RELIABILITY ANALYSIS AND OPTIMIZATION OF COMPLEX SYSTEMS

A Dissertation

Presented to

the Faculty of the Department of Chemical Engineering

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy in Chemical Engineering

---

by

Satish Loonkaran Gandhi

December, 1973

ACKNOWLEDGMENT

I would like to express my gratitude to Dr. Ernest J. Henley for his friendly guidance and encouragement throughout the course of this dissertation research, and to Dr. Koichi Inoue for his interest and many useful suggestions. A special debt of gratitude is due my parents, Mr. and Mrs. L. D. Gandhi, for their constant encouragement and support in my efforts to seek advanced studies. I would also like to thank Dr. R. L. Motard and the staff of the Engineering Systems Simulation Laboratory for their assistance in the use of the computer facilities. The financial support provided by the Office of Naval Research under ONR Contract N0014-68-A-0151 and the National Science Foundation Grant CK 34542 are gratefully acknowledged. Special thanks are due to Mrs. H. J. Stover for her expert typing of the manuscript. Lastly, I wish to thank my wife for her love and understanding during these times.

RELIABILITY ANALYSIS AND OPTIMIZATION OF COMPLEX SYSTEMS

An Abstract of a Dissertation

Presented to

the Faculty of the Department of Chemical Engineering

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy in Chemical Engineering

---

by

Satish Loonkaran Gandhi

December, 1973

ABSTRACT

Most of the earlier literature on system reliability optimization consider only the simple series-parallel systems subject to one or two constraints. Practical systems have complex rather than the simple series parallel configurations. With a view towards solving these complex system reliability optimization problems, an efficient computer algorithm based on the path enumeration method has been developed. An important feature of the method is the module representation of the reliability graph which considerably simplifies the calculation of reliability and sensitivity functions of complex systems. A modified integer gradient method is used for system optimization. Although the method does not insure a global optimum, it does find various near-optimum solutions. From a practical consideration, this could provide for a wider choice during the design phase.

In this research an effort has also been made to apply basic reliability concepts to process plants. A new formulation of the optimal reliability design of process plants which takes into account the quantitative aspects of systems throughput is proposed. It is based on the k-out-of-n configuration instead of the conventional parallel redundancy configuration. The problem is so formulated that determining the optimum configuration also determines the optimum capacity of units to be used at each stage of the system. A computer program based on a pseudo-Boolean algorithm is used to solve this non-linear integer programming problem.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

Considerable literature has appeared in the last two decades in the area of system reliability analysis and optimization, particularly by researchers in electrical engineering and operations research. However, there are two areas where very little progress has been made.

Most of the earlier literature on system reliability optimization consider only the simple series-parallel systems subject to one or two constraints. The reason only this particular system has received so much attention is the separability and concavity of the reliability function. This makes it amenable to be solved using the techniques of dynamic programming [6,23,30,33,38,43,61], discrete maximum principle [19,55], integer programming [24,41,44,54,56], Lagrangaian multipliers [2,3,18,38,40], and the dominating sequence concept [3,8,30,38,48]. Practical systems have complex rather than the simple series-parallel configurations, and, therefore, require methods capable of solving complex system optimization problems.

In the design of electrical, electronic, and safety systems, the flow of information is the most important factor, whereas in process systems the flow of materials and energy is also very important. Therefore, some innovations and modifications are required before the previous research in the area of reliability engineering can be effectively applied to chemical process systems. The vital role reliability plays in the operation of today's large, complex and expensive chemical plants has been widely recognized [10,17,26,35,58].

In this research an effort has been made to apply basic reliability concepts to process plants. In Chapter II a general computer algorithm based on path enumeration method has been developed to calculate the

reliability of complex systems. It also evaluates an important parameter; namely, the sensitivity of the system reliability to individual system units. The introduction of the module concept considerably simplifies the evaluation of the reliability and sensitivity functions. In Chpater III this algorithm is used in conjunction with a modified integer gradient method [50] for solving a large class of reliability and availability optimization problems. In Chapter IV an optimal reliability design of process systems is proposed which takes into account the quantitative aspects of systems throughput or capacity. The problem is so formulated that finding the optimum configuration, also determines the optimum capacity of units to be used at each stage of system.

All the procedures developed in this research have been programmed in FORTRAN IV. A number of illustrative examples are included in each chapter.

CHAPTER II

RELIABILITY AND SENSITIVITY ANALYSIS OF COMPLEX SYSTEMS USING FLOW GRAPH METHODS

In this chapter a general computer algorithm has been developed to calculate the system reliability when given reliabilities of each unit in the system and relationships between the units in the form of a reliability graph. Besides giving the system reliability, the programs calculate the sensitivity of the system reliability to individual system units.

Reliability

Reliability has been defined in many ways. A widely accepted definition [5] reads: "Reliability is the probability of a device performing its purpose adequately for the period of time intended under the operating conditions encountered."

The mathematical derivation of the reliability function is treated in most reliability texts [5,25,49,59]. Here the general formula is given without proof

$$R(t) = \exp\left[-\int_0^t z(\tau)d\tau\right] \tag{2.1}$$

where, the function z(t) is called the failure rate function, hazard function, or hazard rate.

If the system has a failure time distribution with density function f(t), then the failure time distribution function is given by

$$F(t) = \int_0^t f(\tau)d\tau \tag{2.2}$$

called the unreliability of the system at time t. Thus, the system reliability is given by

$$R(t) = 1-F(t) = \exp\left[-\int_0^t z(\tau)d\tau\right] \tag{2.3}$$

From Equations (2.2) and (2.3) we obtain

$$z(t) = f(t)/R(t) \tag{2.4}$$

Typical failure time density functions with their corresponding reliability and failure rate functions are:

(1) Exponential

$$f(t) = \lambda e^{-\lambda t} \qquad R(t) = e^{-\lambda t} \qquad z(t) = \lambda$$

(2) Weibull

$$f(t) = \lambda\alpha t^{\alpha-1}\, e^{-\lambda t^{\alpha}} \qquad R(t) = e^{-\lambda t^{\alpha}} \qquad z(t) = \lambda\alpha t^{\alpha-1}$$

(where α, $\lambda > 0$)

(3) Gamma

$$f(t) = \frac{\lambda}{(r-1)!}(\lambda t)^{r-1}\, e^{-\lambda t} \qquad R(t) = \sum_{j=0}^{r-1} \frac{(\lambda t)^j e^{-\lambda t}}{j!} \qquad z(t) = \frac{\lambda^r t^{r-1}}{(r-1)!}\frac{e^{-\lambda t}}{R(t)}$$

(r is a positive integer)

In the following it is assumed that individual units forming a system have exponential failure time density functions. The analysis, however, with other failure time density functions is an extention of the techniques presented in this thesis.

Methods for Computing Complex System Reliability

1. State Enumeration or Event-Space Method

In this method we first make a list of all possible mutually exclusive states of the system. A state is defined by listing the successful

and failed elements in the system. In general the total number of states will be given by 2^n, where n is the number of units or elements in the system. Next, we identify the states which result in successful system operation and determine the probability of occurrence of each successful state. Finally, we sum up all the successful state probabilities which gives the system reliability. Although the method is easily programmed [9] it is computationally not feasible for systems having large number of elements.

2. Network Reduction Method

In this method the series, parallel and series-parallel subsystems are combined until we end up with a non-series parallel system which cannot be further reduced. Reference [45] then suggests the use of a factoring theorem. A particular element x is selected, and the two networks are obtained and generated when x is replaced by a short circuit (perfect connection) and an open circuit. If these two networks are simple series-parallel, they can be reduced. Otherwise the next block, y, must be selected and the procedure repeated.

Suppose block x is selected and the two networks are generated. The system reliability is then [45]

$$R = R_x * R\Big|_{R_{x=1}} + (1-R_x) * R\Big|_{R_{x=0}} \tag{2.5}$$

where R_x is the reliability of block x. The method is discussed in [1,13,14,39,45].

3. Path Enumeration and Cut Enumeration Methods

These methods can be used to determine the reliability of any system not containing dependent failures [16,29,52]. A path is a set of elements which form a connection between input and output when traversed

in a stated direction. A minimal path is one in which no node is traversed more than once in tracing the path.

Let P_i (i=1, 2, ... M) be the ith minimal path in the system. If any path is operable, the system performs adequately. Thus, the system reliability is

$$R = P_r \left\{ \bigcup_{i=1}^{M} P_i \right\} \tag{2.6}$$

where $\cup$ denotes the union.

By use of the expansion rule for the probability of the union of M events [21], we have the formula

$$\begin{aligned} R = & \sum_{i=1}^{M} P_r \{P_i\} - \sum_{i=1}^{M} \sum_{j>i}^{M} P_r \{P_i \cap P_j\} \\ & + \sum_{i=1}^{M} \sum_{j>i}^{M} \sum_{k>j}^{M} P_r \{P_i \cap P_j \cap P_k\} + \ldots + (-1)^{M-1} P_r \left\{ \bigcap_{i=1}^{M} P_i \right\} \end{aligned} \tag{2.7}$$

where $\cap$ denotes the intersection.

A cut is defined as a set of elements that if they fail, the system will fail regardless of the condition of the other elements in the system. A minimal cut is one in which there is no proper subset of elements whose failure alone will cause the system to fail.

Let C_j (j=1, 2, ... M) be the minimal cuts in the system. If any cut is inoperable, the system fails. The system reliability is thus given by

$$R = 1 - P_r \left\{ \bigcup_{j=1}^{M} \overline{C}_j \right\} \tag{2.8}$$

$$= 1 - \sum_{j=1}^{M} P_r\left\{\overline{C}_j\right\} + \sum_{j=1}^{M}\sum_{k>j}^{M} P_r\left\{\overline{C}_j \cap \overline{C}_k\right\} - \sum_{j=1}^{M}\sum_{k>j}^{M}\sum_{l>k}^{M} P_r\left\{\overline{C}_j \cap \overline{C}_k \cap \overline{C}_l\right\}$$

$$+ \ldots + (-1)^j P_r\left\{\bigcap_{i=1}^{M} \overline{C}_j\right\} \tag{2.9}$$

where $\overline{C}_j$ denotes the complement of the event C_j; i.e., $\overline{C}_j$ denotes the failure of all elements of the cut C_j.

The algorithm developed in this chapter is based both on the network reduction and path enumeration methods. Advantages and disadvantages of this approach have been discussed and compared with typical algorithms based on the state enumeration or event space methods.

Module Representation of Reliability Graphs

A reliability flow graph consists of nodes, branches, and modules with directed arrows between them. Consider the modules in the complexly connected reliability flow graphs shown in Figures 2.1, 2.2, and 2.4. The reliability flow graph of Figure 2.1 represents a series-parallel-series system distinct from simple series, parallel or series-parallel system. The reliability flow graph of Figures 2.2 and 2.4 represent non-series-parallel systems. In the module representation of reliability graphs more than one module between any two arbitrary nodes is permitted. A module is defined to be a single unit or simply connected parallel units. A configuration permitted as a module is limited to either of the following:

1. Single Unit Module

A single unit module is one which consists of a single unit. The module reliability is the reliability of the unit itself.

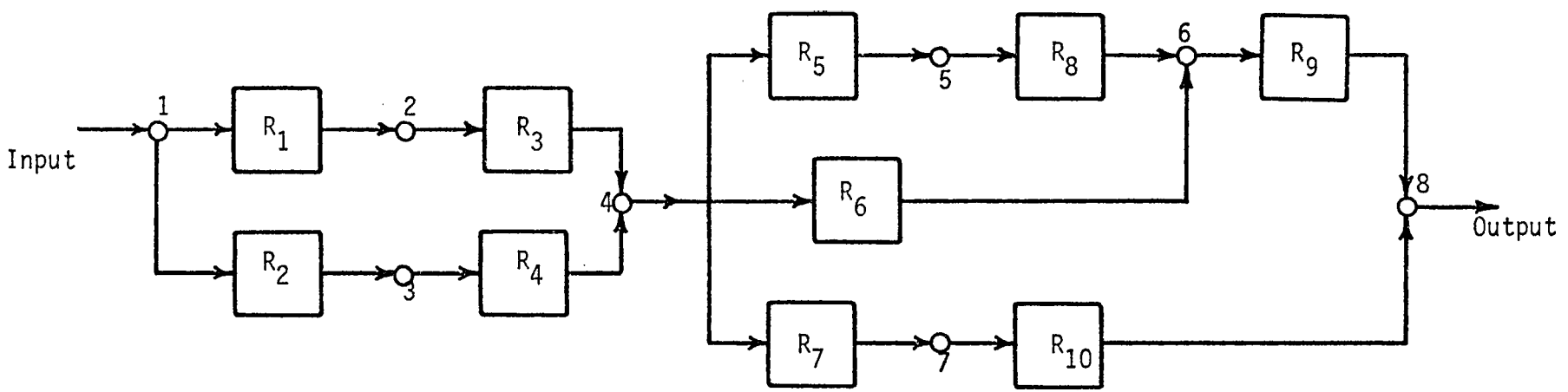


Figure 2.1

Reliability Graph - Series-Parallel-Series System

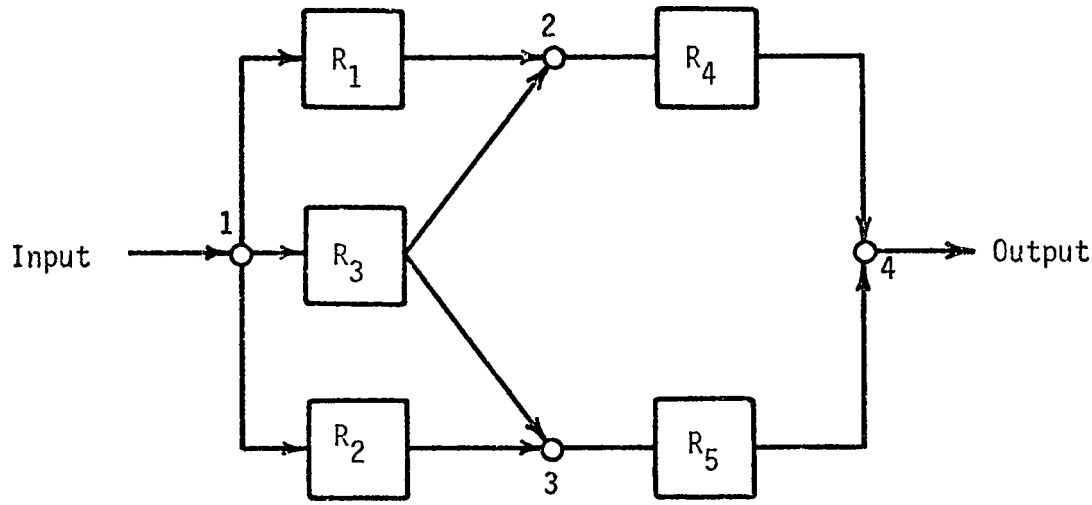


Figure 2.2

Reliability Graph - Non-Series-Parallel System

2. Multi-Unit Module

A module which consists of multiple parallel units is either an active redundancy module or a standby redundancy module as shown in Figure 2.3. We distinguish between three standby redundancy modules. A unit is called a cold standby when the failure probability in standby is zero. When the failure probability is the same in standby as in the service, it is called a hot standby. Cases that lie in between these two extremes are called warm standby. Depending on the type of redundancy, we have different expressions for module reliability as shown in Appendix A.

The advantages of the module representation of a reliability graph will become clear in the next few sections.

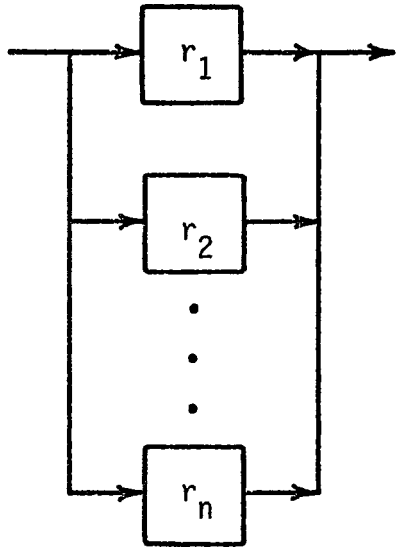
Reliability Calculation

The system reliability is defined here as the probability of the successful functioning of all the modules in at least one minimal path. A minimal path is one which contains a minimum number of modules and no node is traversed more than once in tracing the path.

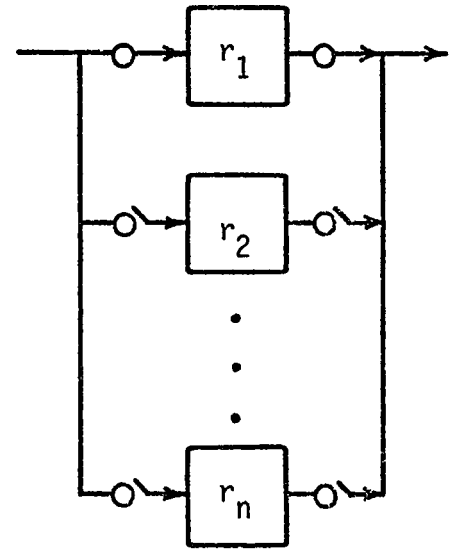
If P_i (i=1, 2, ... M) denotes the minimal paths in the system, the system reliability R is as shown in Equations (2.6) and (2.7).

The system reliability R is thus given in terms of module reliabilities by

$$R = \sum_{i=1}^{M} \prod_{l \varepsilon P_i} R_l - \sum_{i=1}^{M} \sum_{j>i}^{M} \prod_{l \varepsilon P_i \cup P_j} R_l + \sum_{i=1}^{M} \sum_{j>i}^{M} \sum_{k>j}^{M} \prod_{l \varepsilon P_i \cup P_j \cup P_k} R_l + \ldots + (-1)^{M-1} \prod_{l \varepsilon \bigcup_{i=1}^{M} P_i} R_l \tag{2.10}$$



Active Redundancy Module



Standby Redundancy Module

Figure 2.3

Multi-Unit Module

where the members of the ith path, the union of the ith and the jth paths, etc., are denoted by $l \varepsilon P_i$, $l \varepsilon P_i \cup P_j$, etc.

The total number of terms Z involved in equation (2.10) is given by

$$Z = 2^M - 1 \tag{2.11}$$

It should be noted in Equations (2.7) and (2.10) that if, for example, the path P_1 has modules R_1, R_2, and R_3, and the path P_2 has modules R_2, R_3 R_4, and R_5, then $P_r\{P_1 \cap P_2\}$ is given by $R_1R_2R_3R_4R_5$, not $R_1R_2^2R_3^2R_4R_5$.

Sensitivity Calculation

It is very interesting and important that the system reliability expression given by Equation (2.10) is a bilinear function of each module's reliability. By use of this property, the sensitivity of the system reliability to the module reliability R_i can be obtained by the simple rule

$$S_{R_i} = \partial R/\partial R_i = R\Big|_{R_i=1} - R\Big|_{R_i=0} \quad i=1, 2, \ldots N \tag{2.12}$$

where N is the total number of modules in the system. We call S_{Ri} a module sensitivity.

Two types of sensitivities are of interest. The first is the sensitivity of the system reliability to a unit reliability. A module sensitivity is itself of this type if the module is a single-unit module. For a multiple-unit module, the sensitivity is given by

$$S_{r_{ij}} = \frac{\partial R}{\partial r_{ij}} = \frac{\partial R}{\partial R_i} * \frac{\partial R_i}{\partial r_{ij}} \tag{2.13}$$

where r_{ij} is the reliability of the jth unit of the ith module. For an active redundancy module with different units, we have the following expression:

$$S_{r_{ij}} = S_{R_i} \left[R_i \Big|_{r_{ij}=1} - R_i \Big|_{r_{ij}=0} \right] = S_{R_i} \prod_{\substack{j=1 \\ j \neq i}}^{N_i} (1-r_{ij}) \tag{2.14}$$

where N_i is the number of units in the ith module.

The second type of sensitivity is the sensitivity of the system reliability to the number of units in a module. This type of sensitivity is defined for the active redundancy module or the standby redundancy module with identical units.

$$S_{N_i} = \frac{\partial R}{\partial N_i} = \frac{\partial R}{\partial R_i} * \frac{\partial R_i}{\partial N_i} = S_{R_i} * \frac{\partial R_i}{\partial N_i} \tag{2.15}$$

The sensitivity for the active redundancy module with identical units is given by

$$S_{N_i} = S_{R_i} \left[- (1-r_i)^{N_i} \ln (1-r_i) \right] \tag{2.16}$$

The sensitivity for the standby modules can be approximately calculated by

$$S_{N_i} \simeq S_{R_i} \left[R_i (N_i+1) - R_i(N_i) \right] \tag{2.17}$$

A large sensitivity value means that we may increase the system reliability greatly by increasing the corresponding individual reliability or by increasing the number of units in the corresponding module. Sensitivity, therefore, is a measure of system reliability improvement and provides important information for maximizing the reliability of complex systems. Sensitivity expressions for multi-unit modules are presented in Appendix A.

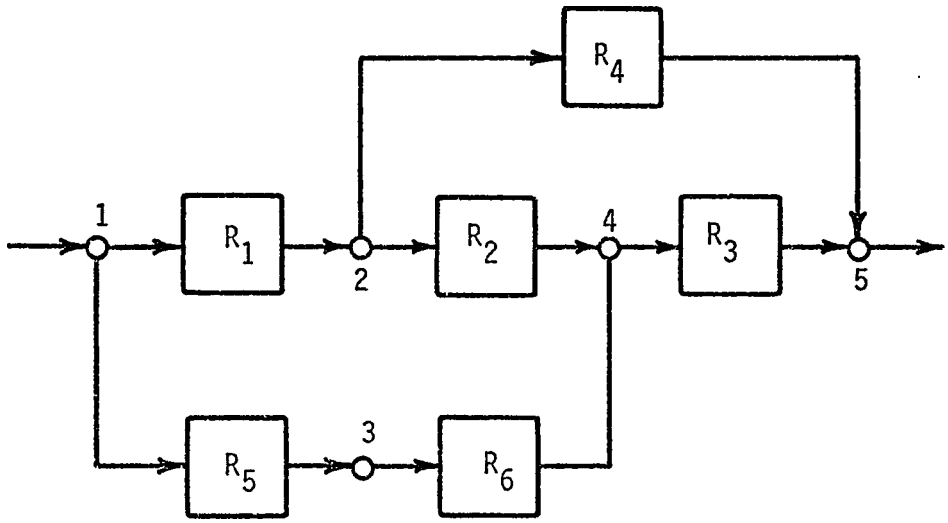


Figure 2.4

Reliability Graph - Non-Series-Parallel System

Basic Algorithm

The total system reliability and sensitivities can be obtained by the following six-step procedure:

1. Find all minimal paths using the reliability graph;

2. Find all the required unions of the paths;

3. Give each path union a reliability expression in terms of module reliability;

4. Sum up all the reliability expressions obtained above according to Equation (2.10);

5. Evaluate the system reliability by substituting numerical values of each module reliability; and,

6. Evaluate the desired sensitivities from Equations (2.12) to (2.17).

A brief description of each step is given and illustrated by way of a non-series-parallel reliability graph in Figure 2.4.

Path Finding Algorithm

We require a path-finding routine which detects all of the minimal paths connecting input and output when given a system reliability graph. The path finding algorithm developed by Henley and Williams [27] is ideally suited for this purpose. Having been given the reliability graph of Figure 2.4, the routine detects three paths in the node form shown in Table (2.1). By the nature of the algorithm only minimal paths are detected. Due to the possibility that a branch may have more than one module and/or a module may be present in more than one branch, the path finding algorithm [27] was modified to trace the modules in each minimal path as shown in Table 2.1. This allows a unique representation of each

PATH	TRACE NODES	TRACE MODULES
P1	1 ⟶ 2 ⟶ 4 ⟶ 5	R1 ⟶ R2 ⟶ R3
P2	1 ⟶ 2 ⟶ 5	R1 ⟶ R4
P3	1 ⟶ 3 ⟶ 4 ⟶ 5	R5 ⟶ R6 ⟶ R3

Table 2.1

PATH DETECTION

path which in general is not possible when paths are traced in the node form. Details of the modified path finding algorithm are given in Appendix B.

Another path finding algorithm suitable for reliability calculation and its Fortran listing are available in the literature [4,46].

Algorithm to Find Path-Unions

Each detected path is converted into the binary form shown in Figure 2.5. The locations containing 1's indicate the modules present in the path. N is the total number of modules in the system. Each path is given a positive sign. This is the sign of the first summation term in the reliability expression given by Equation (2.7) or (2.10).

The next step is to generate the binary representation of all the necessary unions of paths. This is easily done by using a bitwise logical OR operation in the order of ascending path number. That is, (P_1) OR (P_2), (P_1) OR (P_3), ..., (P_1) OR (P_M), (P_2) OR (P_3), ..., (P_2) OR (P_M), ..., (P_{M-1}) OR (P_M), which completes the 2-path unions. We then proceed to 3-path unions, $((P_1)$ OR $(P_2))$ OR (P_3), ..., $((P_1)$ OR $(P_2))$ OR (P_M), ..., $((P_{M-2})$ OR $(P_{M-1}))$ OR (P_M). Similar procedures continue until we reach M-path union, (P_1) OR (P_2) OR ... OR (P_{M-1}) OR (P_M). The sign changes with the generation of each higher order path union. Thus, it is negative for 2-union paths, positive for 3-union paths, ..., and $(-1)^{M-1}$ for M-union path. Details of the algorithm that generates all the necessary path unions are given in Appendix C.

