\\5\end{pmatrix} \leq P_G \leq \begin{pmatrix}10\\15\end{pmatrix}$$

Let $P_D = 20$, $L2 = \{2\}$, $C_L = (15,5,5)$ and $y = (1,2,1)$.

STEP 1. The set C will be defined by the following constraints.

$$\begin{cases} 10 \leq (P_L)_1 + (P_L)_3 \leq 20 \\ -15 \leq -(P_L)_1 + (P_L)_2 \leq -5 \\ -15 \leq (P_L)_1 \leq 15 \\ -5 \leq (P_L)_2 \leq 5 \\ -5 \leq (P_L)_3 \leq 5 \end{cases}$$

STEP 2

Let the hyperboxes H_c^1, H_c^2, $H_c^3 \ni H_c^i \subset C$ be as follows:

$$H_c^1 = \left\{ P_L \in \mathbb{R}^3 \;\middle|\; \begin{array}{r} 12.5 \leq (P_L)_1 \leq 15 \\ 0 \leq (O_L)_2 \leq 5 \\ -2.5 \leq (P_L)_3 \leq 5 \end{array} \right\}$$

$$H_c^2 = \left\{ P_L \in \mathbb{R}^3 \;\middle|\; \begin{array}{r} 10 \leq (P_L)_1 \leq 12.5 \\ -2.5 \leq (P_L)_2 \leq 5 \\ 0 \leq (P_L)_3 \leq 5 \end{array} \right\}$$

$$H_c^3 = \left\{ P_L \in \mathbb{R}^3 \;\middle|\; \begin{array}{r} 7.5 \leq (P_L)_1 \leq 10 \\ -5 \leq (P_L)_2 \leq 2.5 \\ 2.5 \leq (P_L)_3 \leq 5 \end{array} \right\}$$

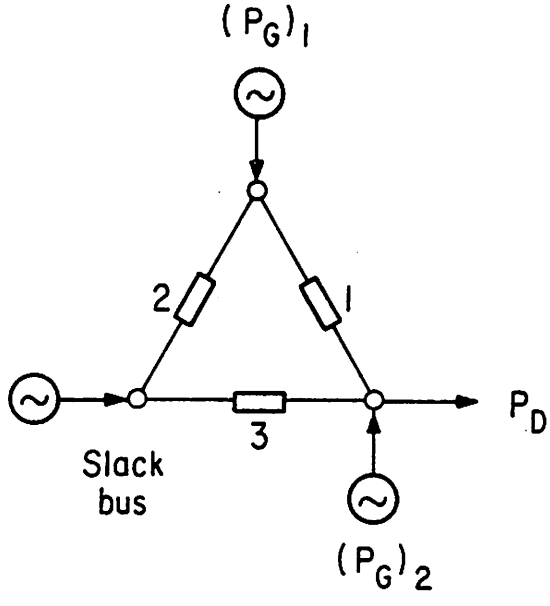


Fig. IV-1.

STEP 3

Consider H_c^1, we have:

$$0 \leq (P_L)_2 \leq 5$$

Hence,

$$0 \leq \delta_2 \leq 2.5$$

which gives

$$D^1 = \left\{ \begin{pmatrix} \delta \\ \delta_3^1 \end{pmatrix} \middle| 0 \leq -\delta_1 + \delta_3 \leq 2.5 \right\} \qquad (1)$$

STEP 4,5,6,

From the definition of H_c^1 we have

$$12.5 \leq (P_L) \leq 15$$

Consequently, we must have $(Y)_1 = y_1 \neq 0$ which implies:

$$12.5 \leq \delta_1 \leq 15$$

This requires

$$12.5 \leq \delta_3 \leq 17.5$$

or

$$12.5 \leq \delta_3 \leq 17.5$$

which does not satisfy the constraint on $(P_L)_3$ for H_c^1. Hence, for H_c^1 there is no solution.

NOTE. Since H_c^1 is such that the upper and lower bounds on $(P_L)_1$ have the same sign for H_c^1, line 1 must be working. This easily gives constraint on flow in line 3. Clearly, there is no need for selection of hyperboxex HD^1.

STEP 7

If we repeat the procedure for H_c^2, it reveals that for $Y = \begin{bmatrix} y_1 & & \\ & y_2 & \\ & & 0 \end{bmatrix}$, i.e., when line 3 is failed, the system will be working.

also, repeating the procedure for H_c^3 reveals that when $Y = \begin{bmatrix} y_1 & & \\ & y_2 & \\ & & y_3 \end{bmatrix}$, i.e., when all lines are working, the system is working.