The binary representation of the paths and path-unions for the reliability graph of Figure 2.4 are shown in Figure 2.6. Each path or path union corresponds to a term in Equation (2.10).

Sign	Module					
	1	2	3	4	...	N
+	1	0	0	1	...	1

Figure 2.5

Binary Representation of a Path or a Path Union

	SIGN	MODULE 1	2	3	4	5	6	
P1	+	1	1	1	0	0	0	MINIMUM PATHS
P2	+	1	0	0	1	0	0	
P3	+	0	0	1	0	1	1	
P1 OR P2	-	1	1	1	1	0	0	2 PATH UNIONS
P1 OR P3	-	1	1	1	0	1	1	
P2 OR P3	-	1	0	1	1	1	1	
P1 OR P2 OR P3	+	1	1	1	1	1	1	3 PATH UNION

Figure 2.6

Binary Representation of 3 Paths and Unions of Paths

System Reliability

We now pick up every module that has a 1 in the corresponding location and form the product expressions for each path and path union. The collection of all the terms with their proper signs gives the required system reliability function as given in Equation (2.10).

For example, the system reliability expression in Figure 2.6 is given as

$$\begin{aligned} R = {} & (R_1R_2R_3 + R_1R_4 + R_3R_5R_6) \\ & - (R_1R_2R_3R_4 + R_1R_2R_3R_5R_6 + R_1R_3R_4R_5R_6) \\ & + R_1R_2R_3R_4R_5R_6 \end{aligned} \tag{2.18}$$

When given numerical values for the reliability of each module, we can obtain the system reliability. The module sensitivity S_{R_i} can also be calculated from Equation (2.12) by evaluating the system reliability twice, once with R_i=1 and once with R_i=0. The other sensitivities can also be calculated through the module sensitivities by Equations (2.13) to (2.17).

A number of terms in the reliability expression given in Equation (2.10) and generated as indicated above are identical and have opposite signs. A significant number of such terms can be eliminated by modifying the algorithm that generates the path-unions. Details of this modification together with an example problem are discussed in Appendix C. This reduces the storage requirement and computation time considerably, especially during system optimization discussed in the next chapter.

Comparison with State Enumeration Algorithm

An alternate method of calculating system reliability is the method of state enumeration. The algorithm for the state enumeration can

be briefly stated as follows [9]:

1. Find all the possible combinations of the states of the units (up or down);

2. Find each combination which connects input and output; calculate the product of the unreliabilities of the down units and the reliabilities of the up units;

3. Sum up the products obtained in Step 2. This gives the system reliability expression.

The algorithm can be easily computerized by using Boolean algebra [9]. As an example, consider the reliability graph shown in Figure 2.7. The algorithm can be understood by referring to Table 2.2. From the table, the system reliability function is given by

$$\begin{aligned} R = {} & (1-r_1)(1-r_2)r_3\,(1-r_4)r_5 + (1-r_1)(1-r_2)r_3r_4r_5 \\ & + (1-r_1)r_2(1-r_3)r_4(1-r_5) + (1-r_1)r_2(1-r_3)r_4r_5 \\ & + (1-r_1)r_2r_3(1-r_4)r_5 + (1-r_1)r_2r_3r_4(1-r_5) \\ & + (1-r_1)r_2r_3r_4r_5 + r_1(1-r_2)(1-r_3)r_4(1-r_5) \\ & + r_1(1-r_2)(1-r_3)r_4r_5 + r_1(1-r_2)r_3(1-r_4)r_5 \\ & + r_1(1-r_2)r_3r_4(1-r_5) + r_1(1-r_2)r_3r_4r_5 \\ & + r_1r_2(1-r_3)r_4(1-r_5) + r_1r_2(1-r_3)r_4r_5 \\ & + r_1r_2r_3(1-r_4)r_5 + r_1r_2r_3r_4(1-r_5) + r_1r_2r_3r_4r_5 \end{aligned} \tag{2.19}$$

Paths and path unions required in the path enumeration algorithm are listed in Table 2.3. From Table 2.3 the system reliability function is easily derived as

$$R = r_1r_4 + r_2r_4 + r_3r_5 - (r_1r_2r_4 + r_1r_3r_4r_5 + r_2r_3r_4r_5) + r_1r_2r_3r_4r_5 \tag{2.20}$$

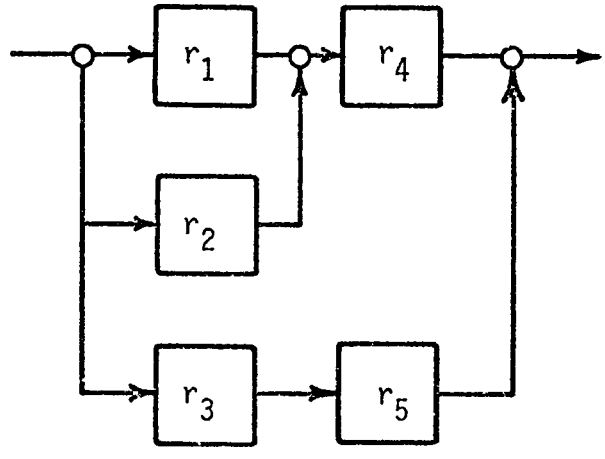


Figure 2.7

Reliability Graph

STATE NUMBER	BINARY NUMBER $r_1 r_2 r_3 r_4 r_5$	GIVING A PATH?
0	0 0 0 0 0	No
1	0 0 0 0 1	No
2	0 0 0 1 0	No
3	0 0 0 1 1	No
4	0 0 1 0 0	No
5	0 0 1 0 1	Yes
6	0 0 1 1 0	No
7	0 0 1 1 1	Yes
8	0 1 0 0 0	No
9	0 1 0 0 1	No
10	0 1 0 1 0	Yes
11	0 1 0 1 1	Yes
12	0 1 1 0 0	No
13	0 1 1 0 1	Yes
14	0 1 1 1 0	Yes
15	0 1 1 1 1	Yes
16	1 0 0 0 0	No
17	1 0 0 0 1	No
18	1 0 0 1 0	Yes
19	1 0 0 1 1	Yes
20	1 0 1 0 0	No
21	1 0 1 0 1	Yes
22	1 0 1 1 0	Yes
23	1 0 1 1 1	Yes
24	1 1 0 0 0	No
25	1 1 0 0 1	No
26	1 1 0 1 0	Yes
27	1 1 0 1 1	Yes
28	1 1 1 0 0	No
29	1 1 1 0 1	Yes
30	1 1 1 1 0	Yes
31	1 1 1 1 1	Yes

Table 2.2 All combination of the states

PATHS AND PATH UNIONS	SIGN	MODULE				
		r_1	r_2	r_3	r_4	r_5
P1	+	1	0	0	1	0
P2	+	0	1	0	1	0
P3	+	0	0	1	0	1
P1 OR P2	-	1	1	0	1	0
P1 OR P3	-	1	0	1	1	1
P2 OR P3	-	0	1	1	1	1
(P1 OR P2) OR P3	+	1	1	1	1	1

Table 2.3

Paths and Path Unions

Although the two expressions given by Equations (2.19) and (2.20) are different in form, it can be easily verified that both are equivalent.

We can conclude from the above comparison and from more general consideration that the advantages of the path enumeration method are:

1. Steps to reach the system reliability function are much less if the number of paths is fairly small and is unaffected by the total number of modules in the system;

2. The number of terms in the system reliability function are fewer; and,

3. Any term in the system reliability function contains equal or fewer multipliers.

The module structure introduced earlier in the chapter brings about a further reduction in the number of paths. Thus, the units r_1 and r_2 are combined and treated as the active redundancy modules R_1 as shown in Figure 2.8. The path enumeration algorithm then gives the path and path unions shown in Table 2.4. The system reliability is given by

$$R = R_1 r_4 + r_3 r_5 - R_1 r_3 r_4 r_5 \tag{2.21}$$

where

$$R_1 = 1 - (1-r_1)(1-r_2) = r_1 + r_2 - r_1 r_2 \tag{2.22}$$

In this way, the previously stated advantages of the path enumeration method are enhanced by introducing the module structure. Since it is customary to use parallel redundancy configurations to increase the reliability of a weak unit, the module concept significantly reduces the number of paths in the system thus reducing the computing time as shown by the following examples:

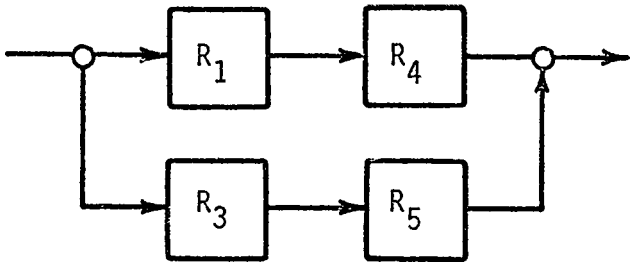


Figure 2.8

Module Representation of Figure 2.7

PATHS AND PATH UNION	SIGN	MODULE			
		R_1	r_3	r_4	r_5
P1	+	1	0	1	0
P2	+	0	1	0	1
P1 OR P2	-	1	1	1	1

Table 2.4

Paths and Path Union

Examples

At first, consider the fairly complicated reliability graph [46] shown in Figure 2.9. The reliability of each unit is given in Table 2.5. In this original graph, there are 16 units and 55 paths. The number of path and path unions required for calculating the system reliability, therefore, will be $2^{55}-1$ from Equation (2.11), which is far in excess of 10^{16}. The application of the path enumeration method to the original graph is impossible unless some approximations are introduced.

By using the module concept, the graph is automatically converted to Figure 2.1, where R_1, R_3, R_5, R_9, and R_{10} are actively connected multi-unit modules which consist of units (r_1, r_2), (r_4, r_5), (r_7, r_8), (r_{12}, r_{13}, r_{14}) and (r_{15}, r_{16}), respectively; R_2, R_4, R_6, R_7, and R_8 are single unit modules consisting of units r_3, r_6, r_9, r_{10}, and r_{11}, respectively. The module representation of the reliability graph is shown in Table 2.6.

The converted graph has only six paths as shown in Table 2.7. The binary representation of the six paths is shown in Table 2.8. Since the number of paths is greatly reduced, the path enumeration algorithm discussed earlier can be applied easily and efficiently to produce the system reliability value of 0.97043 in a few seconds.

The sensitivities are listed in Table 2.9, where attention is focused on the first type of sensitivity, $\partial R/\partial_{r_{ij}}$ given by Equation (2.14). The table shows that the module R_4, that is, unit r_6, has the larger sensitivity value (0.085), and the first unit of module R_{10}, unit r_{15}, has the smallest. It is, therefore, more efficient to increase the unit reliability of r_6 in order to increase the total reliability of the system.

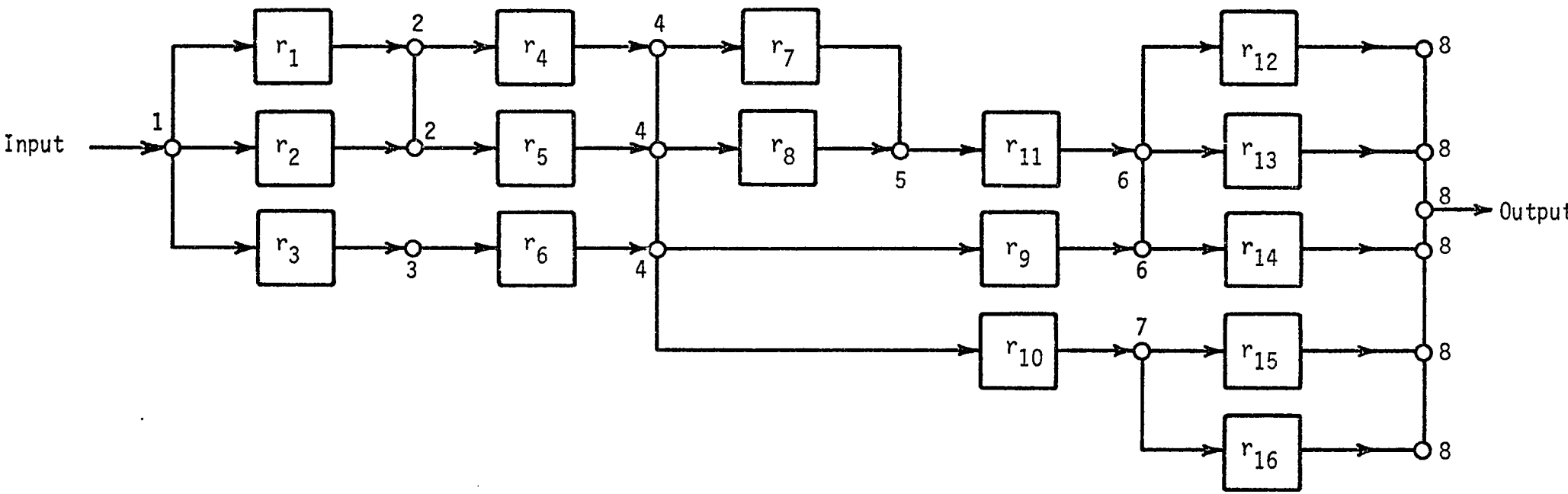


Figure 2.9

Reliability Graph - 55 Paths Between Input and Output

UNIT	RELIABILITY
r_1	0.80
r_2	0.70
r_3	0.90
r_4	0.75
r_5	0.85
r_6	0.87
r_7	0.82
r_8	0.62
r_9	0.88
r_{10}	0.75
r_{11}	0.89
r_{12}	0.85
r_{13}	0.75
r_{14}	0.65
r_{15}	0.70
r_{16}	0.90

Table 2.5

Unit Reliabilities

MODULE	BRANCH	UNIT RELIABILITY	NUMBER OF UNITS
1	1 ⟶ 2	0.800000 0.700000	2
2	1 ⟶ 3	0.900000	1
3	2 ⟶ 4	0.750000 0.850000	2
4	3 ⟶ 4	0.870000	1
5	4 ⟶ 5	0.820000 0.620000	2
6	4 ⟶ 6	0.880000	1
7	4 ⟶ 7	0.750000	1
8	5 ⟶ 6	0.890000	1
9	6 ⟶ 8	0.850000 0.750000 0.650000	3
10	7 ⟶ 8	0.700000 0.900000	2

Table 2.6

Module Representation

PATH	TRACE NODES	TRACE MODULES
P1	1 → 2 → 4 → 6 → 8 →	1 → 3 → 6 → 9
P2	1 → 3 → 4 → 6 → 8 →	2 → 4 → 6 → 9
P3	1 → 2 → 4 → 5 → 6 → 8 →	1 → 3 → 5 → 8 → 9
P4	1 → 3 → 4 → 5 → 6 → 8 →	2 → 4 → 5 → 8 → 9
P5	1 → 2 → 4 → 7 → 8 →	1 → 3 → 7 → 10
P6	1 → 3 → 4 → 7 → 8 →	2 → 4 → 7 → 10

Table 2.7

Path Detection

PATH	MODULE									
	1	2	3	4	5	6	7	8	9	10
P1	1	0	1	0	0	1	0	0	1	0
P2	0	1	0	1	0	1	0	0	1	0
P3	1	0	1	0	1	0	0	1	1	0
P4	0	1	0	1	1	0	0	1	1	0
P5	1	0	1	0	0	0	1	0	0	1
P6	0	1	0	1	0	0	1	0	0	1

Table 2.8

Binary Representation of Paths

MODULE	MODULE RELIABILITY	MODULE SENSITIVITY $\partial R/\partial R_i$	UNIT SENSITIVITY $\partial R_i/\partial r_{ij}$	SYSTEM SENSITIVITY $\partial R/\partial r_{ij}$
1	0.940000	0.206964	0.300000 0.200000	0.620891E-01 0.413928E-01
2	0.900000	0.821141E-01	1.00000	0.821141E-01
3	0.962500	0.202126	0.150000 0.250000	0.303189E-01 0.505314E-01
4	0.870000	0.849457E-01	1.00000	0.849456E-01
5	0.931600	0.281274E-01	0.380000 0.180000	0.106884E-01 0.506293E-02
6	0.880000	0.450027E-01	1.00000	0.450027E-01
7	0.750000	0.316913E-01	1.00000	0.316913E-01
8	0.890000	0.294421E-01	1.00000	0.294421E-01
9	0.986875	0.261395	0.875000E-01 0.525000E-01 0.375000E-01	0.228721E-01 0.137233E-01 0.980232E-02
10	0.970000	0.245036E-01	0.100000 0.300000	0.245036E-02 0.735107E-02

Table 2.9

Sensitivities $\partial R/\partial r_{ij}$

Moreover, the sensitivity tells us that the increase in total reliability will amount to approximately 0.0085 when the increases in the unit reliability of r_6 is 0.1.

As the second example, we consider a safety system [53]. Figure 2.10 shows a boiler DDC safety system (pressure safety system). Reliabilities of each unit are listed in Table 2.10. The original graph has 26 paths. After being converted to a module configuration shown in Figure 2.11 and Table 2.11, the number of paths is reduced to only nine as shown in Tables 2.12 and 2.13.

The total reliability of the system is computed to be 0.99927. The sensitivities are listed in Table 2.14. In this case, the sensitivities are evaluated with respect to the number of units in a module, that is, $\partial R/\partial N_i$ defined by Equation (2.16). The table shows that the module R_4 has the largest sensitivity and the module R_5 has the smallest sensitivity. Adding one more unit (Converter 1) to the module R_5 is of no use, as the increase in the total reliability will only be 0.0000009.

Advantages of Module Structure

It can be seen from the above examples and more general consideration that the introduction of module structure has the following advantages:

1. It considerably reduces the number of paths and, hence, provides a means for the efficient evaluation of the reliability of complex systems;

2. It greatly simplifies the evaluation of the sensitivity function, an important design parameter that will be used in system optimization;

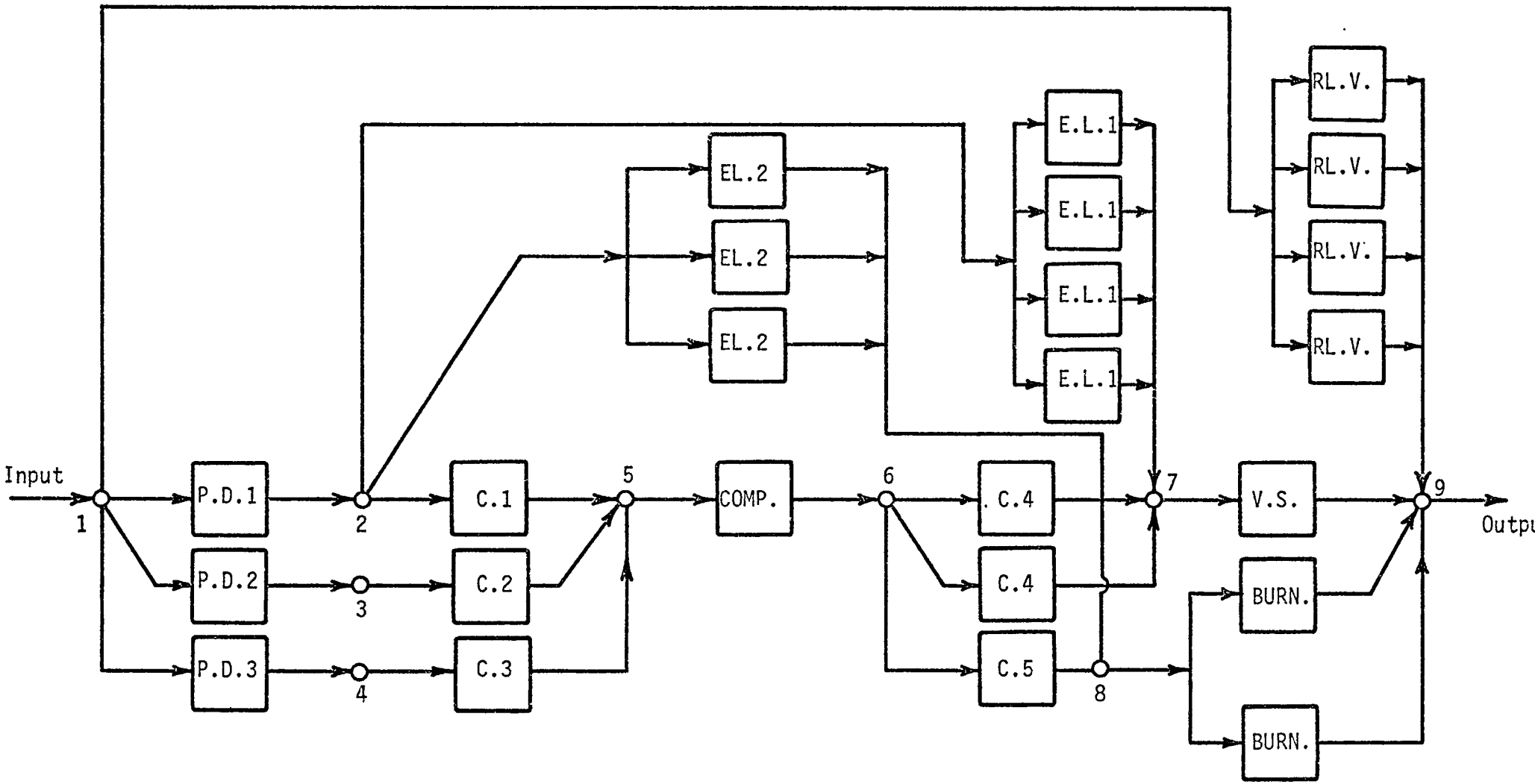


Figure 2.10

Reliability Graph - Boiler DDC Safety System

UNIT		RELIABILITY
P.D. 1:	Pressure Detector 1	0.68
P.D. 2:	Pressure Detector 2	0.75
P.D. 3:	Pressure Detector 3	0.75
C.1 :	Converter 1	0.70
C.2 :	Converter 2	0.70
C.3 :	Converter 3	0.70
C.4 :	Converter 4	0.70
C.5 :	Converter 5	0.70
COMP. :	Computer	0.70
V.S. :	Valve Servo	0.75
BURN. :	Burner	0.68
RL.V. :	Relief Valve	0.75
E.L. 1:	Emergency Line 1	0.68
E.L. 2:	Emergency Line 2	0.68

Table 2.10

Unit Reliabilities

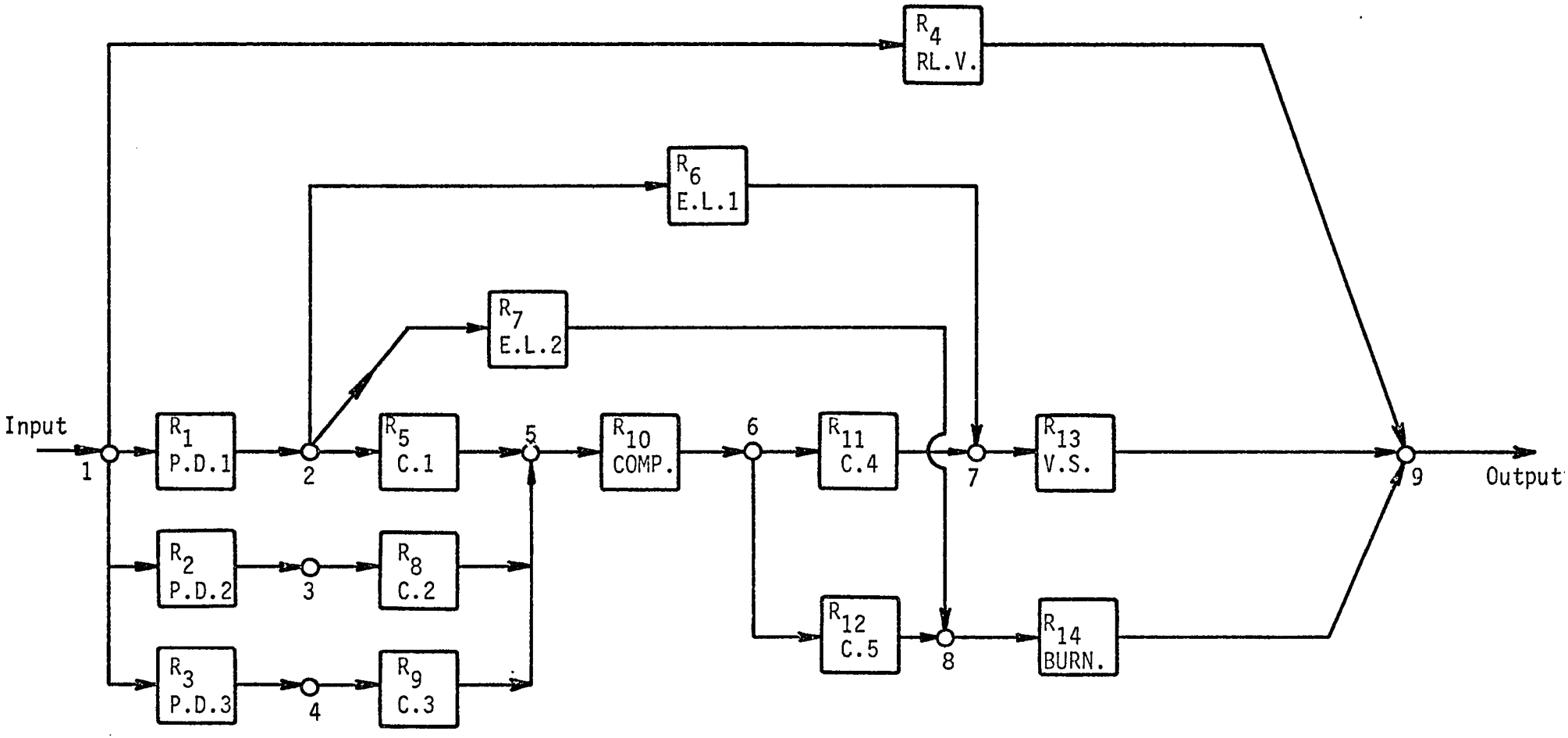


Figure 2.11

Module Representation of the Boiler DDC Safety System

MODULE	BRANCH	UNIT RELIABILITY	NUMBER OF UNITS
1	1 ⟶ 2	0.680000	1
2	1 ⟶ 3	0.750000	1
3	1 ⟶ 4	0.750000	1
4	1 ⟶ 9	0.750000	4
5	2 ⟶ 5	0.700000	1
6	2 ⟶ 7	0.680000	4
7	2 ⟶ 8	0.680000	3
8	3 ⟶ 5	0.700000	1
9	4 ⟶ 5	0.700000	1
10	5 ⟶ 6	0.700000	1
11	6 ⟶ 7	0.700000	2
12	6 ⟶ 8	0.700000	1
13	7 ⟶ 9	0.750000	1
14	8 ⟶ 9	0.680000	2

Table 2.11

Module Representation

PATH	TRACE NODES	TRACE MODULES
P1	1 ⟶ 9 ⟶	4
P2	1 ⟶ 2 ⟶ 7 ⟶ 9 ⟶	1 ⟶ 6 ⟶ 13
P3	1 ⟶ 2 ⟶ 5 ⟶ 6 ⟶ 7 ⟶ 9 ⟶	1 ⟶ 5 ⟶ 10 ⟶ 11 ⟶ 13
P4	1 ⟶ 3 ⟶ 5 ⟶ 6 ⟶ 7 ⟶ 9 ⟶	2 ⟶ 8 ⟶ 10 ⟶ 11 ⟶ 13
P5	1 ⟶ 4 ⟶ 5 ⟶ 6 ⟶ 7 ⟶ 9 ⟶	3 ⟶ 9 ⟶ 10 ⟶ 11 ⟶ 13
P6	1 ⟶ 2 ⟶ 8 ⟶ 9 ⟶	1 ⟶ 7 ⟶ 14
P7	1 ⟶ 2 ⟶ 5 ⟶ 6 ⟶ 8 ⟶ 9 ⟶	1 ⟶ 5 ⟶ 10 ⟶ 12 ⟶ 14
P8	1 ⟶ 3 ⟶ 5 ⟶ 6 ⟶ 8 ⟶ 9 ⟶	2 ⟶ 8 ⟶ 10 ⟶ 12 ⟶ 14
P9	1 ⟶ 4 ⟶ 5 ⟶ 6 ⟶ 8 ⟶ 9 ⟶	3 ⟶ 9 ⟶ 10 ⟶ 12 ⟶ 14

Table 2.12

Path Detection

PATH	MODULE													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
P1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
P2	1	0	0	0	0	1	0	0	0	0	0	0	1	0
P3	1	0	0	0	1	0	0	0	0	1	1	0	1	0
P4	0	1	0	0	0	0	0	1	0	1	1	0	1	0
P5	0	0	1	0	0	0	0	0	1	1	1	0	1	0
P6	1	0	0	0	0	0	1	0	0	0	0	0	0	1
P7	1	0	0	0	1	0	0	0	0	1	0	1	0	1
P8	0	1	0	0	0	0	0	1	0	1	0	1	0	1
P9	0	0	1	0	0	0	0	0	1	1	0	1	0	1

Table 2.13

Binary Representation of Paths

MODULE	MODULE RELIABILITY	MODULE SENSITIVITY $\partial R/\partial R_i$	UNIT SENSITIVITY $\partial R_i/\partial N_i$	SYSTEM SENSITIVITY $\partial R/\partial N_i$
1	0.680000	0.192148E-02	0.364619	0.700609E-03
2	0.750000	0.257737E-03	0.346574	0.893248E-04
3	0.750000	0.257737E-03	0.346574	0.893248E-04
4	0.996094	0.187413	0.541521E-02	0.101488E-02
5	0.700000	0.256121E-05	0.361192	0.925088E-06
6	0.989514	0.104235E-03	0.119478E-01	0.124538E-05
7	0.967232	0.330605E-03	0.373370E-01	0.123438E-04
8	0.700000	0.276147E-03	0.361192	0.997419E-04
9	0.700000	0.276147E-03	0.361192	0.997419E-04
10	0.700000	0.868859E-03	0.361192	0.313825E-03
11	0.910000	0.190398E-03	0.108357	0.206310E-04
12	0.700000	0.205885E-03	0.361192	0.743638E-04
13	0.750000	0.542155E-03	0.346574	0.187897E-03
14	0.897600	0.811069E-03	0.116678	0.946339E-04

Table 2.14

Sensitivities $\partial R/\partial N_i$

3. Although the module itself may be composed of multiple units subject to dependent failures, for example, a standby redundancy module, the module representation of the system has no dependent failures. In other words, failure of a module does not affect the operation of other modules in the system. This allows the path enumeration algorithm to be used for reliability calculation.

Extension to System MTBF Calculation

The most important parameter of the system reliability is the mean time between failures (MTBF). The system MTBF is defined as

$$\overline{M} = \int_0^\infty R(t)dt \tag{2.23}$$

The sensitivity of the MTBF to the reliability r_{ij} of the jth unit belonging to the ith module, and to the number of units in the ith module is given by

$$\frac{\partial \overline{M}}{\partial r_{ij}} = \int_0^\infty \frac{\partial R}{\partial R_i} \frac{\partial R_i}{\partial r_{ij}} dt \tag{2.24}$$

and

$$\frac{d\overline{M}}{\partial N_i} = \int_0^\infty \frac{\partial R}{\partial R_i} \frac{\partial R_i}{\partial N_i} dt \tag{2.25}$$

Expressions for $\frac{\partial R_i}{\partial r_{ij}}$ and $\frac{\partial R_i}{\partial N_i}$ are shown in Appendix A.

Therefore, by the addition of an integration routine to the reliability calculation program described earlier, the system MTBF and its sensitivities can be easily calculated. A computer program containing a numerical integration routine based on Simpson's rule has been developed.

The MTBF calculation routine developed here is more flexible and easier to apply to more complicated systems than the RELCOMP routine

of Fleming [22] which is only applicable to series-parallel configurations.

Approximations to System Reliability

Earlier, path and cut enumeration methods for system reliability calculation were discussed. The method adopted here was based on the path enumeration algorithm because an efficient path finding algorithm was readily available. The introduction of module structure allowed reliability evaluation of fairly complex networks.

If the number of paths is large (more than 10), the computation time increases rapidly. This is evident from Equation (2.11) which gives the total number of terms involved in the reliability expression as a function of total number of paths in the system. Same is true for reliability calculation based on the cut enumeration. To avoid this difficulty, some lower and upper bounds approximations to system reliability are presented below [4,28,34,37,46,52].

First we develop reliability approximations based on the path enumeration method. The series given by Equation (2.7) has the following properties [37,46,52]:

$$R \leq R_{U_1} = \sum_{i=1}^{M} P_r \{P_i\}$$

$$R \geq R_{L_1} = \sum_{i=1}^{M} P_r \{P_i\} - \sum_{i=1}^{M} \sum_{j>i}^{M} P_r \{P_i \cap P_j\} \tag{2.26}$$

$$R \leq R_{U_2} = \sum_{i=1}^{M} P_r \{P_i\} - \sum_{i=1}^{M} \sum_{j}^{M}{}_{i} P_r \{P_i \cap P_j\}$$

$$+ \sum_{i=1}^{M} \sum_{j>i}^{M} \sum_{k>j}^{M} P_r \{P_i \cap P_j \cap P_k\}$$

$$\vdots$$

and

$$R_{U_1} \geq R_{U_2} \geq \cdots$$
$$R_{L_1} \leq R_{L_2} \leq \cdots \tag{2.27}$$

Therefore, R_{U_1}, R_{U_2}, ... can be used as successive upper bounds for R, and R_{L_1}, R_{L_2}, ... can be used as lower bounds for R. This is the lower and upper bounds reliability approximation. These approximations are very close when element reliabilities are small. This is also called the low reliability region approximation.

In the analogous way, we can develop high reliability region approximation formulae from the cut enumeration method [28,37,46,52].

From Equation (2.9) we have

$$R \geq R_{L_1} = 1 - \sum_{j=1}^{M} P_r \{\overline{C}_j\}$$
$$R \leq R_{U_1} = 1 - \sum_{j=1}^{M} P_r \{\overline{C}_j\} + \sum_{j=1}^{M} \sum_{k>j}^{M} P_r \{\overline{C}_j \overline{C}_k\} \tag{2.28}$$
$$\vdots$$

and

$$R_{U_1} \geq R_{U_2} \geq \cdots$$
$$R_{L_1} \leq R_{L_2} \leq \cdots \tag{2.29}$$

From Equation (2.28) the number of terms in the lower and upper bound computations R_{L_1} and R_{U_1} are M and M(M+1)/2, respectively. This is compared to 2^M-1 terms obtained by expanding Equation (2.9).

In the system MTBF calculation both the approximations based on path enumeration and cut enumeration will play an important role because element reliabilities change from high reliability to low reliability region as time increases from 0 to ∞.

Conclusions

The proposed algorithm is more powerful and efficient for deriving and evaluating the total system reliability, the sensitivities, and the MTBF of complex systems than other algorithms based on state enumeration method, since systems are often complex enough to have many units, but not so complicated as to have many paths in the module representation.

The procedures presented above have been programmed in FORTAN IV and consist of eight basic subroutines. The nature of the MAIN program depends on the application and may call one or more of these subroutines. Appendix G contains a description of each subroutine, including a schematic diagram, and a listing of the corresponding FORTRAN program. In addition to these basic subroutines there are several others which are concerned with specific applications. These will be discussed in later chapters.

CHAPTER III

OPTIMAL DESIGN FOR RELIABILITY AND AVAILABILITY OF COMPLEX SYSTEMS

Most of the earlier literature in the area of system reliability optimization consider only the series-parallel system [2,3,6,8,18-20,23, 24,30,31,33,36,38,40-44,48,51,54-56,61]. Such a system is shown in Figure 3.1 where each module has several identical components in parallel to provide redundancy, and the system fails if all the components in a module fail. The reason that this particular system has received so much attention is the separability of the reliability function. For the configuration shown in Figure 3.1 the system reliability is

$$\begin{aligned} R(\underline{n},\underline{r}) &= \prod_{j=1}^{N} R_j(n_j,r_j) \\ &= \prod_{j=1}^{N} \left[1 - (1-r_j)^{n_j}\right] \end{aligned} \tag{3.1}$$

where r_j is the reliability of a component in the jth module and n_j is the total number of identical components in the jth module. Taking logarithms of both sides of Equation (3.1)

$$\begin{aligned} \ln R(\underline{n},\underline{r}) &= \sum_{j=1}^{N} \ln R_j(n_j,r_j) \\ &= \sum_{j=1}^{N} \ln \left[1 - (1-r_j)^{n_j}\right] \end{aligned} \tag{3.2}$$

Since the natural logarithm is monotonic, maximization of the logarithm is equivalent to maximization of its argument. The form of Equation (3.2) for reliability maximization is more convenient to use since each term of the

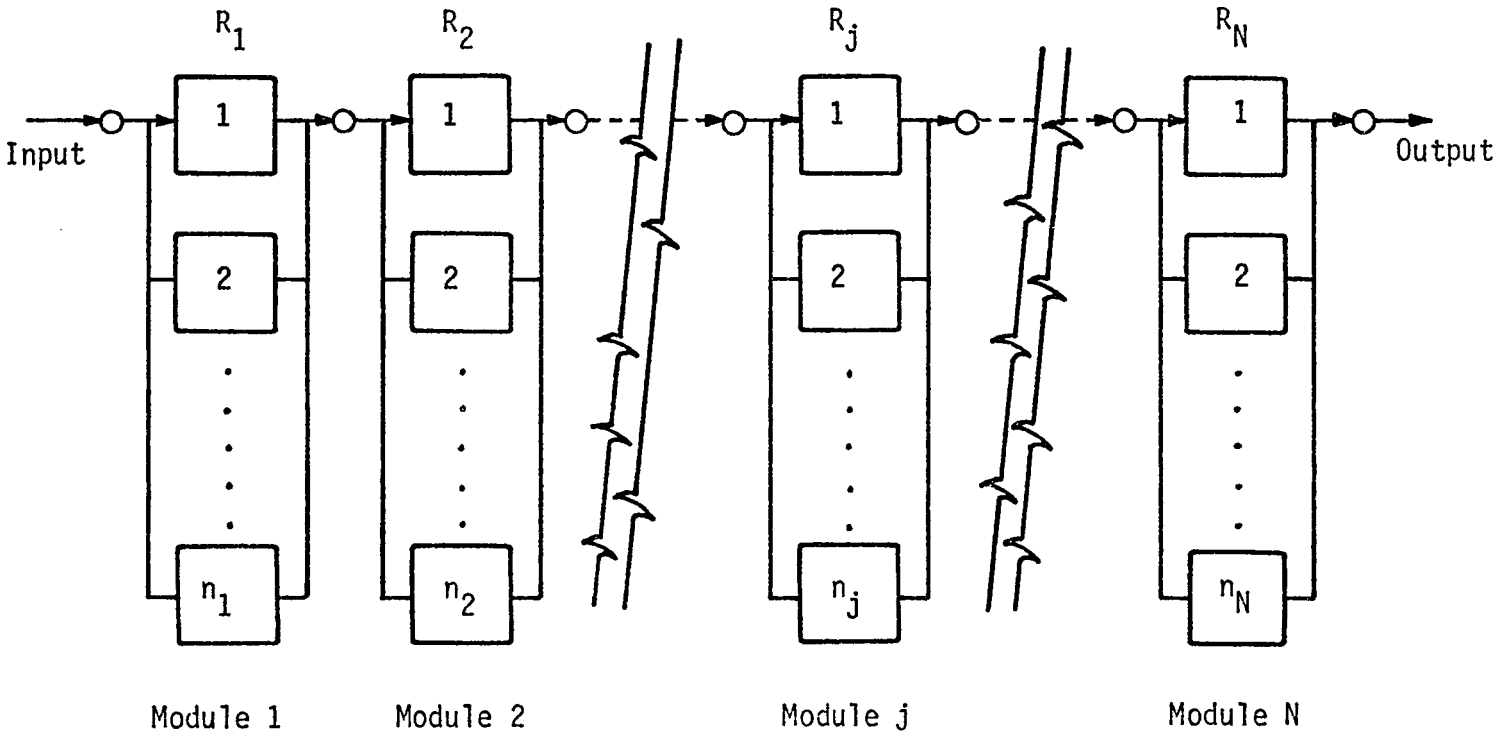


Figure 3.1

Series-Parallel System

sum depends on a single variable. A separable function can be analyzed as a multistage process to which the methods of dynamic programming [6,23,30,33, 38,43,61] and discrete maximum principle [19,55] are applicable. The problem may also be transformed and solved as an integer programming problem [24,41,44,54,56] or by using the Lagrangian multipliers technique [2,3,18,38,40]. The concept of dominating sequence (or a family of undominated allocations) has also been used to solve the problem [3,8,30, 38,48].

Burton and Howard [11,12] consider the reliability optimization of a series-parallel-series system. They present a dynamic programming model. The notion of the generalized decomposition operator is used to develop a set of recursive relations. Terano et.al. [53] also consider a series-parallel-series system. They linearize the system at the nominal point and apply dynamic programming to find optimum redundancies. A non-series-parallel system is treated by Tillman et.al. [57]. They use the sequential unconstrained minimization technique to find optimum module reliabilities that maximize the system reliability.

In the previous chapter a general computer algorithm has been developed to calculate the reliability of a complex system having been given the reliability of each unit and the system configuration in the form of a reliability graph. The algorithm also calculates the sensitivity of the system reliability to individual system units.

In this chapter the sensitivity function is used for maximizing the reliability,availability or profit of a complex system (series-parallel-series or non-series-parallel) subject to linear or non-linear constraints. The optimum seeking algorithm which finds the optimum redundancies and preventive maintenance schedules is based on an integer gradient method of Reiter and

Rice (R-R) [50]. The proposed algorithm can be used to optimize a large class of reliability problems encountered in practice.

Three types of reliability optimization problems are formulated:

(1) Optimal Allocation of Redundancy

The problem is to find the optimal redundancy for each module (unit reliabilities are specified) so as to maximize the system reliability subject to linear or non-linear cost constraints and is stated as follows:

Maximize the system reliability

$$R(\underline{n},\underline{r}) \tag{3.3}$$

where $\underline{r}$ is specified.

Subject to linear or non-linear cost constraints

$$\sum_{i=1}^{N} g_{ij}(n_i,r_i) \leq G_j \quad j = 1,2,\ldots m \tag{3.4}$$

and non-negative and integer constraints

$$n_i \geq k_i \qquad i = 1,2,\ldots N \tag{3.5}$$

$$n_i : \text{integer}$$

where k_i is the minimum number of units in the ith module that must operate for the module to function successfully.

This constitutes a non-linear integer programming problem.

(2) Optimal Allocation of Unit-Reliability or Unit Maintenance Interval

The problem is to find the optimal unit reliabilities or the optimum maintenance interval for each unit (module redundancies are specified) so as to maximize the system reliability subject to linear or non-linear cost constraints.

Once the optimal unit reliabilities are known the optimal maintenance interval can be evaluated by the relationship

$$r_i = e^{-T_i/(MTBF)_i} \tag{3.6}$$

where T_i is the maintenance interval of a unit belonging to the ith module, and $(MTBF)_i$ is the mean time between failure of a unit belonging to the ith module.

The optimization problem can be stated as follows

Maximize the system reliability

$$R(\underline{n},\underline{r}) \tag{3.7}$$

where $\underline{n}$ is specified.

Subject to linear or non-linear cost constraints

$$\sum_{i=1}^{N} g_{ij}\,(n_i,r_i) \leq G_j \qquad j=1,2,\ldots m \tag{3.8}$$

and non-negative constraints.

$$1 \geq r_i \geq 0 \qquad i=1,2,\ldots N \tag{3.9}$$

The problem can be extended to 1-out-of-n active redundancy module with dissimilar units.

The above problem constitutes a non-linear programming problem.

(3) Mixed Optimal Problem

A combination of problems (1) and (2) can be stated as follows:

Maximize the system reliability $R(\underline{n},\underline{r})$ with respect to the number of redundant units ($\underline{n}$) and unit reliabilities ($\underline{r}$) subject to linear or non-linear cost constraints.

This is a non-linear mixed integer programming problem.

System Availability and System Profit

System availability can be improved by adding redundancy to the system and/or by performing preventive maintenance on the system according to some prescribed schedule [5,49].

In the absence of scheduled preventive maintenace, the system availability A is

$$A(\underline{n}) = \frac{MTBF}{MTBF + MDT} = \frac{\int_0^\infty R(\underline{n},t)\,dt}{\int_0^\infty R(\underline{n},t)dt + MDT} \tag{3.10}$$

where MTBF is the mean time to system failure, MDT is system mean downtime or mean repair time.

Now suppose preventive maintenance is performed on the system every "T" hours of continuing operation. If the system fails before "T" hours have elapsed, emergency maintenance is performed at that time. Preventive maintenance is then rescheduled. We assume that the system is as good as new after any type of maintenance, scheduled or emergency, and that the system either operates at full capacity or is down for maintenance. Let,

$f(\underline{n},t)$ = system failure time probability density function

T_E = mean time to perform emergency maintenance on the system

T_S = mean time to perform scheduled maintenance on the system.

Thus, mean time between system maintenance is given by:

$$\begin{aligned} MTBM &= T * R(\underline{n},t) + \int_0^T t\, f(\underline{n},t)dt \\ &= \int_0^T R(\underline{n},t)dt \end{aligned} \tag{3.11}$$

Also, assuming that T_E and T_S are independent of $\underline{n}$ and T, the mean down time for the system is:

$$\begin{aligned} MDT &= T_E\,(1 - R(\underline{n},T)) + T_S * R(\underline{n},t) \\ &= T_E - (T_E - T_S) * R(\underline{n},t) \end{aligned} \tag{3.12}$$

System availability is then defined as

$$A\,(\underline{n},T) = \frac{\int_0^T R(\underline{n},t)dt}{\int_0^T R(\underline{n},T)dt + T_E - (T_E - T_S)R(\underline{n},t)} \tag{3.13}$$

In many situations, system profit is closely related to system availability. To examine this relationship, we define the following terms:

T^+ = total life time of the system in hours

T_u = total time during which the system is operating at full capacity

$$T_u = A(\underline{n},T) * T^+ \tag{3.14}$$

T_D = total time during which the system is down

$$T_D = (1-A(\underline{n},T))\,T^+ \tag{3.15}$$

$$T^+ = T_u + T_D \tag{3.16}$$

C_{Fi} = fixed capital cost of a unit in the ith module

C_{TF} = total fixed capital investment

$$C_{TF} = \sum_{i=1}^{N} n_i C_{Fi} \tag{3.17}$$

C_M = system maintenance cost per hour of system downtime. This can be taken as some percentage, X, of fixed capital investment.

$$C_M = \left(\sum_{i=1}^{N} n_i C_{Fi}\right) \frac{X}{100} \tag{3.18}$$

C_{TM} = total maintenance costs for a system lifetime of T^+ hours

$$C_{TM} = C_M * T_D \tag{3.19}$$

C_I = net income per hour; the difference between the selling price of the product and all expenses other than the maintenance costs and capital costs.

C_{TI} = total net income during system lifetime of T^+ hours.

$$C_{TI} = C_I * T_u \tag{3.20}$$

We can now write a cash balance for the system lifetime period of T^+ hours.

Profit = net income - capital cost - maintenance cost

or

$$P = C_{TI} - C_{TF} - C_{TM}$$

or

$$P = C_I * T_u - \sum_{i=1}^{N} n_i C_{Fi} - C_M * T_D$$

or

$$P = C_I * A(\underline{n},T) * T^+ - \sum_{i=1}^{N} n_i C_{Fi} - \left(\sum_{i=1}^{N} n_i C_{Fi}\right) * \frac{X}{100} \; (1 - A(\underline{n},T))T^+ \tag{3.21}$$

The equation (3.21) does not take into account the cost of capital, the time value of money, or any other of the widely used economic methods.

Two types of optimization problems based on system availability are how formulated:

(1) Availability Maximization

The problem can be stated as follows:

Maximize the system availability

$$A(\underline{n},T) = \frac{\int_0^T R(\underline{n},t)dt}{\int_0^T R(\underline{n},t)dt + T_E - (T_E - T_S)R(\underline{n},T)} \tag{3.22}$$

subject to the cost constraint

$$C_{TF} + C_{TM} \leq C_{AO}$$

or

$$\left(\sum_{i=1}^{N} n_i C_{Fi}\right)\left(1 + \frac{X}{100}(1 - A(\underline{n},T)\ T^+\right) \leq C_{AO} \tag{3.23}$$

and non-negative and integer constraints

$$\begin{aligned} n_i &\geq k_i \qquad\qquad i = 1,2,\ldots N \\ n_i &: \text{integer} \\ T &\geq 0 \end{aligned} \tag{3.24}$$

Other constraints, if present, can also be included.

This is a non-linear mixed integer programming problem.

(2) Profit Maximization

This problem can be stated as follows:

Maximize the system profit over its lifetime of T^+ hours

$$P = C_I * A(\underline{n},T) * T^+ - \left(\sum_{i=1}^{N} n_i C_{Fi}\right)\left(1 + (1-A(\underline{n},T))\frac{X}{100} * T^+\right) \tag{3.25}$$

subject to the cost constraint

$$C_{TF} \leq C_{PO}$$

or

$$\sum_{i=1}^{N} n_i C_{Fi} \leq C_{PO} \tag{3.26}$$

and non-negative and integer constraints

$$\begin{aligned} n_i &\geq k_i \qquad\qquad i = 1,2,\ldots N \\ n_i &: \text{integer} \\ T &\geq 0 \end{aligned} \tag{3.27}$$

This is also a non-linear mixed integer programming problem.