Therefore, since the union of hyperboxes H_c^i $i = 1,2,3$ is a proper subset of C, i.e., $\bigcup_{i=1}^{3} H_c^i \subset C$, the set of working states obtained is a subset of W for $\mathbb{P}$.

REMARK If assumption (IV-1) is not satisfied then in order to take into account generator outages we may substitute system $\mathbb{P}$ by an augmented system $\mathbb{P}^a$. System $\mathbb{P}^a$ is obtained by substituting any generator i for which $\bar{x}_i = (C_G)_i$ and $\underline{x}_i = 0$ with a transmission line and a generator in series. For this equivalent network the capacity of the generator is $(C_G)_i$ and the generator is always working. However the transmission line may fail with the same outage rate as the generator i. The admittance of this line can be selected as any positive value and its capacity limit must be larger than $(C_G)_i$.

Consider system $\mathbb{P}^a$, since Assumption (IV-1) is not satisfied, the set C will be a subset of a linear manifold in $\mathbb{R}^{\ell}$. Hence, we cannot successively approximate C by a sequence of hyperboxes contained in C. However, we may select a set of disjoint hyperboxes whose union covers the set C to approximate the set. This method is, of course expected to be less efficient than the method presented for the case when Assumption (IV-1) is satisfied.

4.2. Present and Projected Works

In connection with the results obtained in this study, we are presently engaged in studying the following topics:

(a) Efficient methods for identifying locally coherent subspaces.

(b) Methods for approximating a set in $\mathbb{R}^n$ = 1,2,... by a sequence of hyperboxes.

(c) Developing an efficient method for identifying the set W by combining the concept of local coherency and the direct method introduced in section IV.

APPENDIX (IV-1)

We prove $\underline{d}_i \neq \bar{d}_i$ in relation (4.9). Let us partition B^2 into its positive and negative parts, i.e.,

$$B^2 = (B^2)^+ + (B^2)^-$$

Then, from (4.8) it follows:

$$\begin{cases} \bar{d} = \max(B^2\delta^2) = (B^2)^+ (Y^2)^{-1}\bar{u}^2 + (B^2)^- (Y^2)^{-1}\underline{u}^2 \\ \underline{d} = \min(B^2\delta^2) = (B^2)^+ (Y^2)^{-1}\underline{u}^2 + (B^2)^- (Y^2)^{-1}\bar{u}^2 \end{cases}$$

for the sake of contradiction suppose there is some $i \in \{1,2,\ldots,N\}$ such that $\underline{d}_i = \bar{d}_i$, then we must have:

$$(B^2)^+_i (Y^2)^{-1}\bar{u}^2 + (B^2)^-_i (Y^2)^{-1}\underline{u}^2 = (B^2)^+_i (Y^2)^{-1}\underline{u}^2 + (B^2)^-_i (Y^2)^{-1}\bar{u}^2$$

which implies:

$$[(B^2)^+_i - (B^2)^-_i](Y^2)^{-1}(\bar{u}^2 - \underline{u}^2) = 0$$

but, we have $\bar{u}^2 > \underline{u}^2$; hence $\underline{d}_i = \bar{d}_i$ is impossible.

References

[1] Proceedings of E.P.R.I. Workshop on Power System Reliability - Research Needs and Priorities, Pacific Grove, California, March 5-9, 1979.

[2] R. E. Barlow and F. Proschan, Statistical Theory of Reliability and Life Testing, Holt, Rinehart and Winston, 1975.

[3] R. T. Rockafellar, Convex Analysis, Princeton University Press, 1970.

[4] J. D. Esary. F. Proschan and D. W. Walkup, "Association of random variables, with applications," Ann. Math. Statistics, vol. 38, pp. 1466-1474, 1967.

[5] W. I. Zangwill, Nonlinear Programming, A Unified Approach, pp. 47-53, Prentice-Hall, 1969.

[6] A. Talbot, "Some fundamental properties of networks without mutual inductance," Proc. IEEE (London), vol. 102, pp. 168-175, Sept. 1955.

[7] Z. G. Todd, "A probability method for transmission and distribution outage calculations," IEEE Trans., vol. PAS-83, pp. 695-701, July 1964.

[8] P. Doulliez and E. Jamoulle, "Transportation networks with random arc capacities," Revue Francaise d'Automatique, Informatique de Recherche Operationnelle, vol. 3, pp. 45-59, Nov. 1972.

[9] W. Feller, An Introduction to Probability Theory and Its Applications, vol. I, 3rd ed., pp. 98-101, John Wiley and Sons, 1968.

Acknowledgement

I wish to express my gratitude and appreciation to Professor Felix Wu for his suggestion of the topic of this research and for his continuous guidance and help during the course of this work.

I would also like to thank Doris Simpson for her help in typing this manuscript.