Method of Solution

There are few existing methods that can solve non-linear integer and mixed integer programming problems. The two methods considred here are (1) a pseudo-Boolean programming method based on partial enumeration developed by Lawler and Bell [32]. The method is used to solve the non-linear integer programming problems formulated in Chapter IV. The details of the algorithm are discussed in Appendix F. (2) The other method proposed by Reiter and Rice (R-R) [50] is based on the modified integer gradients of the objective function. It can be extended to solve non-linear mixed integer programming problems. The method is effectively applicable to the problems formulated earlier since we can easily generate the gradients of the objective function or the sensitivities in our terminology.

The R-R algorithm is used to solve the system optimization problems formulated in this chapter. The solution procedure has three phases:

Phase 1: Setting up of an initial point $\underline{X}_I$ using uniformly distributed random numbers.

Phase 2: Searching for a feasible solution $\underline{X}^+$ of the problem, starting with $\underline{X}_I$. A search by the method of weighted perpendicular is performed. If the feasible solution is not obtained in a given number of iterations, the search is terminated, and we go back to Phase 1 of the search algorithm.

Phase 3: Locating a point $\underline{X}_p$ which maximizes the objective function in the feasible region round $\underline{X}^+$. This involves calculating the modified gradients of the objective function which define the path on which we search for improved values of the objective function.

$\underline{X}_I$, $\underline{X}^+$, and $\underline{X}_p$ denote vectors of variables, real and/or integer.

Details of the search algorithm are discussed in Appendix D. The module reliability and sensitivity expressions are given in Appendix A, and the sensitivity expressions for system availability and system profit are shown in Appendix E.

The R-R solution procedure provides locally maximal points. Repeated use of the algorithm, with judicious improvement in the lower and upper bounds on the system variables based on the previous results, gives better results, and, hopefully, the global optimum. The method is illustrated by the following examples:

Illustrative Examples

Solutions are now obtained for some of the optimization problems formulated earlier in this chapter. Two differeint module reliability graphs are considered. The module reliability graph of Figure 3.2 shows a series-parallel-series system and that of Figure 3.3 represents a non-series-parallel system. Henceforth, these will be referred to as System 1 and System 2, respectively. For the sake of illustration it is assumed that

all modules have 1-out-of-n_i active redundancy in the following examples:

Example (1) Optimal allocation of redundancy to maximize the system reliability.

Tables 3.1 and 3.3 give the unit reliability, unit, cost, and maximum number of units (upper bound in n_i) for each module of System 1 and System 2, respectively. These upper bounds on n_i's were obtained from an initial analysis of the problem. There is a linear cost constraint of the type

$$\sum_{i=1}^{N} C_i n_i \leq C \tag{3.28}$$

where C is the total allowable cost. C_i is the cost per unit of the ith module. The allowable cost C is given in Table 3.1 and Table 3.3 for System 1 and System 2, respectively.

The problem then is to maximize the system reliability subject to the cost constraint of Equation (3.28).

Discussion of Results: Some of the locally optimal results obtained are tabulated in Tables 3.2 and 3.4 and correspond to System 1 and System 2, respectively. The problem was also solved for comparison purposes using the partial enumeration algorithm of Lawler and Bell which finds a unique optimal solution or the global optimum. It was observed that the R-R algorithm not only found locally maximal points but the best solution obtained, Result 10 in Tables 3.2 and 3.4, was also the global optimum.

Example (2) Optimal allocation of redundancy and maintenance interval to maximize the system availability or the system profit.

Consider the module reliability graph of System 1. Assume that units in each module have exponential failure pdf. Table 3.5 gives the capital cost, failure rate data, and upper bounds on n_i for identical units of each module.

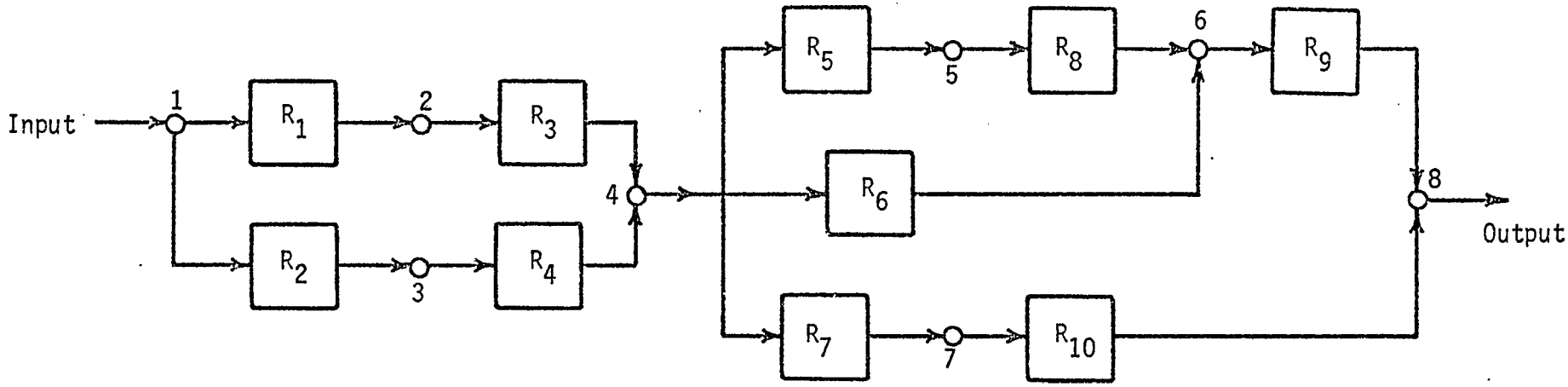


Figure 3.2

Reliability Graph - System 1

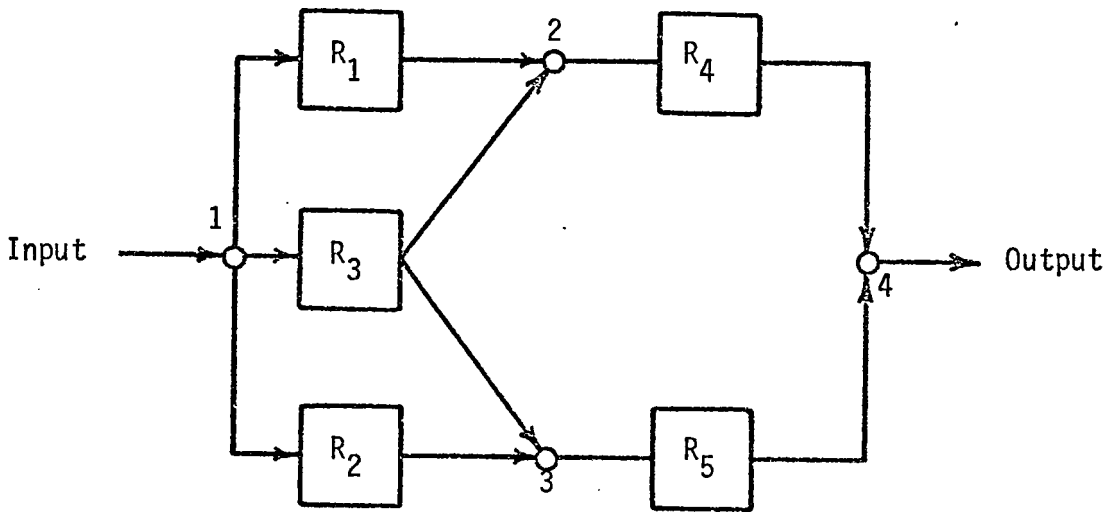


Figure 3.3

Reliability Graph - System 2

MODULE i	UNIT COST C_i	UNIT RELIABILITY r_i	UPPER BOUND $n_{i(max)}$
1	75.00	0.75	4
2	90.00	0.90	4
3	80.00	0.80	4
4	87.00	0.87	4
5	72.00	0.72	4
6	88.00	0.88	4
7	75.00	0.75	4
8	89.00	0.89	4
9	75.00	0.75	4
10	80.00	0.80	4

Allowable Cost = 1300.00

Table 3.1

Reliability Optimization Data - System 1

MODULE	Optimum Number of Units in Each Module									
	Result 1	Result 2	Result 3	Result 4	Result 5	Result 6	Result 7	Result 8	Result 9	Result 10
1	2	2	2	2	1	2	1	1	1	1
2	2	2	1	2	2	1	2	2	2	2
3	2	2	2	1	1	2	1	1	2	1
4	1	1	2	2	2	2	2	2	2	2
5	1	1	1	1	2	1	2	2	1	1
6	1	2	1	1	1	2	1	2	1	1
7	2	1	2	2	1	1	2	1	2	3
8	1	1	1	1	2	1	1	1	1	1
9	2	3	2	3	3	3	2	3	2	2
10	2	1	2	1	1	1	2	1	2	2
COST UTILIZED	1286.0	1294.0	1283.0	1288.0	1299.0	1291.0	1290.0	1298.0	1298.0	1293.0
MAXIMUM RELIA-BILITY	0.975982	0.977958	0.978306	0.978900	0.978938	0.980288	0.981123	0.982116	0.982301	0.983708

Table 3.2

Reliability Optimization - System 1

MODULE i	UNIT COST C_i	UNIT RELIABILITY r_i	UPPER BOUND n_i
1	7000.	0.90	3
2	3000.	0.70	4
3	5000.	0.70	4
4	3000.	0.70	6
5	7000.	0.90	4

Allowable Cost = 50,000.00

Table 3.3

Reliability Optimization Data - System 2

MODULE	Optimum Number of Units in Each Module									
	Result 1	Result 2	Result 3	Result 4	Result 5	Result 6	Result 7	Result 8	Result 9	Result 10
1	2	1	1	1	1	1	1	1	2	2
2	2	3	3	3	1	2	2	3	1	3
3	1	2	3	1	3	2	3	2	2	1
4	3	1	4	5	6	4	5	3	5	5
5	2	3	1	2	1	2	1	2	1	1
COST UTILIZED	48,000	50,000	50,000	50,000	50,000	49,000	50,000	49,000	49,000	50,000
MAXIMUM RELIA-BILITY	0.998717	0.998792	0.998848	0.998856	0.998919	0.998966	0.999262	0.999340	0.999364	0.999364

Table 3.4

Reliability Optimization - System 2

Other data are:

C_I = \$10.00/hour

X = 0.02%

T^+ = 80,000 hours

T_E = 400 hours

T_S = 100 hours

C_{AO} = \$350,000

C_{PO} = \$200,000

Discussion of Results: The results of the ten randomly sampled points for the system availability maximization and the system profit maximization are tabulated in Tables 3.6 and 3.7, respectively. Also, the solutions obtained at each step in the optimization procedure are shown in Tables 3.8 and 3.9 for the Result 1 of each problem. The results show that in the case of system availability maximization the local optima are dependant on the starting point as far as n_i's are concerned. This is due to the system availability being a monotonically increasing function with respect to the n_i's, the gradient always points forward increasing n_i's, and thus, the solution lies on or near the constraint boundary. The system profit maximization problem, on the other hand, gives fewer local optima and the solution does not necessarily lie on the constraint boundary. The method is thus seen to be more effective in the case of a problem with non-monotonically increasing objective function and/or the optimum solution being in the interior of the constraint boundary.

The computer time to find a single locally maximal point is about one minute on the IBM 360 and fifteen seconds on the UNIVAC 1108.

MODULE i	UPPER BOUND $n_{i(max)}$	CAPITAL COST/UNIT (C_{Fi}) \$/UNIT	FAILURE RATE (λ_i) HOUR^{-1}
1	6	5000.00	$3.0 * 10^{-4}$
2	6	15000.00	$1.0 * 10^{-4}$
3	6	10000.00	$2.2 * 10^{-4}$
4	6	15000.00	$1.2 * 10^{-4}$
5	6	5000.00	$3.2 * 10^{-4}$
6	6	15000.00	$1.1 * 10^{-4}$
7	6	5000.00	$3.0 * 10^{-4}$
8	6	15000.00	$1.0 * 10^{-4}$
9	12	5000.00	$3.0 * 10^{-4}$
10	6	10000.00	$2.2 * 10^{-4}$

Table 3.5

Availability or Profit Maximization Data - System 1

MODULE NUMBER	OPTIMUM NUMBER OF UNITS IN EACH MODULE									
	Result 1	Result 2	Result 3	Result 4	Result 5	Result 6	Result 7	Result 8	Result 9	Result 10
1	5	1	2	2	1	2	1	1	3	1
2	2	3	3	2	2	2	2	3	2	3
3	2	1	1	1	1	1	1	1	1	1
4	2	3	3	3	3	3	3	3	2	3
5	1	2	1	2	1	3	1	1	1	1
6	2	2	2	2	2	1	2	2	2	2
7	2	1	2	2	4	4	3	2	4	3
8	1	1	1	1	1	1	1	1	1	1
9	6	8	6	8	5	4	8	7	7	6
10	1	1	1	1	2	2	1	1	1	1
Fixed Capital Cost ($)	205,000	215,000	210,000	210,000	205,000	200,000	205,000	210,000	200,000	210,000
Total Maintenance Cost ($)	140,822	134,648	138,428	139,485	142,286	146,922	141,368	135,422	145,765	137,502
Optimum Maintenance Interval (Hours)	3,190	3,575	3,375	3,490	3,380	3,030	3,510	3,490	3,260	3,430
Optimized System Availability	0.957066	0.960858	0.958801	0.958487	0.956620	0.954087	0.956900	0.959696	0.954448	0.959077
System Profit ($)	419,831	419,038	418,613	417,304	418,010	416,347	419,153	422,334	417,794	419,759

Table 3.6

Availability Optimization - System 1

MODULE NUMBER	OPTIMUM NUMBER OF UNITS IN EACH MODULE									
	Result 1	Result 2	Result 3	Result 4	Result 5	Result 6	Result 7	Result 8	Result 9	Result 10
1	3	3	1	3	3	3	1	1	3	3
2	1	1	2	1	1	1	2	2	1	1
3	2	2	1	2	2	2	1	1	2	2
4	1	1	2	1	1	1	2	2	1	1
5	1	1	1	1	1	1	1	2	1	2
6	1	1	2	1	1	1	1	1	1	1
7	2	2	1	2	2	2	3	2	2	1
8	1	1	1	1	1	1	1	1	1	1
9	3	3	5	3	3	4	3	4	4	4
10	1	1	1	1	1	1	2	1	1	1
Fixed Capital Cost ($)	135,000	135,000	165,000	135,000	135,000	140,000	160,000	155,000	140,000	140,000
Total Maintenance Cost ($)	155,112	155,138	143,377	155,116	155,130	153,017	148,688	150,276	153,045	152,779
Optimum Maintenance Interval (Hours)	2,145	2,170	2,840	2,150	2,170	2,240	2,560	2,580	2,290	2,190
Optimized System Profit ($)	452,439	452,404	448,176	452,433	452,415	452,334	444,847	446,248	452,297	452,657
Availability	.928189	.928177	.945691	.928187	.928181	.931689	.941919	.939405	.931676	.931795

Table 3.7

Profit Optimization - System 1

MODULE NUMBER	1	2	3	4	5	6	7	8	9	10	MAINTENANCE INTERVAL (Hours)	COST CONSTRAINT ($)	SYSTEM AVAILABILITY ($)
Initial Point (1)	5	2	6	6	1	3	2	3	10	4	2131	+338,723	--
Feasible Point (2)	5	1	1	2	1	1	1	1	2	1	2484	- 11,563	0.911412
(3)	5	1	1	2	1	1	1	1	3	1	2481	- 24,423	0.922165
(4)	5	1	1	2	1	2	1	1	3	1	2478	- 14,032	0.931263
(5)	5	2	1	2	1	2	1	1	3	1	2478	- 6,748	0.939910
(6)	5	2	1	2	1	2	1	1	4	1	2480	- 13,996	0.945832
(7)	5	2	2	2	1	2	1	1	4	1	2490	- 4,856	0.948966
(8)	5	2	2	2	1	2	1	1	5	1	2503	- 5,588	0.952112
(9)	5	2	2	2	1	2	1	1	6	1	2539	- 2,652	0.953954
(10)	5	2	2	2	1	2	1	1	6	1	2639	- 4,518	0.954537
(11)	5	2	2	2	1	2	2	1	6	1	2690	- 271	0.955875
(12)	5	2	2	2	1	2	2	1	6	1	2790	- 1,734	0.956321
(13)	5	2	2	2	1	2	2	1	6	1	2890	- 2,824	0.956654
(14)	5	2	2	2	1	2	2	1	6	1	2990	- 3,574	0.956882
(15)	5	2	2	2	1	2	2	1	6	1	3090	- 4,016	0.957017
Locally Optimum Point (16)	5	2	2	2	1	2	2	1	6	1	3190	- 4,178	0.957066

Table 3.8
Steps to Reach Locally Optimal Solution - Result 1 of Table 3.6

MODULE NUMBER	1	2	3	4	5	6	7	8	9	10	MAINTENANCE INTERVAL (Hours)	COST CONSTRAINT ($)	SYSTEM PROFIT
Initial Point (1)	5	2	6	6	1	5	3	5	10	6	1131	+285,000	--
Feasible Point (2)	3	1	2	1	1	1	1	1	8	2	1131	- 35,000	330,717
(3)	3	1	2	1	1	1	1	1	8	1	1211	- 45,000	363,167
(4)	3	1	1	1	1	1	1	1	8	1	1298	- 55,000	366,674
(5)	3	1	1	1	1	1	1	1	7	1	1380	- 60,000	386,472
(6)	3	1	1	1	1	1	1	1	6	1	1448	- 65,000	403,457
(7)	3	1	1	1	1	1	1	1	5	1	1507	- 70,000	418,106
(8)	3	1	1	1	1	1	1	1	4	1	1558	- 75,000	429,769
(9)	3	1	1	1	1	1	1	1	3	1	1606	- 80,000	435,455
(10)	2	1	1	1	1	1	1	1	3	1	1640	- 85,000	439,400
(11)	2	1	2	1	1	1	1	1	3	1	1771	- 75,000	445,012
(12)	3	1	2	1	1	1	1	1	3	1	1870	- 70,000	448,389
(13)	3	1	2	1	1	1	2	1	3	1	1945	- 65,000	451,396
(14)	3	1	2	1	1	1	2	1	3	1	2045	- 65,000	452,222
Locally Optimum Point (15)	3	1	2	1	1	1	2	1	3	1	2145	- 65,000	452,439

Table 3.9

Steps to Reach Locally Optimal Solution - Result 1 of Table 3.7

Conclusions

The efficient computer algorithm developed in Chapter II to calculate the reliability and the sensitivities of complex systems is used in conjunction with the R-R modified integer gradient method for system optimization. The proposed method is easily applicable to series-parallel-series or non-series-parallel systems and can handle multiple linear and/or non-linear constraints. Although this method does not insure a global optimum, it does find various near-optimum solutions. From a practical consideration, this could provide for a wider choice during the design phase.

The method presented above has been programmed in FORTRAN IV. In addition to the eight basic subroutines, there are seven others which are used during system optimization. Appendix G contains a description of each subroutine, including a listing of the corresponding FORTRAN program and a schematic diagram of the main program.

CHAPTER IV

OPTIMAL RELIABILITY DESIGN OF PROCESS SYSTEMS

In the design of electrical, electronic and safety systems, the flow of information is the most important factor, whereas in process systems, the flow of materials is also very important. In this chapter an optimal reliability design of process systems is proposed which takes into account the quantitative aspect of systems throughput or capacity. It is based on the k-out-of-n configuration instead of the conventional parallel 1-out-of-n configuration. The problem is so formulated that determining the optimum k-out-of-n configuration also determines the optimum capacity of units in each stage of the system.

Basic Idea

Consider a pump system which must deliver at least L gallons per second. If we install one pump of capacity L and cost C_L, the reliability of the system is that of the single pump. If we appropriately install N pumps of capacity L in parallel, the reliability will increase but so will the cost which will go up by a factor of N as shown in Figure 4.1. If the allowed cost is more than C_L but less than $2C_L$, the reliability cannot be increased by the conventional parallel redundancy configuration.

On the other hand, if we install three pumps each of capacity L/2, and assuming for the moment that the cost is proportional to capacity, this can be achieved at a cost of 1.5 C_L which is less than the allowed cost of $2C_L$. The system can now operate if at least any two out of the three pumps function as shown in Figure 4.2. This gives rise to the well-known k-out-of-n configuration, and the system has a higher reliability compared to a single pump of capacity L. Here it is assumed that

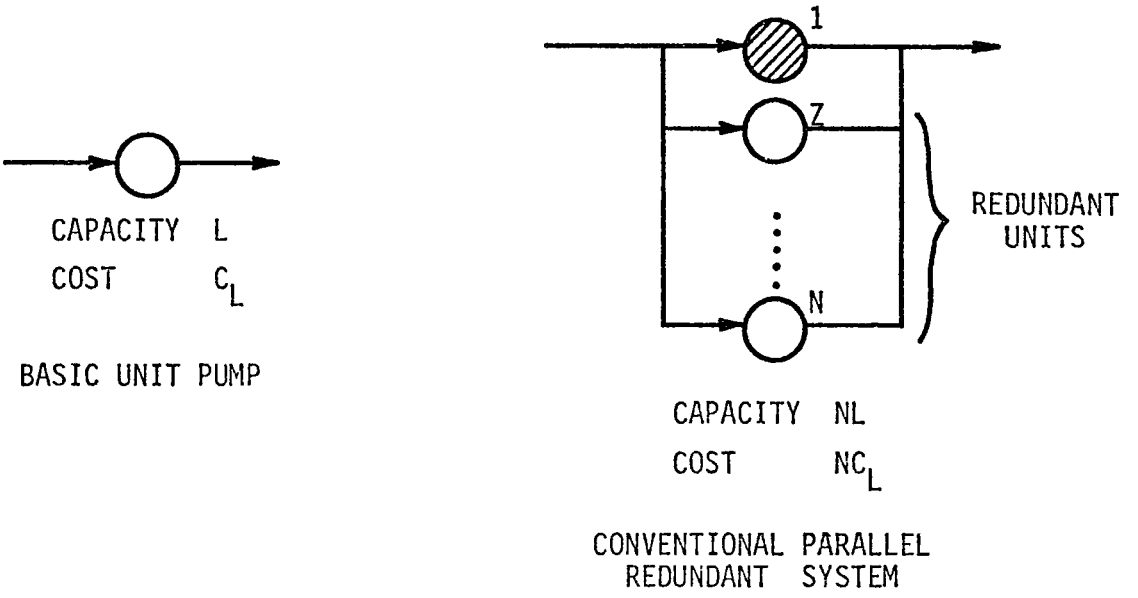


Figure 4.1

Conventional Parallel Configuration

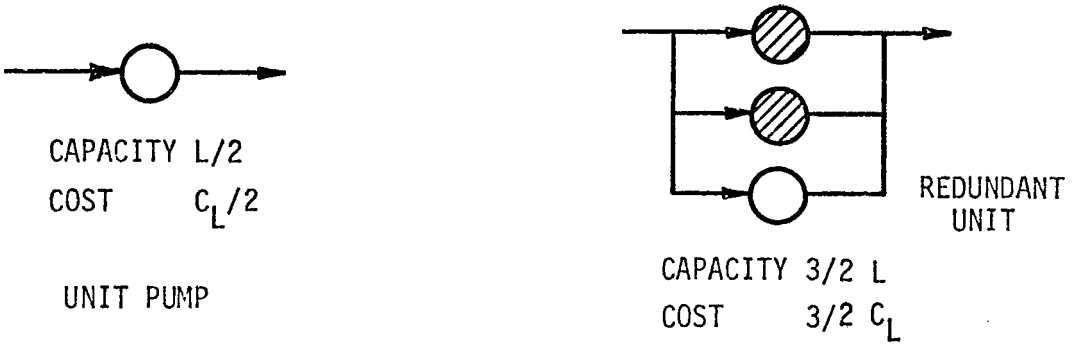


Figure 4.2

2-Out-Of-3 Configuration

(1) the reliability of a unit is independent of its capacity; and,

(2) the pump system has perfect valving.

There are many system configurations of this type which cost less than 2 C_L. As calculated from Equation (4.1) the reliabilities of some of these configurations are listed in Table 4.1. One can clearly see the possibility of increasing the system reliability by this means. It is important to note that changes in k affect both the reliability of the system and the capacity of units in the system.

Problem Formulation

Although the basic idea can be extended to complex systems, the attention is first restricted to the optimal design of a single stage. Consider a system which operates if at least k out of n units function. This system configuration is called a k-out-of-n configuration. The two types of configurations considered are: (1) an active k-out-of-n configuration, and (2) a standby k-out-of-n configuration.

1. Active k-out-of-n Configuration

All the n idential units in parallel are operating independent of each other. The probability that at least k-out-of-n units are operating is given by (see Appendix A):

$$R_A\ (n,k;p) = \sum_{l=k}^{n} \binom{n}{l} p^l\ (1-p)^{n-l} \tag{4.1}$$

where p is the reliability of a single unit.

2. Standby k-out-of-n Configuration

If k units are functioning while the other (n-k) units in standby have zero failure probability, the system is called a cold standby k-out-of-n configuration. Assuming that all units are identical and have exponential

		SYSTEM RELIABILITY (ACTIVE)				
Configuration		3-out-of-4	2-out-of-3	4-out-of-6	3-out-of-5	4-out-of-7
Cost		1.33 C_L	1.5 C_L	1.5 C_L	1.67 C_L	1.75 C_L
UNIT RELIABILITY	0.95	0.9860	0.9928	0.9978	0.9988	0.9998
	0.85	0.8905	0.9392	0.9527	0.9734	0.9879
	0.75	0.7383	0.8438	0.8306	0.8965	0.9294

Cost is proportional to capcity; valving is perfect; reliability of a unit is independent of its capacity.

Table 4.1

System reliability comparison for several k-out-of-n configurations

failure probability density function, the reliability expression is (see Appendix A):

$$R_C\ (n,k;p) = \sum_{j=0}^{n-k} p^k \frac{(-k\ln p)^j}{j!} \tag{4.2}$$

where $p=e^{-\lambda t}$ is the probability of success in service of identical units with constant hazard rate λ for an operating time t.

If the units in standby have the same failure probability as those in service, it is called a hot standby system. The reliability expression for a hot standby k-out-of-n configuration is the same as Equation (4.1) of the active k-out-of-n configuration; that is,

$$R_H\ (n,k;p) = R_A(n,k;p) \tag{4.3}$$

It is assumed that any one of the three configurations is allowed as a subsystem of the total system. The use of a particular configuration depends on the properties and the purpose of the subsystem.

System Cost

The assumption that the cost is proportional to the capacity is somewhat unrealistic. The cost-capacity relationship used here is given by

$$\text{cost} = K\ (\text{capacity})^s \tag{4.4}$$

where K and s are positive constants and $0 < s < 1$.

If k units are necessary to support the minimum required capacity L, then one unit must support the capacity L/k. Therefore, the total cost of the system which consists of such n units is:

$$\text{cost} = n \cdot K \cdot \left(\frac{L}{k}\right)^s \tag{4.5}$$

or

$$\text{cost} = n \cdot k_c \cdot k^{-s} \tag{4.6}$$

where k_c is the cost of a unit with capacity L.

System Optimization

The two types of problems considered are (1) reliability maximization, and, (2) cost minimization.

1. Reliability Maximization

If the amount $l \cdot k_c$ can be spent to improve the reliability of the system, the cost constraint is

$$n \cdot k_c \cdot k^{-s} \leq l \cdot k_c \tag{4.7}$$

or

$$l - n \cdot k^{-s} \geq 0 \tag{4.8}$$

where l is a positive real number greater than 1.

The optimization problem can then be stated as follows:

Select n and k to maximize the system reliability given in either Equation (4.1), (4.2) or (4.3) subject to the cost constraint given in Equation (4.8) and the inherent constraints

$$n - k \geq 0 \tag{4.9}$$

$$n \text{ and } k\text{: integers not less than 1} \tag{4.10}$$

This is a nonlinear integer programming problem with a nonlinear objective function and two constraints, a nonlinear and a linear.

2. Cost Minimization

When it is necessary to minimize the cost under the condition that the system reliability is not less than a preassigned value, say R*, the optimization problem can then be stated as follows:

Select n and k to minimize the system cost given in Equation (4.6) subject to the constraint that the reliability given in either Equation (4.1), (4.2) or (4.3) is not less than a preassigned value R* and also satisfies the inherent constraints given in Equations (4.9) and (4.10).

This is again a nonlinear integer programming problem with a nonlinear objective function and two constraints, a nonlinear and a linear.

In some cases in addition to the cost constraint in the reliability maximization problem or the reliability constraint in the cost minimization problem, nonlinear weight and/or volume constraints may be present. A weight constraint may be of the form

$$W = n \cdot k_W \cdot k^{-s_W} \leq W^* \tag{4.11}$$

and a volume constraint may be

$$V = n \cdot k_V \cdot k^{-s_V} \leq V^* \tag{4.12}$$

where k_W and k_V are constants corresponding to the standard weight and volume of a single unit of capacity L. s_W and s_V are non-negative constants less than one. W^* and V^* are maximum allowable weight and volume, respectively.

A general problem of this type is, therefore, a nonlinear integer programming problem with a nonlinear objective function, several nonlinear constraints, and a linear constraint.

Extension to Series Parallel and Complex Systems

Consider a series-parallel system with N stages as shown in Figure 4.3. The ith stage is assumed to be a k_i-out-of-n_i configuration in either active, cold standby, or hot standby condition.

The total reliability of the sytem is expressed by

$$R_T\ (\underline{n},\underline{k};\underline{p}) = \prod_{i=1}^{N} \left[R_A(n_i,k_i;p_i),\ R_C\ (n_i,k_i;p_i) \text{ or } R_H\ (n_i,k_i;p_i)\right] \tag{4.13}$$

where p_i is the unit reliability of the ith stage.

The total system cost is

$$\text{Total cost} = \sum_{i=1}^{N} n_i \cdot k_{ci} \cdot k_i^{-s_i} \tag{4.14}$$

Similarly, the total weight and volume are

$$\text{Total weight} = \sum_{i=1}^{N} n_i \cdot k_{wi} \cdot k_i^{-s_{wi}} \tag{4.15}$$

and

$$\text{Total volume} = \sum_{i=1}^{N} n_i \cdot k_{vi} \cdot k_i^{-s_{vi}} \tag{4.16}$$

When the system to be considered is a complex system, as shown in Figure 4.4, and has several paths between input and output, the total reliability is (see Chapter II):

$$\begin{aligned} R_T\ (\underline{n},\underline{k};\underline{p}) &= \sum_{i=1}^{M} \prod_{l\varepsilon P_i} (R_{Al},\ R_{Cl},\ \text{or } R_{Hl}) \\ &- \sum_{i=1}^{M} \sum_{j>i}^{M} \prod_{l\varepsilon P_i UP_j} (R_{Al},\ R_{Cl},\ \text{or } R_{Hl}) + \ldots \\ &+ (-1)^{M-1} \prod_{\substack{l\varepsilon UP_i \\ i=1}}^{M} (R_{Al},\ R_{Cl},\ \text{or } R_{Hl}) \end{aligned} \tag{4.17}$$

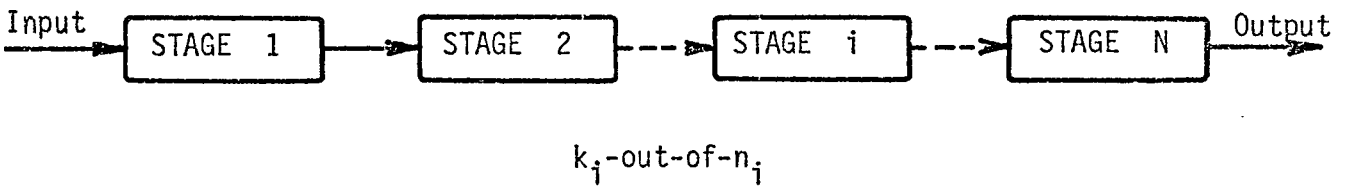


Figure 4.3

Series-Parallel System with N Stages

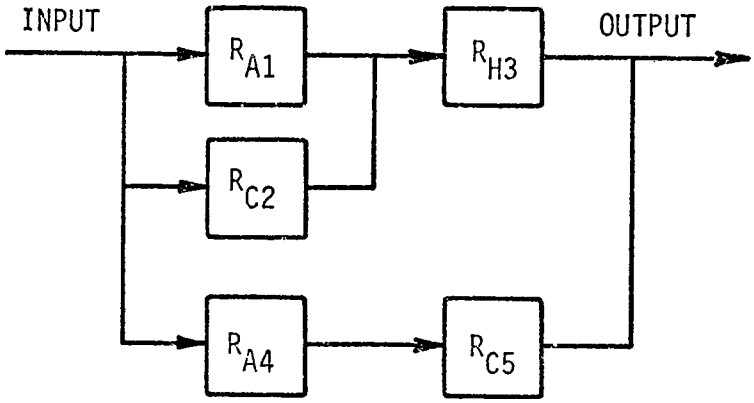


Figure 4.4

Example of a Complex System

where P_i is a path, M is the total number of paths, and U represents path unions.

In the case of Figure 4.4, we have

$$R_T\ (\underline{n},\underline{k};\underline{p}) = R_{A1} \cdot R_{H3} + R_{C2} \cdot R_{H3} + R_{A4}\ R_{C5}$$
$$- (R_{A1} \cdot R_{C2} \cdot R_{H3} + R_{A1} \cdot R_{H3} \cdot R_{A4} \cdot R_{C5} + R_{C2} \cdot R_{H3} \cdot R_{A4} \cdot R_{C5}) \qquad (4.18)$$
$$+ R_{A1} \cdot R_{C2} \cdot R_{H3} \cdot R_{A4} \cdot R_{C5}$$

Since total cost, weight, and volume for the complex system can also be described by Equations (4.14), (4.15) and (4.16), respectively, it is easy to construct a reliability maximization or a cost minimization problem for a complex system. The computer algorithm which can calculate the reliability expression from a given reliability graph has been described in Chapter II.

Solution Method

Two methods have been considered for solving the non-linear integer programming problems. One is the integer gradient method proposed by Reiter and Rice [50] and used to solve the system optimization problems formulated in Chapter III. The other is the pseudo-Boolean programming method based on partial enumeration developed by Lawler and Bell (L-B) [32].

The L-B algorithm is used to solve the system optimization problem formulated in this chapter. It is applicable to any problem that can be put into the form

Minimize

$$g_0\ (\underline{x}) \qquad (4.19)$$

subject to

$$g_{i1}(\underline{x}) - g_{i2}(\underline{x}) \geq 0 \quad i = 1,2,\ldots m \tag{4.20}$$

where

$$x = (x_1, x_2, \ldots x_n) \tag{4.21}$$

and

$$x_j = 0 \text{ or } 1 \qquad j = 1,2,\ldots n \tag{4.22}$$

and, where the restriction is applied, each of the functions g_0, g_{i1}, g_{i2} ($i = 1,2,\ldots m$) is monotone non-decreasing in each of the variables $x_1, x_2, \ldots x_n$.

Consider the single stage reliability maximization problem. It is noted from physical reasoning that the system reliability to be maximized, R_A, R_H, or R_C, is monotone increasing with respect to n and monotone decreasing with respect to k. Mathematical proof of this property is shown below for R_A or R_H. By use of the formula

$$\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k} \qquad k \leq n \tag{4.23}$$

we have

$$R_A\,(n+1,k;p) - R_A\,(n,k;p) = \binom{n}{k-1} p^k\,(1-p)^{n+1-k} \geq 0 \tag{4.24}$$

Also,

$$R_A\,(n,k+1;p) - R_A\,(n,k;p) = -\binom{n}{k} p^k\,(1-p)^{n-k} \leq 0 \tag{4.25}$$

The objective function to be minimized, $g_0 = -R_A$, $-R_H$, or $-R_C$, is thus monotone decreasing with respect to n and monotone nondecreasing with respect to k. By the simple transformation of variable, $n = n_{max} - n'$

or $n' = n_{max} - n$, where n_{max} is an upper bound on n, we can make $g_0(n',k)$ monotone nondecreasing in both n' and k.

Non-negative integer variables can be transformed into binary variables by means of the substitution

$$n = 1 + x_{n1} + 2x_{n2} + 2^2x_{n3} + \ldots + 2^{j-1}x_{nj} \tag{4.26}$$

$$k = 1 + x_{k1} + 2x_{k2} + 2^2x_{k3} + \ldots + 2^{j-1}x_{kj} \tag{4.27}$$

where x_{ni} and x_{ki} are binary variables and j is chosen to be sufficiently large for 2^j to be an upper bound on the value of n.

Applying the substitution

$$x_{ni} = 1 - x'_{ni} \qquad i = 1,2,\ldots j \tag{4.28}$$

where x'_{ni} is a binary variable, we have

$$n = 1 + (1 - x'_{n1}) + 2(1 - x'_{n2}) + 2^2(1 - x'_{n3}) + \ldots + 2^{j-1}(1 - x'_{nj})$$

or (4.29)

$$n = 2^j - x'_{n1} - 2x'_{n2} - 2^2x'_{n3} - \ldots - 2^{j-1}x'_{nj}$$

Using Equations (4.27) and (4.29), the objective function $g_0(n,k)$ can be expressed in terms of x'_{ni}'s and x_{ki}'s, and it is obviously monotone decreasing in each of the variables x'_{ni} and x_{ki} $(i = 1,2,\ldots j)$.

The cost constraint of Equation (4.8) can then be rewritten as

$$1 - \frac{2^j - x'_{n1} - 2x'_{n2} - 2^2x'_{n3} - \ldots - 2^{j-1}x'_{nj}}{(1 + x_{k1} + 2x_{k2} + 2^2x_{k3} + \ldots + 2^{j-1}x_{kj})^s} \geq 0 \tag{4.30}$$

Comparing with Equation (4.20) we have

$$g_{11}(\underline{x}) = 1 - \frac{2^j - x'_{n1} - 2x'_{n2} - 2^2x'_{n3} - \ldots - 2^{j-1}x'_{nj}}{(1 + x_{k1} + 2x_{k2} + 2^2x_{k3} + \ldots + 2^{j-1}x_{kj})^s} \tag{4.31}$$

and

$$g_{12}(\underline{x}) = 0 \tag{4.32}$$

The volume and weight constraints can be converted into similar forms.

The linear constraint of Equation (4.9) can be written as

$$\begin{aligned}&(2^j - x'_{n1} - 2x'_{n2} - \ldots - 2^{j-1}x'_{nj})\\ &- (1 + x_{k1} + 2x_{k2} + \ldots + 2^{j-1}x_{kj}) \geq 0\end{aligned} \tag{4.33}$$

Again comparing with Equation (4.20), we have

$$g_{21}(\underline{x}) = 0 \tag{4.34}$$

and

$$\begin{aligned}g_{22}(\underline{x}) = &(x'_{n1} + 2x'_{n2} + 2^2x'_{n3} + \ldots + 2^{j-1}x'_{nj} - 2^j)\\ &+ (1 + x_{k1} + 2x_{k2} + 2^2x_{k3} + \ldots + 2^{j-1}x_{kj})\end{aligned} \tag{4.35}$$

Thus, the problem has been converted to a standard form to which the L-B algorithm is applicable. It is easily seen that the cost minimization problem can also be converted to the standard form. Similarly, complex multistage problems can be transformed and solved by the L-B algorithm.

The important features of the L-B algorithm are:

(1) the true optimum solution found;

(2) the extreme ease of programming;

(3) the almost complete absence of housekeeping operations (no lists, no pushdown storage routines);

(4) the very small amount of storage required; and,

(5) the wide applicability to fairly general nonlinear integer programming problems.

A major limitation of the L-B algorithm is the computational infeasibility when the number of binary variables is large, approximately more than 30. Details of the L-B algorithm are explained in Appendix F.

Illustrative Examples

1. Single Stage Problem

The single stage reliability maximization problem is solved for optimum n and k using the values

$$l = 1.8 \quad \text{and} \quad s = 0.85 \tag{4.36}$$

An upper bound on n is estimated by Equation (4.8), that is,

$$n \leq lk^s \underset{0<k\leq n}{\leq} lk^s_{max} = ln^s = 1.8\, n^{0.85} \tag{4.37}$$

This gives

$$n \leq 50 \tag{4.38}$$

The results with p as a parameter are shown in Table 4.2.

Unit Reliability	ACTIVE OR HOT STANDBY			COLD STANDBY		
	OPTIMUM		Maximum Reliability	OPTIMUM		Maximum Reliability
	n	k		n	k	
0.95	7	5	0.9962	7	5	0.9977
0.85	3	2	0.9392	3	2	0.9573
0.75	3	2	0.8437	3	2	0.8861

$l = 1.8$

$s = 0.85$

Table 4.2

Optimum solutions for single-stage problem

2. Two Stage Problem

Consider a two-stage reliability maximization problem as shown in Figure 4.5. The first stage is a cold standby k_1-out-of-n_1 configuration and the second stage is an active k_2-out-of-n_2 configuration.

The problem is to find the optimum values of n_1, k_1, n_2 and k_2 that maximize the total system reliability expressed by

$$R_T\,(n_1, n_2, k_1, k_2; p_1, p_2)$$

$$= \left[\sum_{j=0}^{n_1-k_1} \frac{p_1^{k_1}}{j!} (-k_1 \ln p_1)^j\right] \left[\sum_{l=k_2}^{n_2} \binom{n_2}{l} p_2^{\,l} (1-p_2)^{n_2-l}\right] \tag{4.39}$$

subject to

$$n_1 k_{c1} k_1^{-s_1} + n_2 k_{c2} k_2^{-s_2} \leq C^* \tag{4.40}$$

$$n_1 - k_1 \geq 0 \tag{4.41}$$

$$n_2 - k_2 \geq 0 \tag{4.42}$$

$$n_1, n_2, k_1, k_2\text{: integers not less than one} \tag{4.43}$$

C^*: the total allowable cost

In the case of multistage problems, there is no way to estimate appropriate upper bounds on the n's, so we write an extra constraint

$$n_1, n_2 \leq 16 \tag{4.44}$$

The results with p_1 and p_2 as parameters are shown in Table 4.3. For the extreme case of $s_1 = s_2 = 0$, the results correspond to the optimal

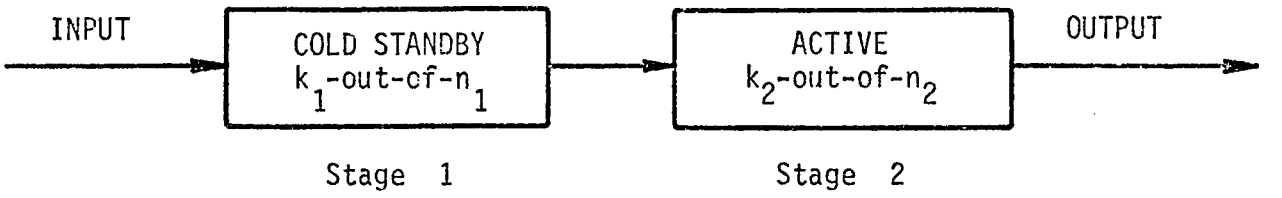


Figure 4.5

Two-Stage System

STAGE-1 (COLD STANDBY)					STAGE-2 (ACTIVE)					Cost	C*	R_T
P_1	k_{C1}	s_1	n_1	k_1	P_2	k_{C2}	s_2	n_2	k_2			
0.90	200	0.80	3	2	0.80	100	0.80	2	1	545	570	0.9415
0.90	200	0	1	1	0.80	100	0	3	1	500		0.8928
0.80	200	0.80	2	1	0.90	100	0.80	4	3	566		0.9273
0.80	200	0	2	1	0.90	100	0	1	1	500		0.8807
0.90	200	0.75	2	1	0.80	100	0.75	10	5	699	700	0.9885
0.90	200	0	2	1	0.80	100	0	3	1	700		0.9869
0.80	200	0.75	4	2	0.90	100	0.75	5	3	695		0.9809
0.80	200	0	2	1	0.90	100	0	3	1	700		0.9775

Table 4.3

Optimum solutions for two-stage problem

solutions for the conventional 1-out-of-n series-parallel configuration and are shown for comparison. The same results are obtained for a loosely guessed upper bounds on the n's.

$$n_1, n_2 \leq 32 \tag{4.45}$$

Therefore, the results shown in Table 4.3 are believed to be the optimal solutions without the extra constraint (4.44) or (4.45).

Conclusions

A new formulation of the optimal reliability design of process systems has been proposed, and is based on the k-out-of-n configuration. It is more suitable for reliability optimization of process systems than that based on the conventional 1-out-of-n parallel redundancy configuration.

An efficient computer program based on the Lawler and Bell [32] pseudo-Boolean algorithm has been developed to solve the above optimization problem. The program was written in FORTRAN IV. Appendix G contains a description of the program including a schematic diagram and a listing of the FORTRAN program.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

This dissertation consists of two main parts. First, an efficient general computer algorithm based on the path enumeration method has been developed to calculate reliability and sensitivity functions of complex systems. An important feature of the method is the module representation of the reliability graph. A modified integer gradient method is used for system optimization, which in general is a non-linear mixed integer programming problem subject to multiple linear or non-linear constraints. Although the method does not ensure a global optimum, it does find various near-optimum solutions. From a practical consideration, this could provide for a wider choice during the design phase. A number of examples are included to illustrate the method.

Second, a new formulation of the optimal reliability design of process systems is proposed. It takes into account the quantitative aspects of systems throughout and is based on the k-out-of-n configuration instead of the conventional parallel redundancy configuration. The problem is so formulated that determining the optimum configuration also determines the optimum capacity of units to be used at each stage of the system. A computer program based on the pseudo-Boolean algorithm of Lawler and Bell [32] is used to solve this non-linear integer programming problem. Several examples are included to illustrate the method.

With regard to further research in this area, the following two related problems are suggested:

1. Development of an algorithm to calculate exact reliability of systems which cannot be handled by path finding algorithm. This will be the case for systems with approximately more than 10 paths. The proposed method will first use the network reduction technique to reduce the reliability network to a non-series-parallel system and then apply the path enumeration technique to calculate the system reliability.

2. In the type of reliability network considered here it is assumed that modules are composed of different type of units. In practice, however, a particular type of unit may appear at many different locations in the system and it is usually desired to find the optimum unit reliability of each type to maximize the system objective function. Such a system is shown in Figure 5.1. The development of reliability and sensitivity expressions for this system and finally its optimization will be an extension of the techniques presented in this dissertation.

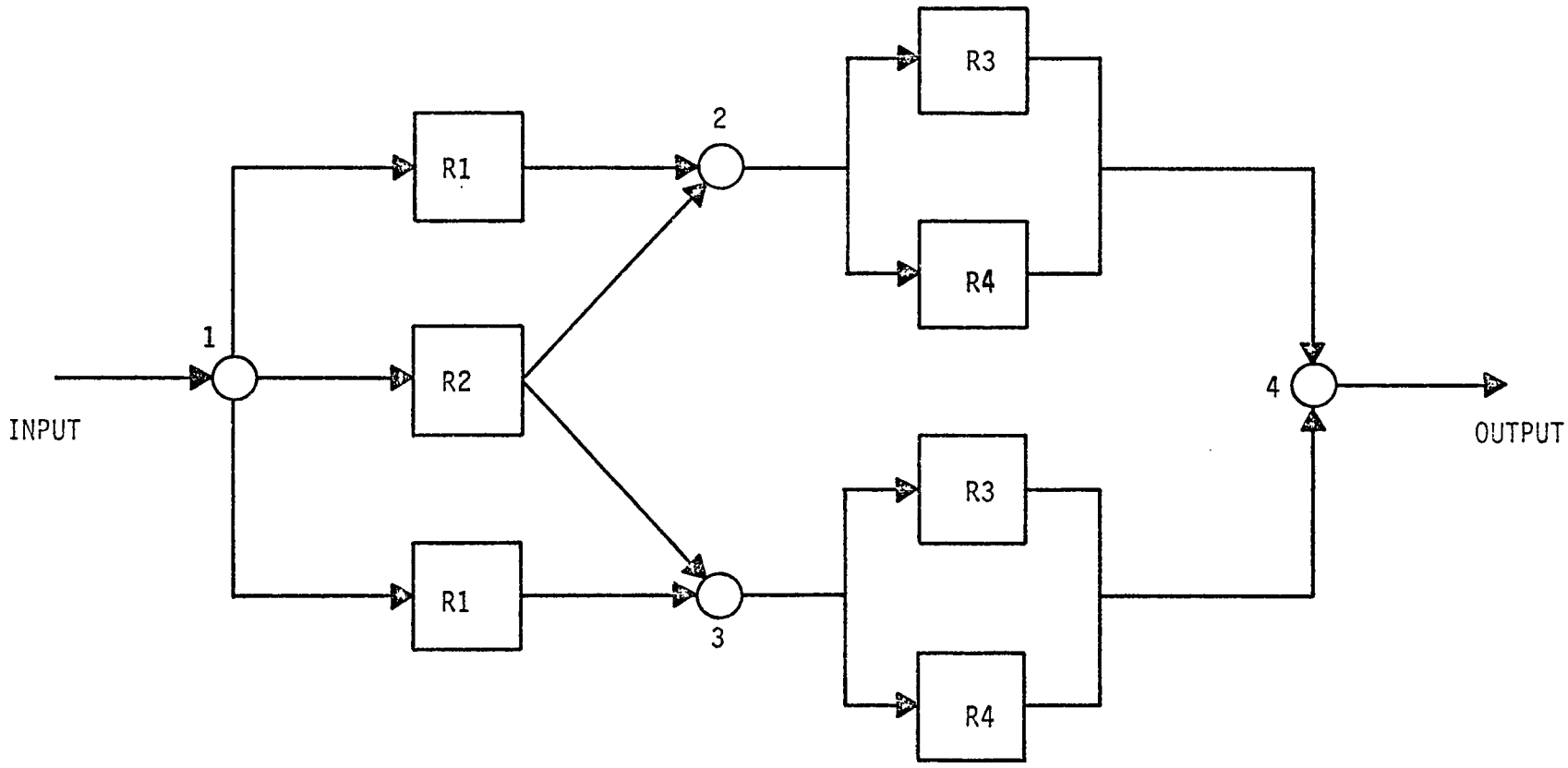


Figure 5.1

BIBLIOGRAPHY

1. Banerjee, S. K. and K. Rajamani, "Parametric Representation of Probability in Two Dimension - A New Approach in System Reliability Evaluation," IEEE Trans. Rel., R-21, 56-60, 1972.

2. Banerjee, S. K. and K. Rajamani, "Optimization of System Reliability Using a Parametric Approach," IEEE Trans. Rel., R-22, 35-39, 1973.

3. Barlow, R. E., F. Proschan and L. C. Hunter, Mathematical Theory of Reliability, John Wiley & Sons, Inc., 1965.

4. Batts, J. R., "Computer Program for Approximating System Reliability, Part II," IEEE Trans. Rel., R-20, 88-90, 1971.

5. Bazovsky, I., Reliability Theory and Practice, Prentice Hall, Inc., New Jersey, 1961.

6. Bellman, R. and S. Dreyfus, "Dynamic Programming and the Reliability of Multicomponent Devices," Operations Research, 6,200-206, 1958.

7. Benning, C. J., "Reliability Prediction Formulas for Standby Redundant Structures," IEEE Trans. Rel., (Lett.), R-16, 136-137, 1967.

8. Black, G. and F. Proschan, "On Optimal Redundancy," Operations Research, 7, 581-588, 1959.

9. Brown, D. B., "A Computerized Algorithm for Determining the Reliability of Redundant Configurations," IEEE Trans. Rel., R-20, 102-107, 1971.

10. Buffham, B. A., D. C. Freshwater and F. P. Lees, "Reliability Engineering," I. Chem. E. Symposium, London, No. 34, pp. 87, 1971.

11. Burton, R. M. and G. T. Howard, "Optimal System Reliability for a Mixed Series and Parallel Structure," J. of Math. Anal. Appl., 28, 370-382, 1969.

12. Burton, R. M. and G. T. Howard, "Optimal Design for System Reliability and Maintainability," IEEE Trans. Rel., R-20, 56-60, 1971.

13. Buzacott, J. A., "Network Approaches to Finding the Reliability of Repairable Systems," IEEE Trans. Rel., R-19, 140-146, 1970.

14. Buzacott, J. A., "Finding the MTBF of Repairable Systems by Reduction of the Reliability Block Diagram," Micro-electronics & Reliability, 6, 105-112, 1967.

15. Chan, W. C., "A Generalized Reliability Function for Systems of Parallel Components," IEEE Trans. Rel., R-17, 199-201, 1968.

16. Chung, W., "Generalized Reliability Function for Systems of Arbitrary Configurations," IEEE Trans. Rel., R-20, 85-87, 1971.

17. Coulter, K. E. and V. S. Morello, "Improving on Stream Time in Process Plants," Chemical Engineering Process, 68, 56-59, 1972.

18. Everett, H., III, "Generalized Language Multiplier Method for Solving Problems of Optimum Allocation of Resources," Operations Research, 11, 399-417, 1963.

19. Fan, L. T., C. S. Wang, F. A. Tillman and C. L. Hwang, "Optimization of Systems Reliability, IEEE Trans. Rel., R-16, 81-86, 1967.

20. Federowicz, A. J. and M. Mazumdar, "Use of Geometric Programming to Maximize Reliability Achieved by Redundancy," Operations Research, 16, 949-954, 1968.

21. Feller, W., An Introduction to Probability Theory and Its Applications, Volume I, John Wiley & Sons, New York, 1957.

22. Fleming, J. L., "Relcomp: A Computer Program for Calculating System Reliability and MTBF," IEEE Trans. Rel., R-20, 102-107, 1971.

23. Fyffe, D. E., W. W. Hines and N. K. Lee, "System Reliability Allocation and A Computational Algorithm," IEEE Trans. Rel., R-17, 64-69, 1968.

24. Ghare, P. M. and R. E. Taylor, "Optimal Redundancy for Reliability in Series Systems," Operations Research, 17, 838-847, 1969.

25. Gnedenko, B. V., Yu. K. Belyayev, and A. D. Solovyev, Mathematical Methods of Reliability Theory, Academic Press, New York, 1969.

26. Green, E. A. and A. J. Bourne, Reliability Technology, John Wiley & Sons, New York, 1972.

27. Henley, E. J. and R. A. Williams, Graph Theory in Modern Engineering, Academic Press, New York, 1973.

28. Jensen, P. A. and M. Bellmore, "An Algorithm to Determine the Reliability of a Complex System," IEEE Trans. Rel., R-18, 1969-174, 1969.

29. Kim, Y. H., K. E. Case, and P. M. Ghare, "A Method for Computing Complex System Reliability," IEEE Trans. Rel., R-21, 215-219, 1972.

30. Kettelle, J. D., Jr., "Least-cost Allocation of Reliability Investment," Operations Research, 10, 249-265, 1962.

31. Lambert, K. B., A. G. Walvekar, and J. P. Hirmas, "Optimal Redundancy and Availability Allocation in Multistage Systems," IEEE Trans. Rel., R-20, 182-185, 1971.

32. Lawler, E. L. and M. D. Bell, "A Method for Solving Discrete Optimization Problems," Operations Research, 14, 1098-1112, 1966.

33. Liittschwager, J. M., "Dynamic Programming in the Solution of a Multi-stage Reliability Problem," J. Ind. Eng., 15, 168-175, 1964.

34. Locks, M. O., "The Maximum Error in System Reliability Calculation by Using a Subset of the Minimal States," IEEE Trans. Rel., R-20, 231-234, 1971.

35. McFatter, W. E., "Reliability Experiences in a Large Refinery," Chemical Engineering Process, 68, 52-55, 1972.

36. McNichols, R. J. and Messer G. H., Jr., "A Cost-based Availability Allocation: A Tutorial Review," IEEE Trans. Rel., R-20, 1978-182, 1971.

37. Messinger, M. and M. L. Shooman, "Reliability Approximations for Complex Structures," Proc. IEEE Ann. Symposium on Reliability, 292-304, 1967.

38. Messinger, M. and M. L. Shooman, "Techniques for Optimum Spares Allocation: A Tutorial Review," IEEE Trans. Rel., R-19, 156-166, 1970.

39. Misra, K. B., "An Algorithm for the Reliability Evaluation of Redundant Networks," IEEE Trans. Rel., R-19, 146-151, 1970.

40. Misra, K. B., "Reliability Optimization of a Series Parallel System," IEEE Trans. on Rel., R-21, 230-238, 1972.

41. Misra, K. B., "A Method of Solving Redundancy Optimization Problems," IEEE Trans. on Rel., R-20, 117-120, 1971.

42. Misra, K. B., "A Simple Approach for Constrained Redundnacy Optimization Problem," IEEE Trans. Rel., R-21, 30-34, 1972.

43. Misra, K. B., "Dynamic Programming Formulation of the Redundancy Allocation Problem," Int. J. Math. Educ. Sci. Technology, 2, 207-215, 1971.

44. Mizukami, K., "Optimum Redundancy for Maximum System Reliability by the Method of Convex and Integer Programming," Operations Research, 16, 392-406, 1968.

45. Moscowitz, F., "The Analysis of Redundancy Networks," AIEE Trans. (Commun. Electron.), 77, 627-632, 1958.

46. Nelson, A. C., Jr., J. R. Batts, and R. L. Beadles, "A Computer Program for Approximating System Reliability," IEEE Trans. Rel., R-19, 61-65, 1970.

47. Polovko, A. M., Fundamentals of Reliability Theory, Academic Press, New York, 1968.

48. Proschan, F. and T. A. Bray, "Optimal Redundancy Under Multiple Constraints," Operations Research, 13, 800-814, 1965.

49. Rau, J. G., Optimization and Probability in Systems Engineering, Van Nostrand Reinhold Company, New York, 1970.

50. Reiter, S. and D. B. Rice, "Discrete Optimizing Solution Procedures for Linear and Non-linear Programming Problems," Management Science, 12, 829-850, 1966.

51. Sharma, J. and K. V. Venkateswaran, "A Direct Method for Maximizing the System Reliability," IEEE Trans. Rel., R-20, 256-259, 1971.

52. Shooman, M. L., Probabilistic Reliability: An Engineering Approach, McGraw Hill, New York, 1968.

53. Terano, T., Y. Murayama, and K. Kurosu, "Optimum Design of Safety Systems," IFAC Kyoto Symposium, August, 1970, Preprint pp. 52-57.

54. Tillman, F. A. and J. M. Liittschwager, "Integer Programming Formulation of Constrained Reliability Problems," Management Science, 13, 887-899, 1967.

55. Tillman, F. A., C. L. Hwang, L. T. Fan and S. A. Balbale, "Systems Reliability Subject to Multiple Non-linear Constraints," IEEE Trans. Rel., R-17, 153-157, 1968.

56. Tillman, F. A., "Optimization by Interger Programming of Constrained Reliability Problems with Several Modes of Failure," IEEE Trans. Rel., R-18, 47-53, 1969.

57. Tillman, F. A., C. L. Hwang, L. T. Fan and K. C. Lai, "Optimal Reliability of a Complex System," IEEE Trans. Rel., R-19, 95-100, 1970.

58. Ufford, P. S., "Equipment Reliability Analysis for Large Plants," Chemical Engineering Process, 68, 47-49, 1972.

59. Von Alven, W. H., Editor, Reliability Engineering, ARINC Res. Corp., Prentice Hall, 1964.

60. Williams, R. A., Systematic Flow Graph Analysis and Applications, Ph.D. Thesis, University of Houston, 1971.

61. Woodhouse, C. F., "Optimal Redundancy Allocation by Dynamic Programming," IEEE Trans. Rel., R-21, 60-62, 1962.

NOMENCLATURE

A	- system availability
C	- upper limit on system cost
C^*	- upper limit on system cost
C_{AO}	- upper limit on cost constraint - system availability maximization
C_{Fi}	- capital cost of a unit in the ith module
C_i	- cost per unit of the ith module
C_I	- net income per hour
C_j	- jth minimal cut
$\overline{C}_j$	- complement of the event C_j
C_M	- maintenance cost per hour of system downtime
C_{PO}	- upper limit on cost constraint - system profit maximization
C_{TF}	- total fixed capital investment
C_{TI}	- total net income during system lifetime of T^+ hours
C_{TM}	- total maintenance cost during system lifetime of T^+ hours
f	- failure density function
F	- system unreliability
g_o	- objective function to be minimized
g_{i1}	- 1st term in the ith constraint, see Equation (4-20)
g_{i2}	- 2nd term in the ith constraint, see Equation (4-20)
g_{ij}	- contribution of ith module to ith constraint, see Equation (3-4)
G_j	- upper limit on jth constraint, see Equation (3-4)
K	- positive constant, see Equation (4-4)
k_c	- constant, corresponds to the cost of a single unit of capacity L, see Equation (4-6)
k_i	- minimum number of units in the ith module that might function for the module to operate successfully

k_v - constant, corresponds to the volume of a single unit of capacity L, see Equation (4-12)

k_w - constant, corresponds to the weight of a single unit of capacity L, see Equation (4-11)

L - system capacity

$\overline{M}$ - system mean time between failure

MDT - system mean downtime

MTBF - system mean time between failure

$(MTBF)_i$ - mean time between failure of a unit of the ith module

MTBM - system mean time between maintenance

N - total number of modules in the system

n_i - number of units in the ith module

N_i - number of units in the ith module

P - system profit

P_i - ith minimal path

p_i - reliability of a unit belonging to the ith stage, see Equation (4-13)

R - system reliability

R_A - reliability of active redundancy module

R_c - reliability of cold standby redundancy module

R_H - reliability of hot standby redundancy module

R_i - ith module reliability

r_i - reliability of a unit of the ith module

r_{ij} - reliability of jth unit of the ith module

R_L - lower bound approximation for R

R_U - upper bound approximation for R

s, s_v, s_w - non-negative constants less than one

S_{R_i} - sensitivity of system reliability to the ith module reliability

$S_{r_{ij}}$ - sensitivity of system reliability to the reliability of the jth unit of the ith module

T - system maintenance interval

t - variable of integration

T^+ - system lifetime

T_D - total time during which the system is down

T_E - mean time for emergency maintenance on the system

T_i - maintenance interval of a unit of the ith module

T_s - mean time for scheduled maintenance on the system

T_u - total time during which the system is operating

V^* - upper limit on system volume

W^* - upper limit on system weight

X - percent of fixed capital investment

$\underline{x}$ - vector with elements $x_j = 0$ or 1

$\underline{X}_I$, X_p, X^+ - denote vectors of variables, real and/or integer

x_j - jth binary variable

x_{kj}, x_{nj} - jth binary variables belonging to the non-negative integer variables k and n, respectively, see Equations (4-26) and (4-27)

x'_{nj} - given by $(1 - x_{nj})$, see Equation (4-28)

Z - total number of terms involved in Equation (2-10)

λ - exponential failure pdf parameter, also called the conditional failure rate

λ_i - of the ith module

λ'_i - of the ith module in standby

λ^o_i - of the ith module in service

z - hazard rate

APPENDIX A

MODULE RELIABILITY AND SENSITIVITY EXPRESSIONS

APPENDIX A

MODULE RELIABILITY AND SENSITIVITY EXPRESSIONS

Module Reliability

Active Redundancy Module - All n units are actively connected in parallel. If we let r_i be the probability that each component in module i functions for time t, then the probability that at least k_i out of n_i identical components function for time t is given by the binomial distribution [15]

$$R_{Ai}(n_i;k_i;r_i) = \sum_{l=k_i}^{n_i} \binom{n_i}{l} r_i^{l} (1-r_i)^{n_i-l} \tag{A-1}$$

The case $k_i=1$ leads to the familiar expression for the reliability of a parallel connection of n_i units

$$R_{Ai}(n_i,1;r_i) = 1 - (1 - r_i)^{n_i} \tag{A-2}$$

When any one component is sufficient for module operation but the various components are different, the reliability expression is [52]:

$$R_{Ai}(n_i,1;r_{ij}) = 1 - \prod_{j=1}^{n_i} (1 - r_{ij}) \tag{A-3}$$

where r_{ij} is reliability of the jth unit of module i.

When exponential failure probability density function (pdf) is assumed with parameter λ_i for identical components or λ_{ij} for different components, we have

$$\begin{aligned} r_i &= e^{-\lambda_i t} \\ r_{ij} &= e^{-\lambda_{ij} t} \end{aligned} \tag{A-4}$$

Standby Redundancy Module - We distinguish between three standby redundancy modules.

1. Hot Standby Redundnacy Module

The units in hot standby have the same failure pdf as those in service. The module reliability expressions are identical to those in active redundancy module, Equations (A-1) to (A-4), that is,

$$R_{Hi}(n_i,k_i;r_i) = R_{Ai}(n_i,k_i;r_i) \tag{A-5}$$

2. Cold Standby Redundancy Module

A component or unit whose failure probability in standby is zero is called a cold standby. If the exponential failure pdf is assumed, the probability that at least k_i out of n_i identical components will function for time t is [7]:

$$R_{Ci}(n_i,k_i;t) = e^{-k_i\lambda_i t} \sum_{j=0}^{(n_i-k_i)} \frac{(k_i\lambda_i t)^j}{j!} \tag{A-6}$$

A specific case of cold standby arises when $k_i=1$.

$$R_{Ci}(n_i,1;t) = e^{-\lambda_i t} \sum_{j=0}^{(n_i-1)} \frac{(\lambda_i t)^j}{j!} \tag{A-7}$$

Although R_{Ci} cannot be obtained in a closed form for a general failure pdf, a recursive computational procedure can be used in the case $k_i=1$.to obtain an approximation to R_{Ci}. Let $g_i(t)$ be the failure pdf for the components in module i, and let $P_i(n_i,t)$ be the probability that n_i components are sufficient to operate module i for time t. Suppose that the components are used in the order n_i, n_i-1, n_i-2, ... 2, 1, then we have

$$P_i(1,t) \equiv 1 - \int_0^t g_i(\tau)\, d\tau \tag{A-8}$$

$$P_i(m,t) \equiv P_i(1,t) + \int_0^t P_i(m-1,t-\tau)\, g_i(\tau)\, d\tau \tag{A-9}$$

$$2 \leq m \leq n_i$$

The recursive Equation (A-9) gives the probability that m components are sufficient for module i for time t. It is expressed as the probability that the mth component alone will function until time t plus the probability that the mth component will fail at τ and the remaining $m-1$ components will function from τ until t. A discrete approximation to Equation (A-9) can be used to compute recursively for each m, an approximation to $P_i(m,t)$, which provides a point on the R_{Ci} function.

3. Warm Standby Redundancy Module

If the exponential failure pdf is assumed with parameter λ_i^o in service and λ_i^1 in standby for units in ith module, Polovko [47] gives the following expression for the module reliability of a 1-out-of-n_i system

$$R_{Wi}(n_i,1;t) = e^{-\lambda_i^o t}\left[1 + \sum_{l=1}^{(n_i-1)} \frac{A_l}{l!}\left(1 - e^{-\lambda_i^1 t}\right)^l\right] \tag{A-10}$$

where

$$A_l = \prod_{j=0}^{l-1}\left(j + \frac{\lambda_i^o}{\lambda_i^1}\right) \tag{A-11}$$

For a general failure pdf $g_i(t)$ for the components in module i, Polovko [47] gives a recursive relationship for a 1-out-of-n_i system

$$R_{Wi}(n_i,1;t) = R_{Wi}(n_i-1,1;t) + \int_0^t p_i^1(\tau)\, p_i^o(t-\tau)\, g_i^{(n_i-1)}(\tau)\, d\tau \qquad \text{(A-12)}$$

where $p_i^1(\tau)$ is the probability of failure free operation of the standby component up to the instant of its connection at time τ; $p_i^o(t-\tau)$ is the probability of failure free operation of the standby component from the instant of its connection at time τ up to time t; $g_i^{(n_i-1)}(\tau)$ is the failure pdf for a system with (n_i-1) components.

Module Sensitivity

This is evaluated for a system in which individual units in a module have exponential failure pdf with parameter λ_i.

Active Redundancy Module -

$$\frac{\partial R_{Ai}(n_i,k_i;r_i)}{\partial n_i} = R_{Ai}(n_i,k_i;r_i) - R_{Ai}(n_i-1,k_i;r_i) \qquad \text{(A-13)}$$

Using the formula

$$\binom{n_i}{l} = \binom{n_i-1}{l-1} + \binom{n_i-1}{l} \qquad l \geq 1$$

we have from Equation (A-1)

$$R_{Ai}(n_i,k_i;r_i) = \sum_{l=k_i}^{n_i} \left[\binom{n_i-1}{l-1} + \binom{n_i-1}{l}\right] r_i^{\,l}\,(1-r_i)^{n_i-l}$$

Expanding the summation, we obtain

$$R_{Ai}(n_i,k_i;r_i) = R_{Ai}(n_i-1,k_i;r_i) + \binom{n_i-1}{k_i-1} r_i^{k_i} (1-r_i)^{n_i-k_i} \tag{A-14}$$

Substituting in Equation (A-13)

$$\frac{\partial R_{Ai}(n_i,k_i;r_i)}{\partial n_i} = \binom{n_i-1}{k_i-1} r_i^{k_i} (1-r_i)^{n_i-k_i} \tag{A-15}$$

when $k_i=1$, we have from Equation (A-15)

$$\frac{\partial R_{Ai}(n_i,1;r_i)}{\partial n_i} = r_i(1-r_i)^{n_i-1} \tag{A-16}$$

The above result also follows from Equation (A-2).

From Equation (A-3) we have

$$\frac{\partial R_{Ai}(n_i,1;r_{ij})}{\partial r_{ij}} = \prod_{\substack{j=1 \\ j\neq n_i}}^{n_i} (1-r_{ij}) \tag{A-17}$$

When $r_i = e^{-\lambda_i t}$, we have from Equation (A-1)

$$\frac{\partial R_{Ai}(n_i,k_i;t)}{\partial t} = \lambda_i \sum_{l=k_i}^{n_i} \binom{n_i}{l} \Big\{ (n_i-l)(e^{-\lambda_i t})^{l+1} (1-e^{-\lambda_i t})^{n_i-l-1} - l(e^{-\lambda_i t})^{l} (1-e^{-\lambda_i t})^{n_i-l} \tag{A-18}$$

Expanding the summations, we obtain

$$\sum_{l=k_i}^{n_i} \begin{pmatrix} n_i \\ l \end{pmatrix} (n_i-l)\,(e^{-\lambda_i t})^{l+1}\,(1-e^{-\lambda_i t})^{n_i-l-1}$$

$$= n_i(e^{-\lambda_i t})\, R_{Ai}(n_i-1,k_i;t) \tag{A-19}$$

And

$$\sum_{l=k_i}^{n_i} \begin{pmatrix} n_i \\ l \end{pmatrix} l(e^{-\lambda_i t})^{l}\,(1-e^{-\lambda_i t})^{n_i-l}$$

$$= n_i\,(e^{-\lambda_i t})\, R_{Ai}(n_i-1,k_i,t) + k_i \begin{pmatrix} n_i \\ k_i \end{pmatrix} (e^{-\lambda_i t})^{k_i}\,(i-e^{-\lambda_i t})^{n_i-k_i} \tag{A-20}$$

Substituting Equations (A-19) and (A-20) in Equation (A-18)

$$\frac{\partial R_{Ai}(n_i,k_i;t)}{\partial t} = -k_i\lambda_i \begin{pmatrix} n_i \\ k_i \end{pmatrix} \left(e^{-\lambda_i t}\right)^{k_i} \left(1-e^{-\lambda_i t}\right)^{n_i-k_i} \tag{A-21}$$

Comparing Equations (A-15) and (A-21) we have

$$\frac{\partial R_{Ai}(n_i,k_i;t)}{\partial t} = -n_i\lambda_i \frac{\partial R_{Ai}(n_i,k_i;r_i)}{\partial n_i} \tag{A-22}$$

Standby Redundancy Module - Two types of standby redundancy modules are considered

1. Hot Standby Redundancy Module

The sensitivity expressions are identical to those developed for active redundancy module.

2. Cold Standby Redundancy

$$\frac{\partial R_{Ci}(n_i,k_i;t)}{\partial n_i} = R_{Ci}(n_i,k_i;t) - R_{Ci}(n_i-1,k_i;t) \tag{A-23}$$

From Equation (A-6)

$$\frac{\partial R_{Ci}(n_i,k_i;t)}{\partial n_i} = e^{-k_i\lambda_i t} \frac{(k_i\lambda_i t)^{n_i-k_i}}{(n_i-k_i)!} \tag{A-24}$$

When $k_i=1$, we have from above

$$\frac{\partial R_{Ci}(n_i,1;t)}{\partial n_i} = e^{-\lambda_i t} \frac{(\lambda_i t)^{n_i-1}}{(n_i-1)!} \tag{A-25}$$

Using Equation (A-6)

$$\frac{\partial R_{Ci}(n_i,k_i;t)}{\partial t} = -k_i\lambda_i \; e^{-k_i\lambda_i t} \frac{(k_i\lambda_i t)^{n_i-k_i}}{(n_i-k_i)!} \tag{A-26}$$

Comparing Equations (A-24) and (A-26)

$$\frac{\partial R_{Ci}(n_i,k_i;t)}{\partial t} = -k_i\lambda_i \frac{\partial R_{Ci}(n_i,k_i;t)}{\partial n_i} \tag{A-27}$$

APPENDIX B

PATH FINDING ALGORITHM

APPENDIX B

PATH FINDING ALGORITHM

The algorithm is a modified version of the path finding algorithm developed by Henley and Williams [27]. The main features of the algorithm are presented here. For a more detailed analysis please refer to the original work of Henley and Williams [27].

The reliability flow graph consists of a set of independent modules. These are inter-connected by an array of points called nodes and directed lines between nodes called edges or branches. Each module has an input node and one or more output nodes. The nodes are numbered such that the arrows point from a node with a lower address to a node with a higher address. Thus, the input node of the system has the lowest address and the output node of the system has the highest address.

The algorithm to generate all the paths between the input and the output nodes of the system is best explained with the help of an example. Consider the reliability flow graph shown in Figure B-1. All the edges together with the corresponding modules are first ordered in a list such that the edges with the lowest starting address appear highest in the list. This is shown in Table B.1. The following steps then generate all the paths connecting the input node and the output node of the system.

1. List all the edges which originate at the input node of the system. These are the edges which appear at the top of the list shown in Table B.1. These will be referred to as the input edges. The modules corresponding to the edges are listed in another column.

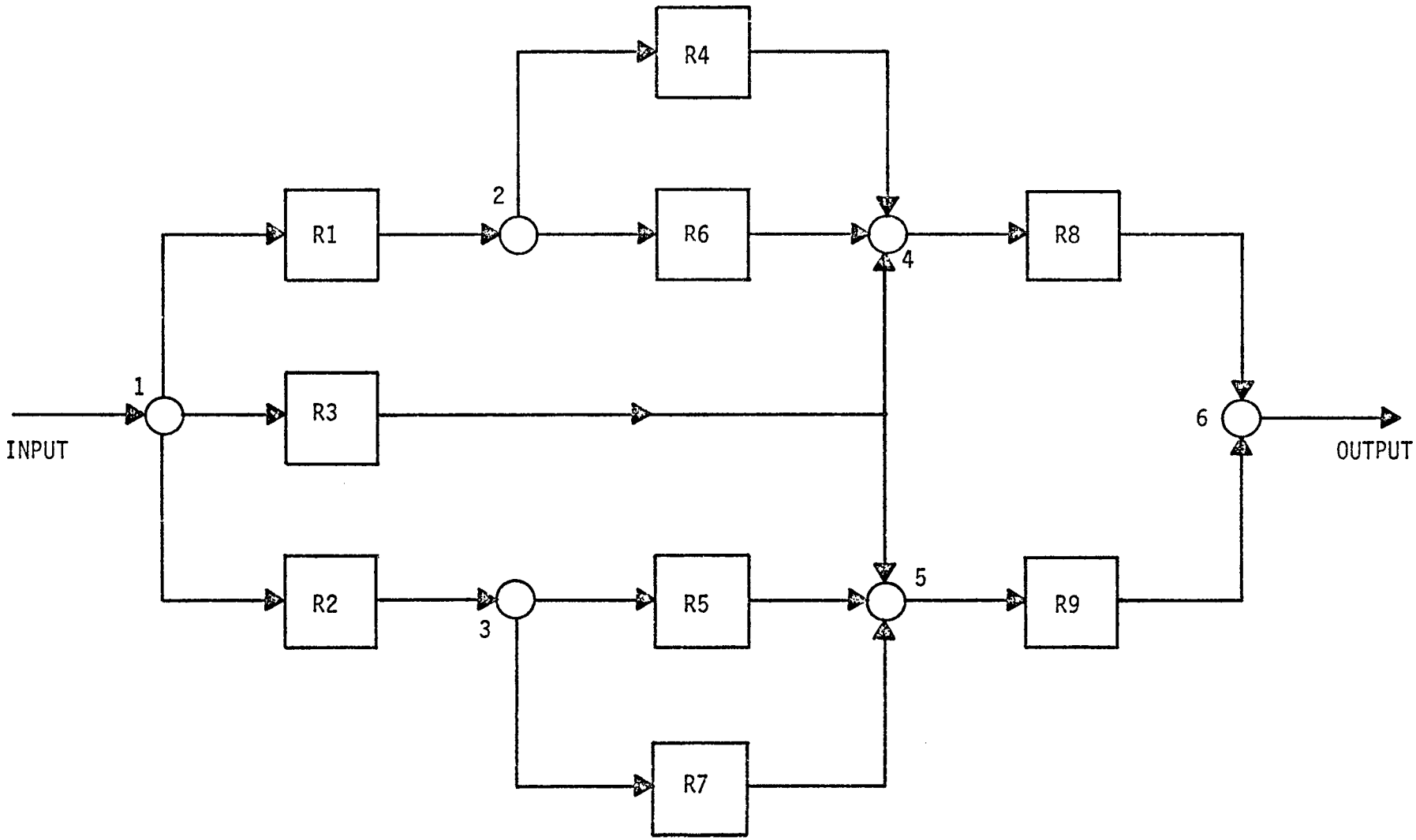


Figure B.1
Non-series-parallel system

EDGE	MODULE
1-2	R_1
1-4	R_3
1-5	R_3
1-3	R_2
2-4	R_4
2-4	R_6
3-5	R_5
3-5	R_7
4-6	R_8
5-6	R_9

Table B.1

System Configuration

2. Pick the first edge from the Table B.1 which follows the input edges. This will be referred to as the output edge. Append the output edge to those input edges whose ending node match its starting node. The corresponding modules are also appended and listed in another column. The new combinations formed are listed in the order they are generated. Next pick the edge following the output edge considered earlier. This will be the new output edge. Again append this output edge to those input edges and combinations formed earlier whose ending nodes match its starting node. The new combinations are listed in the order they are formed. This procedure continues until the last edge listed in Table B.1 has been used to generate new combinations. These operations are shown in Table B.2. The combinations whose last node is the same as the output node of the system, are the system paths. As Table B.2 illustrates, some of these paths have identical node configurations but are distinguishable by their different module configurations.

In the above discussion it is assumed that edges are numbered such that they point from a node with a lower address to a node with a higher address. These are called forward edges. The edges with nodes pointing from a higher address to a lower address are referred to as reverse edges. For a discussion of the path finding algorithm in the presence of reverse edges, please refer to the work of Williams [60].

	Node Configuration	Module Configuration
Edges starting at the input node "1"	1 - 2	R_1
	1 - 4	R_3
	1 - 5	R_3
	1 - 3	R_2
	1 - 2 - 4	$R_1 - R_4$
	1 - 2 - 4	$R_1 - R_6$
	1 - 3 - 5	$R_2 - R_5$
	1 - 3 - 5	$R_2 - R_7$
Combinations ending at the output node "6" giving the system paths	1 - 4 - 6	$R_3 - R_8$
	1 - 2 - 4 - 6	$R_1 - R_4 - R_8$
	1 - 2 - 4 - 6	$R_1 - R_6 - R_8$
	1 - 5 - 6	$R_3 - R_9$
	1 - 3 - 5 - 6	$R_2 - R_5 - R_9$
	1 - 3 - 5 - 6	$R_2 - R_7 - R_9$

Table B.2

Combination of Edges

APPENDIX C

PATH UNIONS

APPENDIX C

PATH UNIONS

The procedures to generate all the necessary path unions are best illustrated with the help of an example. Consider the reliability network shown in Figure C.1. The path finding algorithm traces all the paths connecting the input and the output nodes of the system. These paths in their node and module configurations are listed in Table C.1.

The following steps then generate the path unions:

1. Transform each path into the binary form; i.e. the locations containing 1's (0's) indicate the modules present (absent) in the path.

2. Generate the path unions by using a bitwise logical 'OR' operation in order of ascending path number. The 2-path unions are generated from the single path unions (or paths) P-1 OR P-2, P-1 OR P-3,..., P-1 OR P-M, P-2 OR P-3,..., P-2 OR P-M.,,,. P-(M-1) OR P-M, where M represents the total number of paths in the system. For the above example, 2-path unions are from (5) to (10) in Table C.2. Then proceed to form all the 3-path unions from the 2-path unions obtained earlier. These are P-1 OR P-2 OR P-3,..., P-1 OR P-2 OR P-M,..., P-(M-2) OR P-(M-1) OR P-M. 3-path unions are from (11) to (14) in Table C.2. Similar procedures continue until we form the M-path union, P-1 OR P-2 OR ... OR P-(M-1) OR P-M. For the example, number (15) shows the 4-path union. There is a sign bit attached to each path and path union and it changes sign with the generation of each higher order path union. Thus, it is positive for paths, negative for 2-path unions, positive for 3-path unions, etc., and $(-1)^{M-1}$ for the M-path union.

3. Any path union generated as outlined in Step 2 that has 1's in every location in its binary representation and whose end path in the path

union is not the Mth path is rejected. This is so because it generates identical path unions with equal number of positive and negative sign bits. In the example, the 3-path union numbered (11) i.e. P-1 OR P-2 OR P-3, generates the 4-path union numbered (15), i.e. P-1 OR P-2 OR P-3 OR P-4. The two are identical and have opposite signs and hence cancel each other. These are excluded from the list of path and path unions. The reduction in number of terms may be larger for some other systems. The system shown in Figure 2.9 has six paths. The path enumeration technique predicts $2^6-1 = 63$ terms in the reliability expression. Introducing the above modification reduces the number of terms to 47. The savings in computer time is significant.

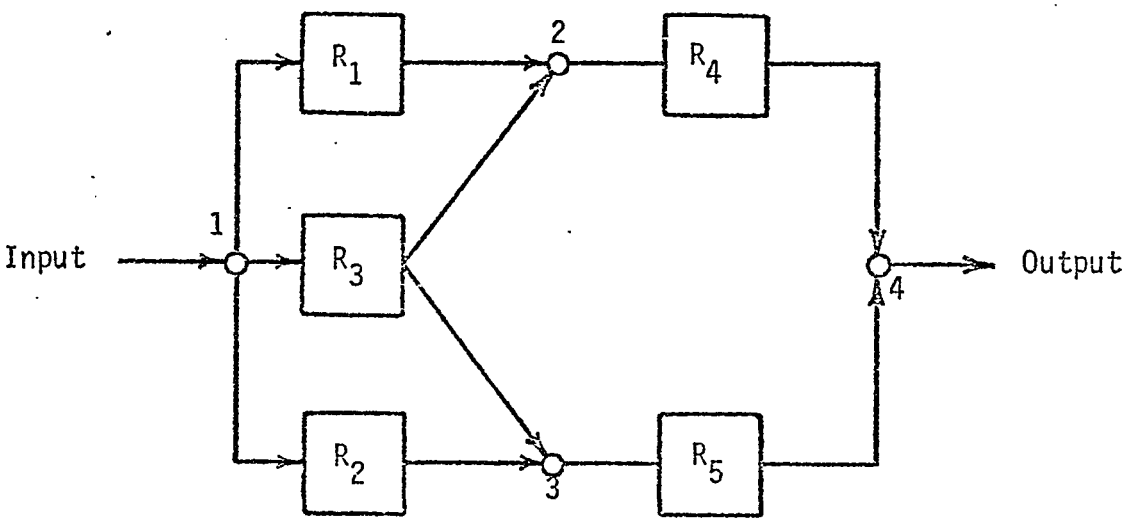


Figure C.1

Reliability Graph

Path	Node Configuration	Module Configuration
P-1	1 - 2 - 4	R1 - R4
P-2	1 - 2 - 4	R3 - R4
P-3	1 - 3 - 4	R2 - R5
P-4	1 - 3 - 4	R3 - R5

Table C.1

Path Detection

No.	Path or Path Union	Sign	Module				
			R1	R2	R3	R4	R5
1	P-1	+	1	0	0	1	0
2	P-2	+	0	0	1	1	0
3	P-3	+	0	1	0	0	1
4	P-4	+	0	0	1	0	1
5	P-1 OR P-2	-	1	0	1	1	0
6	P-1 OR P-3	-	1	1	0	1	1
7	P-1 OR P-4	-	1	0	1	1	1
8	P-2 OR P-3	-	0	1	1	1	1
9	P-2 OR P-4	-	0	0	1	1	1
10	P-3 OR P-4	-	0	1	1	0	1
*11	P-1 OR P-2 OR P-3	+	1	1	1	1	1
12	P-1 OR P-2 OR P-4	+	1	0	1	1	1
13	P-1 OR P-3 OR P-4	+	1	1	1	1	1
14	P-2 OR P-3 OR P-4	+	0	1	1	1	1
*15	P-1 OR P-2 OR P-3 OR P-4	-	1	1	1	1	1

*Path unions (11) and (15) will not appear in the final list.

Table C.2

Path and Path Unions

APPENDIX D

MODIFIED INTEGER GRADIENT METHOD

APPENDIX D

MODIFIED INTEGER GRADIENT METHOD

The general nonlinear integer programming problem can be stated as follows:

Maximize the nonlinear objective function

$$f(\underline{X}) \tag{D-1}$$

subject to several functional nonlinear (or linear) constraints

$$g_i(\underline{X}) \leq 0 \qquad i = 1,2 \ldots m \tag{D-2}$$

non-negative constraints

$$\underline{X} \geq \underline{0} \tag{D-3}$$

and finally the integer constraints

$$\underline{X} : \text{integral} \tag{D-4}$$

where $\underline{X}$ is an n vector.

Optimal Seeking Algorithm

The procedure described below is mainly due to Reiter and Rice [50]. The search algorithm has three phases.

Phase (1): Selection of an Initial Point

The simplest method of selecting initial point is a statistical method using uniform random numbers. Assume that independent upper limits on $\underline{X}$ also exist.

$$0 \leq x_j \leq u_j \qquad j = 1,2, \ldots n \tag{D-5}$$

Equation (D-5) together with Equation (D-4) define a finite set of points enclosed by a hyperrectangle in n-space. So we sample with a uniform distribution from the set of integers in the interval $[0, u_j]$ $j = 1,2, \ldots n$. This results in a uniform sampling of lattice points in the hyperrectangle.

If the selected initial point $\underline{X}_I$ is feasible, we go directly to Phase (3), otherwise, we go to Phase (2).

<u>Phase (2)</u>: Search for a Feasible Solution $\underline{X}^+$

A search by the method of "weighted perpendiculars" (WP) may be the most general one. The quantity S_i

$$S_i = g_i(\underline{X}_I) \qquad i = 1,2, \ldots m \tag{D-6}$$

is the amount by which the ith constraint is violated at $\underline{X}_I$. We define

$$\underline{Z} = \frac{-\left\{\sum_{i:S_i>o} S_i[\nabla_i(g_i(\underline{X})]_{\underline{X}=\underline{X}_I}\right\}}{\sum_{i:S_i>o} S_i} \tag{D-7}$$

where ∇_i denotes the vector of partial derivatives of the ith constraint with respect to x_i's. The vector $-\underline{Z}$ is thus a positive linear combination of these perpendiculars, and, therefore, lies in the convex cone spanned by them, as shown in Figure D-1.

$$\text{Let } Z_j^{\#} = \min._{Z_j \neq 0} |Z_j| \tag{D-8}$$

$$\text{Define } Z^+ = (Z_1^+, Z_2^+, \ldots Z_n^+)^T$$

$$Z_j^+ = \begin{cases} [(Z_j/Z_j^{\#}) + 0.5], \text{ if } Z_j \geq o \\ [(Z_j/Z_j^{\#} - 0.5], \text{ if } Z_j \leq \delta \end{cases} \qquad j = 1,2, \ldots n \tag{D-9}$$

where [w] denotes the largest integer not exceeding w.

The equation

$$\underline{Y}_b = \underline{X}_I + b\,\underline{Z}^+ \tag{D-10}$$

is the equation of a line passing through the point X_I and has possible direction towards feasible region, as shown in Figure D-1. Taking successively the values b = 1,2, ... we test the corresponding $\underline{Y}_b$, necessarily a lattice point, for feasibility; we eventually either locate a feasible point or violate one or more of the lower or upper limits. If one or more of these limits is violated, we hold the corresponding coordinate of $\underline{Y}_b$ constant at those limits and continue to search by increasing b. Ultimately, we find a feasible point $\underline{X}^+$ or all coordinates go outside of their limit. In the second case we abandon X_I and sample another starting point from the hyperrectangle as explained in Phase (1). In the case of linear constraints, another method called "permuted coordinates" (PC) may be better [50].

<u>Phase (3)</u>: Search for the Locally Optimum Point $\underline{X}_p$

Having located a feasible lattice point $\underline{X}^+$ we start Phase (3), where we end up with a locally maximal point.

Let $\underline{h}(\underline{X})$ denote the gradient of the objective function

$$\text{i.e. } \underline{h}(\underline{X}) = \nabla f(\underline{X}) \tag{D-11}$$

Define

$$\underline{Y}_d = X^+ + d \cdot \underline{h}(\underline{X}^+) \tag{D-12}$$

Since, $\underline{h}(\underline{X}^+)$ is not in general integer valued, we apply the procedure used in the WP method to obtain a normalized integer gradient vector $\underline{h}^+$, i.e.

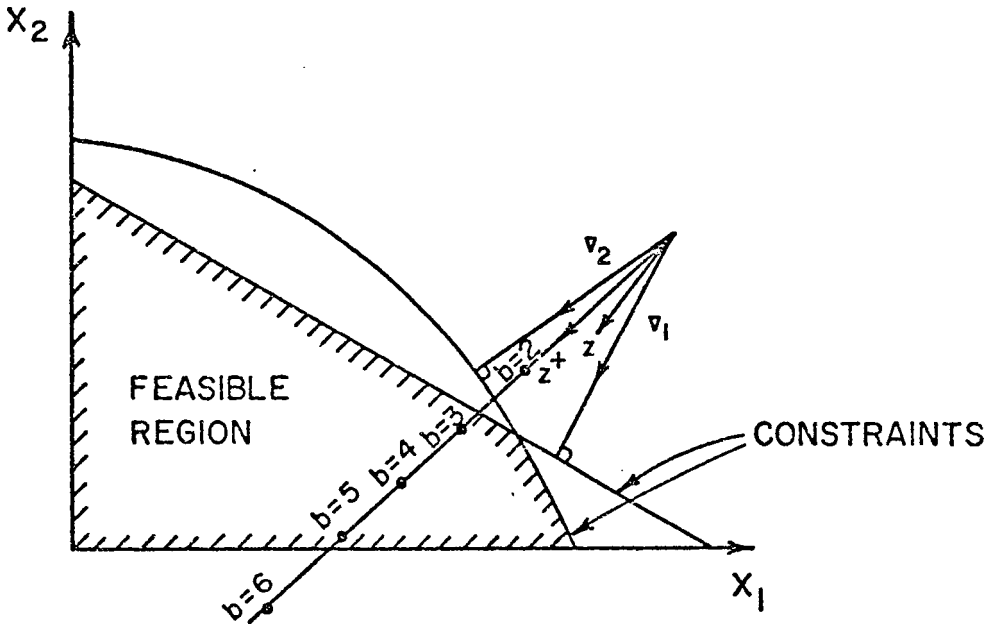


Figure D-1 Method of Weighted Perpendiculars (WP)

$$h_j^{\#} = \min._{h_j \neq o} |h_j| \tag{D-13}$$

$$h_j^{+} = \begin{cases} [(h_j/h_j^{\#}) + 0.5], \text{ if } h_j \geq o \\ [(h_j/h_j^{\#}) - 0.5], \text{ if } h_j \leq o \end{cases} \qquad j = 1,2, \ldots n \tag{D-14}$$

The vector $\underline{h}^+$ is the modified gradient of the objective function $f(\underline{X})$ and

$$\underline{Y}_d = \underline{X}^+ + d \cdot \underline{h}^+ \tag{D-15}$$

defines the path on which we search for improved values of the objective function. We find that value of d for which $f(\underline{Y}_d)$ is a maximum over feasible $\underline{Y}_d$ on this path. This may be done by letting $d = 0,1,2,\ldots$ successively and finding the first value of d for which either $\underline{Y}_{d+1}$ is infeasible or $f(\underline{Y}_{d+1}) \leq f(Y_d)$. If the best value of d denoted by d^*, is not zero, we set $X^+ = \underline{Y}_{d*}$ and repeat the procedure. If $d^* = 0$, we set the element of $h(X^+)$ of the smallest non-zero absolute value equal to zero to obtain $\underline{h}^1(\underline{X}^+)$ and proceed as before with $\underline{h}^1$ in place of $\underline{h}$. If $\underline{h}^1$ leads to no improvement, set the element of $\underline{h}^1$ of smallest non-zero absolute value equal to zero to obtain $\underline{h}^2(\underline{X}^+)$ and proceed with $\underline{h}^2$ in place of $\underline{h}$. This procedure is continued until $\underline{h}^v$ is obtained with only one non-zero element. If using $\underline{h}^v(\underline{X}^+)$ results in no improvement, we try successively h^{v+1} to h^{v+k} where $k+1$ is the number of non-zero elements of $\underline{h}$ and h^{v+j} is a vector with 1 in the coordinate corresponding to the partial derivative in $\underline{h}$ which ranks $(j+1)^{th}$ in absolute value, for $j \leq k$. If h^{v+k} results in no improvement then $\underline{X}^+$ is a locally maximal point. If any $\underline{h}^j$ leads to improvement, we set $\underline{X}^+$ equal to the $\underline{Y}_{d*}$ thus obtained and repeat the procedure.

Extension to Mixed Integer Problem

The solution procedure described above can be modified to apply to mixed integer problems.

To handle a mixed problem with $n'<n$ variables required to be integer valued, the variables are renumbered so that the first n' are the integer valued ones. Equations (D-8) and (D-9) for Z^+ and R^+ are replaced by

$$Z_j^{\#} = \min. \{|Z_j| : Z_j \neq 0,\ 1 \leq j \leq n'\} \tag{D-16}$$

$$Z_j^+ = \begin{cases} [(Z_j/Z_j^{\#}) + 0.5], & \text{if } Z_j \geq o \text{ and } 1 \leq j \leq n' \\ [(Z_j/Z_j^{\#}) - 0.5], & \text{if } Z_j \leq o \text{ and } 1 \leq j \leq n' \\ (Z_j/Z_j^{\#}) \quad , & \text{if } n' \leq j \leq n \end{cases} \tag{D-17}$$

As long as the gradient $\underline{h}^j(\underline{X}^+)$ has some non-zero values among its first n' coordinates, the procedures described earlier, which restrict attention to integer values of the path parameters, apply directly. If all of the first n' coordinates of $h^j(\underline{X}^+)$ are zero, it is unnecessary to normalize and adjust to integers. A grid is superimposed on the modified gradient path and procedures described in Phase (3) are continued.

In the case of functions and constraints which are not differentiable, paths of positive ascent calculated by first differences can be used in place of the gradient.

It should be noted that the method is not well suited to dealing with equality constraints.

APPENDIX E

SENSITIVITY EXPRESSIONS FOR SYSTEM AVAILABILITY AND SYSTEM PROFIT

APPENDIX E

SENSITIVITY EXPRESSIONS FOR SYSTEM AVAILABILITY AND SYSTEM PROFIT

$$A(\underline{n},T) = \frac{\int_0^T R(\underline{n},t)dt}{\int_0^T R(\underline{n},t)dt + T_E - (T_E - T_S)\ R(\underline{n},T)} \tag{E-1}$$

From Equation (E-1)

$$\frac{\partial A(\underline{n},T)}{\partial n_i} = \frac{\left(\frac{\partial}{\partial n_i}\int_0^T R(\underline{n},t)dt\right)\left(T_E - (T_E - T_S)\ R(\underline{n},T)\right) + \int_0^T R(\underline{n},t)dt\left((T_E - T_S)\ \frac{\partial R(\underline{n},t)}{\partial n_i}\right)}{\left[\int_0^T R(\underline{n},t)dt + T_E - (T_E - T_S)\ R(\underline{n},T)\right]^2} \tag{E-2}$$

$$i = 1,2, \ldots N$$

$$\frac{\partial A(\underline{n},T)}{\partial T} = \frac{T_E \cdot R(\underline{n},T) - (T_E - T_S)\ R^2(n,T) + (T_E - T_S)\left(\int_0^T R(\underline{n},t)dt\right)\frac{\partial R(\underline{n},T)}{\partial T}}{\left[\int_0^T R(\underline{n},t)dt + T_E - (T_E - T_S)\ R(\underline{n},T)\right]^2} \tag{E-3}$$

$$P(\underline{n},T) = C_I \cdot T^+ \cdot A(\underline{n},T) - \left(\sum_{i=1}^{N} n_i \cdot C_{Fi}\right)\left(1 + (1-A(\underline{n},T))\ \frac{X}{100} \cdot T^+\right) \tag{E-4}$$

$$\frac{\partial P(\underline{n},T)}{\partial n_i} = \left(C_I + \left(\sum_{i=1}^{N} n_i \cdot C_{Fi}\right) \frac{X}{100}\right) T^+ \cdot \frac{\partial A(\underline{n},T)}{\partial n_i}$$

$$- C_{Fi} \left(1 + (1-A(\underline{n},T) \frac{X}{100} \cdot T^+\right) \tag{E-5}$$

$$i = 1,2, \ldots N$$

$$\frac{\partial P(\underline{n},T)}{\partial T} = \left(C_I + \left(\sum_{i=1}^{N} n_i \cdot C_{Fi}\right) \frac{X}{100}\right) T^+ \cdot \frac{\partial A(\underline{n},T)}{\partial T} \tag{E-6}$$

Solution of Equations (E-1) to (E-6) involves the evaluation of the following terms,

1. $R(\underline{n},T)$ - The system reliability $R(\underline{n},T)$ is calculated according to the general computer algorithm developed in Chapter II.

2. $$\frac{\partial R(\underline{n},T)}{\partial T} = \sum_{i=1}^{N} \frac{\partial R(\underline{n},T)}{\partial R_i(n_i,T)} \cdot \frac{\partial R_i(n_i,T)}{\partial T}$$

$$= \sum_{i=1}^{N} \left(R\Big|_{R_i=1} - R\Big|_{R_i=0}\right) \frac{\partial R_i(n_i,T)}{\partial T} \tag{E-7}$$

where $\frac{\partial R_i(n_i,T)}{\partial T}$ is given by either Equations (A-21) or (A-26).

3. $$\frac{\partial R(\underline{n},T)}{\partial n_i} = \frac{\partial R(\underline{n},T)}{\partial R_i(n_i,T)} \cdot \frac{\partial R_i(n_i,T)}{\partial n_i}$$

$$= \left(R\Big|_{R_i=1} - R\Big|_{R_i=0}\right) \frac{\partial R_i(n_i,T)}{\partial n_i} \tag{E-8}$$

where $\dfrac{\partial R_i(n_i,T)}{\partial n_i}$ is given by either Equations (A-15) or (A-24).

4. $\int_0^T R(\underline{n},t)dt$ - This term is evaluated by numerical integration using Simpson's formula.

5. $$\frac{\partial}{\partial n_i} \int_0^T R(\underline{n},t)dt = \int_0^T \frac{\partial R(\underline{n},t)}{\partial n_i} dt$$

$$= \int_0^T \left(\frac{\partial R(\underline{n},t)}{\partial R_i(n_i,t)}\right)\left(\frac{\partial R_i(n_i,t)}{\partial n_i}\right) dt$$

$$= \int_0^T \left(R\Big|_{R_i=1} - R\Big|_{R_i=0}\right)\left(\frac{\partial R_i(n_i,t)}{\partial n_i}\right) dt$$

where $\dfrac{\partial R_i(n_i,t)}{\partial t}$ is given by either Equations (A-15) or (A-24) at any time t. Again the integral is evaluated by using Simpson's formula.

APPENDIX F

LAWLER AND BELL ALGORITHM

APPENDIX F

LAWLER AND BELL ALGORITHM

Lawler and Bell [32] describe a simple, easily programmed algorithm for solving discrete optimization problems with monotone objective functions and arbitrary constraints.

A brief review of the Lawler-Bell method is provided in this appendix. The type of problems that can be solved by this method may be put in the following form:

Minimize

$$g_0\ (\underline{x})$$

subject to m constraints of the form

$$g_{i1}(\underline{x}) - g_{i2}(\underline{x}) \geq 0 \qquad i=1,2,\ldots m$$

where

$$\underline{x} = (x_1, x_2, x_3, \ldots x_n) \tag{F-1}$$

and

$$x_j = 0 \text{ or } 1, \qquad j=1,2,\ldots n$$

and where the restriction is applied that each of the functions g_0, g_{i1}, g_{i2}, (i=1,2,...m) is monotone non-decreasing in each of the variables $x_1, x_2, \ldots x_n$. With some ingenuity, many problems can be put in this form.

Some preliminaries discussed below are essential for the description of the algorithm.

Vector $\underline{x}$ is "binary" in the sense that each x_j is either 0 or 1; $\underline{x} \leq \underline{y}$ if and only if $x_j \leq y_j$ for j=1,2,...n. This is the vector partial

ordering. There is also the lexicographic or numerical ordering of these vectors that is obtained by identifying with each vector $\underline{x}$, the integer value

$$N(\underline{x}) = x_1 2^{n-1} + x_2 2^{n-2} + \ldots + x_n 2^0 \qquad \text{(F-2)}$$

Numerical ordering is a refinement of the vector partial ordering; i.e. $\underline{x} \leq \underline{y}$ implies $N(\underline{x}) \leq N(\underline{y})$; however, $N(\underline{x}) \leq N(\underline{y})$ does not imply $\underline{x} \leq \underline{y}$.

Suppose all binary n- vectors are listed in numerical order; i.e.

(0,...,0,0,0),
(0,...,0,0,1),
(0,...,0,1,0),
(0,...,0,1,1),
(0,...,1,0,0),
etc.

Immediately following an arbitrary vector $\underline{x}$, there may (or may not) be a number of vectors $\underline{x}'$ with the property that $\underline{x} \leq x'$. Roughly speaking these are vectors that differ from $\underline{x}$ only in that they have 1's in place of one or more of the "right most" 0's of $\underline{x}$. For example, immediately following $\underline{x}$ = (0,1,0,0) are (0,1,0,1), (0,1,1,0), (0,1,1,1), each of which is greater than $\underline{x}$ in the vector partial ordering.

Let $\underline{x}^*$ denote the first vector following $\underline{x}$ in the numerical ordering that has the property that $\underline{x} \nleq \underline{x}^*$. For any given $\underline{x}$, the vector $\underline{x}^*$ is very easily calculated on the computer as follows:

Treat $\underline{x}$ as a binary number:

1. Subtract 1 from $\underline{x}$ to obtain $\underline{x} - 1$,
2. Logically OR $\underline{x}$ and $\underline{x} - 1$ to obtain $\underline{x}^* - 1$,

3. Add 1 to obtain $\underline{x}^*$.

Note that $\underline{x}^* - 1$ is greater than each of $\underline{x}$, $\underline{x} + 1, \ldots, \underline{x}^* - 2$ in the vector partial ordering.

The Lawler-Bell method is basically a search method, which starts with $\underline{x} = (0,0.,,,.0,0)$ and examines the 2^n solution vectors in the numerical ordering described above. Further, the labor of examination is considerably cut down by following certain rules. As the examination proceeds through the list of vectors, a record is kept of the least costly solution found to date. If $\hat{\underline{x}}$ denotes this solution having cost $g_0(\hat{\underline{x}})$ and $\underline{x}$ is the vector being examined, then the following steps indicate the conditions under which certain vectors in the numerical ordering can be skipped over:

1. If $g_0(\underline{x}) \geq g_0(\hat{\underline{x}})$, skip to $\underline{x}^*$ and repeat the operation; otherwise proceed to Step 2.

Justification: Because g_0 is monotone non-decreasing, none of the vectors $\underline{x} + 1$, $\underline{x} + 2, \ldots, \underline{x}^* - 1$ can be less costly than $\hat{\underline{x}}$.

2. If for any i $(i=1,2,\ldots m)$, $g_{i1}(\underline{x}^* - 1) - g_{i2}(\underline{x}) < 0$, skip to $\underline{x}^*$ and go to Step 1; otherwise proceed to Step 3.

Justification: With respect to vectors in the interval $(\underline{x}, \underline{x}^* - 1)$, $\underline{x}^* - 1$ maximizes g_{i1} and $\underline{x}$ maximizes $-g_{i2}$. Hence if it is the case that $g_{i1}(\underline{x}^* - 1) - g_{i2}(\underline{x}) < 0$, there can be no vector $\underline{x}'$ in the interval such that $g_{i1}(\underline{x}') - g_{i2}(\underline{x}') \geq 0$.

3. If $g_{i1}(\underline{x}) - g_{i2}(\underline{x}) \geq 0$, $(i=1,2,\ldots,m)$, replace $\hat{\underline{x}}$ by $\underline{x}$ and skip to x^*; otherwise change $\underline{x}$ to $\underline{x} + 1$. In either case further execution is transferred to Step 1.

Justification: Because g_0 is monotone non-decreasing, none of the vectors $\underline{x} + 1$, $\underline{x} + 2, \ldots, \underline{x}^* - 1$ can be less costly than $\underline{x}$.

Following the above steps, all the vectors are examined and scanning continues until a vector having maximum numerical order. Viz., (1,1,...,1) is found. In case one has skipped to a vector having numerical order higher than (1,1,...,1), designate this state by "overflow" and terminate the procedure. The least "costly" vector recorded provides the optimum solution.

Problems involving non-negative integer variables can be transformed into problems involving binary variables by means of the substitution

$$x_j = x_{j1} + 2x_{j2} + 2^2 x_{j3} + \ldots + 2^{K-1} x_{jK} \tag{F-3}$$

where $x_{ji} = 0$ or 1 ($i = 1,2,\ldots K$). K is chosen to be sufficiently large for $2^K - 1$ to be an upper bound on the value of x_j.

The algorithm was seen to be most efficient if those variables that were roughly speaking "least significant" were assigned positions at the "right" end of the solution vector. The ordering of vector $\underline{x}$ is shown below

$$\underline{x} = (x_{1K}, x_{2K}, \ldots, x_{nK}, x_{1(K-1)}, \ldots, x_{n2}, x_{11}, x_{21}, \ldots, x_{n1}) \tag{F-4}$$

APPENDIX G

COMPUTER PROGRAMS

APPENDIX G

COMPUTER PROGRAMS

This appendix contains the documentation for two computational schemes which solve the system optimization problems in the text.

Each scheme is composed of subroutines. Most of the subroutines do not have calling arguments--the greater part of the communication between subroutines being handled through COMMON.

Computational Scheme I

Computational Scheme I is based on the modified integer gradient method. The details of the method are given in Appendix D. A schematic diagram of the main program is shown in Figure G.2. The computation is composed of fifteen subroutines. Depending on the objective function and the constraints, the appropriate statements are modified in the various subroutines. A computer listing is provided for each of the three system optimization problems discussed in Chapter III of the text. Subroutines PATHS, PATHP, TRACE, and PATHC are identical for the three cases. A functional description of the various subroutines and a list of the major program variables is given below.

SUBROUTINES

DATAIN - Reads and echo checks data.

PATHS - Argument (INPUT). Finds all paths in the reliability flow graph which start at a specified INPUT node. See Appendix B for details.

PATHP - Argument (OUTPUT). Itemizes all paths starting at an INPUT node (defined by a call to subroutine PATHS) and ending at the OUTPUT node specified.

TRACE - Argument (I). Retrieves the nodes and the modules associated with the Ith entry in the P-array and prints them in sequence. Also converts each path in the binary form.

PATHC - Finds the binary representation of all the path combinations with appropriate signs. See Appendix C for details.

RANDU - Arguments (IX, IY, IFL). Generates uniformity distributed number between zero and one.

MODULE - Computes the module reliability and the sensitivity of the module reliability to the module redundancy for all the system modules. Only active redundancy modules or cold standby redundancy modules are considered. See Appendix A for details.

RELIAB - Computes the system reliability from the module reliabilities found earlier.

SENSE - Determines the sensitivity of the system reliability to the module redundancies.

AVLBTY - Determines the system availability, system profit, and the sensitivity of the system availability and the system profit to the module redundancies and to the maintenance interval. The integrals are evaluated by using Simpson's formula. See Appendix E for details.

CONSTR - Computes the system constraints.

FESIBL - Finds the feasible solution by the method of weighted perpendiculars. See Appendix D for details.

SORT - Sorts the system sensitivity vector in ascending order.

OPTIMZ - Finds a locally optimal solution from the feasible solution. The computation is based on the modified integer gradient method discussed in Appendix D.

OPTIM - Checks to see if the movement in the gradient direction gives an improved feasible solution.

PROGRAM VARIABLES

An explanation of the major program variables is given below.

INNOD - Input node.

OUTNOD - Output node.

FOR - Number of forward edges.

FREV - Row number of the first reverse edge appearing in the P-array.

LEND - Row number of the last entry in the P-array.

NEDGES - Total number of edges or branches in the reliability graph.

NEDGS1 - NEDGES + 1.

NNODES - Total number of nodes.

NPATH - Total number of paths between the input and the output node.

PATHL - Row dimension of the PATH-array. Length of vector PSIGN.

PLENTH - Row dimension of the P-array.

PPL - Dimension of PP, PI4, NUNIT, NMAX, NMIN, MUNIT, RUNIT, WUNIT, LAMDA, G, AGRADN, RGRADN, PGRADN, RIGRDN, MTGRDN, COSTF, and NSORT vectors. Also the column dimension of PATH-array.

REV - Number of reverse edges.

STATUS - Flag which is set to zero if an attempt has been made to overflow the PATH-array or the P-array.

P - Array of dimension P (PLENTH,4) which contains the list structure shown in Figure G-1.

PP - Auxiliary vector.

PI4 - Auxiliary vector.

PATH - Array of dimension (PATHL, PPL). Stores each path and path combinations in binary form.

NMOD - Total number of modules.

NMOD1 - NMOD + 1.

MPATH - Total number of paths and path combinations.

PSIGN - Vector. Stores the number + 1 or -1 associated with each path and path combination.

NUNIT - Vector. Stores the module redundancies (number of redundant units in each module).

NMAX - Vector. Contains the upper limit of the module redundancies.

NMIN - Vector. Contains the lower limit of the module redundancies.

MUNIT - Vector. Stores the module redundancies necessary for successful operation of each module.

P-ARRAY

		ORIGIN	END	POINTER	MODULE
Edges which constitute the graph	1				
	⋮				
	FREV				
	⋮				
	NEDGES				
List structure containing the paths starting at the input node	NEDGS1				
	⋮				
	LEND				

Figure G.1

List Structure of P-Array

KODE - Vector. Stores the binary numbers zero or one. A "zero" identifies an active redundancy module. A "one" identifies a cold standby redundancy module.

ASYS - System availability.

RSYS - System reliability.

PSYS - System profit.

RUNIT - Vector. Stores unit reliabilities.

WUNIT - Vector. Stores unit weights.

LAMDA - Vector. Stores unit failure rates.

G - Vector. Contains module reliabilities.

AGRADN - Vector. Contains sensitivity of system availability to the module redundancies.

AGRADT - Sensitivity of system availability to the maintenance interval.

PGRADN - Vector. Contains sensitivity of system profit to the module redundancies.

PGRADT - Sensitivity of system profit to the maintenance interval.

RGRADN - Vector. Contains sensitivity of system reliability to the module redundancies.

RGRADT - Sensitivity of system reliability to the maintenance interval.

RIGRDN - Vector. Contains sensitivity of module reliability to its own redundancy.

MTGRDN - Vector. Stores sensitivity of MTBM to module redundancies.

COSTF - Vector. Stores unit costs.

CONSTR - Vector 10 locations in length. Stores values of the constraints.

MTBM - Mean time between maintenance.

MDT - Mean down time.

OBJFN - System objective function.

RLBTY - System reliability.

CMCOST - Upper limit on the value of cost constraint.

WEIGHT - Upper limit on the value of weight constraint.

INCOM - Net income per hour of system operation.

COSTMF - Maintenance cost as a per cent of system cost per hour of system down time.

T - Variable of integration.

TEM - Emergency maintenance interval.

TSC - Scheduled maintenance interval.

TLF - System life time.

TM - Maintenance interval.

TMAX - Upper limit on the value of TM.

TMIN - Lower limit on the value of TM.

INTVL - Interval of integration used in the Simpson's formula.

TSTEP - Constant to express maintenance interval in terms of (TM/TSTEP).

FCOST - System fixed capital cost.

MCOST - System maintenance cost.

KOUNT - Number of random sample points.

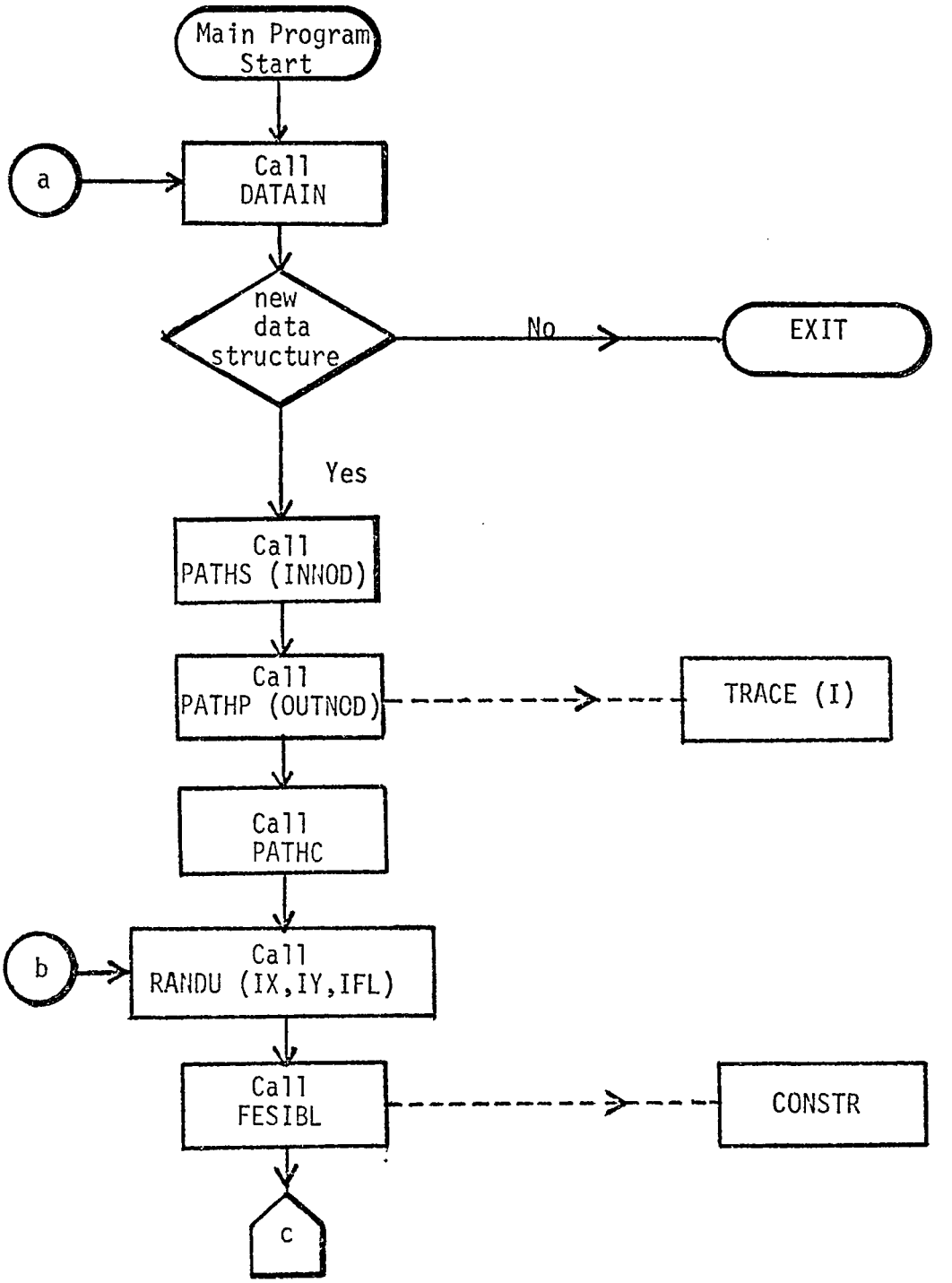
NONZRO - Number of nonzero elements in the system sensitivity vector.

NSORT - Vector. Stores the module numbers in ascending order of magnitude of the system sensitivity to module redundancies.

FLAG - It is either zero or one. When it is "one" only system availability and system profit are evaluated by subroutine AVLBTY. When it is "zero" sensitivity of system availability and system profit to module redundancies and maintenance interval are also evaluated.

INDEX - A value equal to NMOD1 indicates that a feasible point could not be located.

MOVE - As used in subroutine FESIBL - a value greater than 20 indicates that FESIBL point could not be located. As used in subroutines OPTIM and OPTIMZ - a value equal to zero indicates that an improved feasible solution is not possible for a particular gradient vector.


Main Program Start
a
Call DATAIN
new data structure
No
EXIT
Yes
Call PATHS (INNOD)
Call PATHP (OUTNOD)
TRACE (I)
Call PATHC
b
Call RANDU (IX,IY,IFL)
Call FESIBL
CONSTR
c

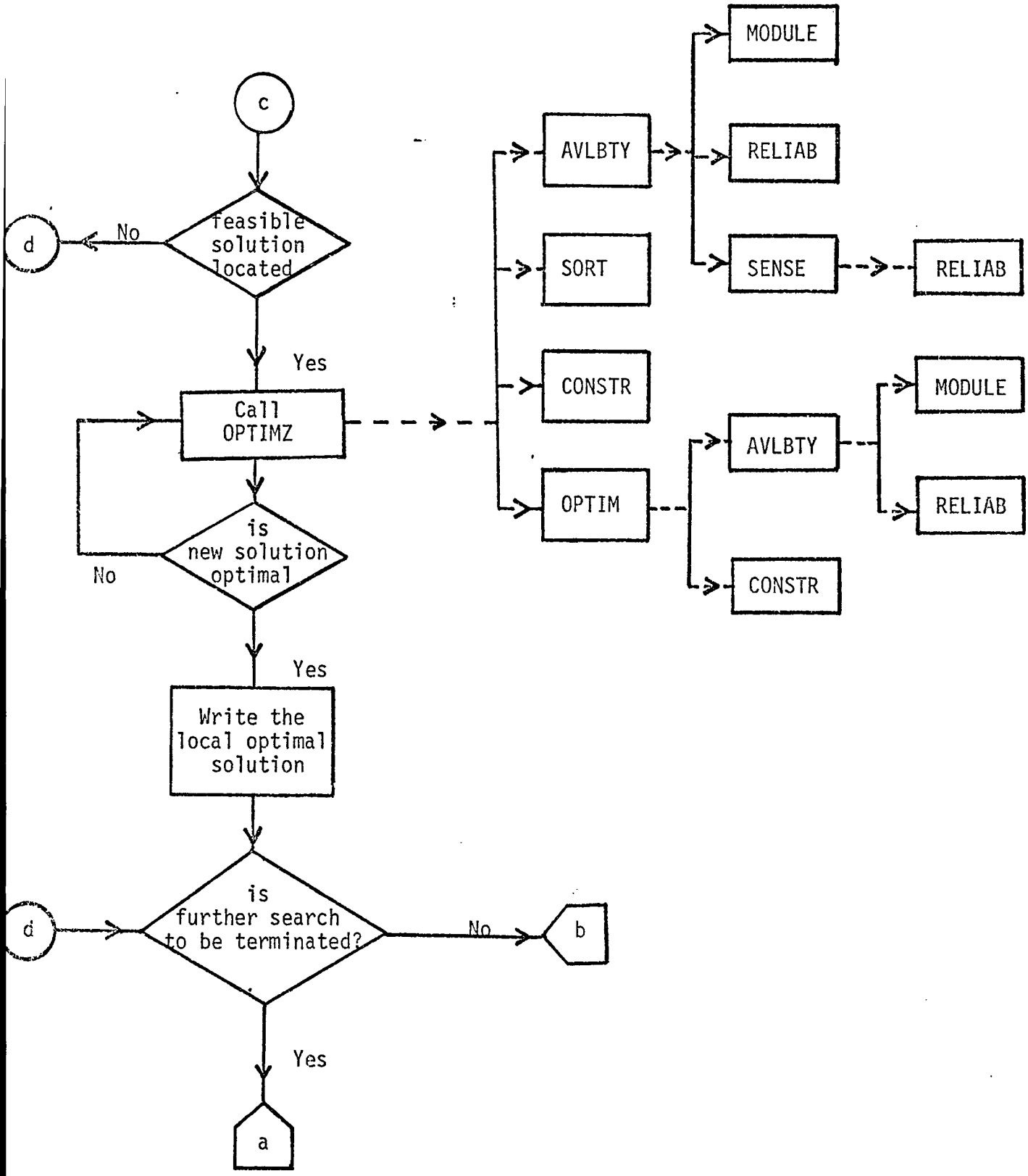


Figure G.2

Schematic Diagram of Computation I

---- indicates communication between subroutines.

Computational Scheme II

Computational Scheme II is based on the partial enumeration method of Lawler and Bell. Details of the method are given in the Appendix F and was used to solve the system reliability optimization problems discussed in Chapters III and IV of the text.

A schematic diagram of the main program is shown in Figure G-3. The computation is composed of seven subroutines. These include DATAIN, PATHS, PATHP, TRACE, PATHC, MODULE and RELIAB. A functional description of each has been presented with the first computational scheme discussed earlier in this appendix. Subroutines PATHS, PATHP, PATHC, TRACE are identical to the ones listed earlier.

PROGRAM VARIABLES

A list of major program variables and their explanations are given below:

RKODE(I) - Binary number zero or one. A "zero" identifies an active redundancy Ith module. A "one" identifies a standby redundancy Ith module.

NKODE(I) - Number of binary variables associated with the number of units in the Ith module.

MKODE(I) - Number of binary variables associated with the number of units necessary for successful operation of the Ith module.

NTEST(I) - Number of units in the Ith module. Generated from the binary variables NXTEST (I,J).

NSTAR1(I)- Number of units in the Ith module. Generated from the binary variables NXSTR1(I,J).

MTEST(I) - Number of units in the Ith module that must operate for its successful operation. Generated from the binary variables MXTEST(I,J).

MSTAR1(I)- Number of units in the Ith module that must operate for its successful operation. Generated from binary variable MXSTR1(I,J).

NXTEST(I,J) - jth binary variable associated with the variable NTEST(I). It is obtained from the vector XTEST.

MXTEST(I,J) - jth binary variable associated with the variable MTEST(I). It is obtained from the vector XTEST.

NXSTR1(I,J) - jth binary variable associated with the variable NSTAR1(I). It is obtained from the vector XSTR1.

MXSTR1(I,J) - jth binary variable associated with the variable MSTAR1(I). It is obtained from the vector XSTR1.

XTEST - Vector containing NVAR binary system variables.

XTEST1 - Vector obtained by the binary subtraction of (XTEST - XONE).

XONE - Vector with the binary number 1 in the first position and zero in all other positions.

XSTR1 - Vector obtained by performing logical "OR" operation on XTEST and XTEST1.

XSTAR - Vector obtained by binary addition of (XSTR1 + XONE).

XHAT - Vector. Stores the previous best value of vector XTEST.

NHAT - Vector. Stores the previous best value of vector NTEST.

MHAT - Vector. Stores the previous best value of vector MTEST.

R(I) - Unit reliability of the Ith module.

G(I) - Ith module reliability.

GHAT(I) - Previous best value of G(I).

KINDEX(I) - Constant "k_c" associated with the unit cost of the Ith module.

SINDEX(I) - Constant "s" associated with the cost of the Ith module.

CONSTR(I) - Value of the Ith system constraint.

COST - Upper limit on the total system cost.

COSTHT - Total system cost corresponding to the system variables NHAT and MHAT.

RLBTY - System reliability.

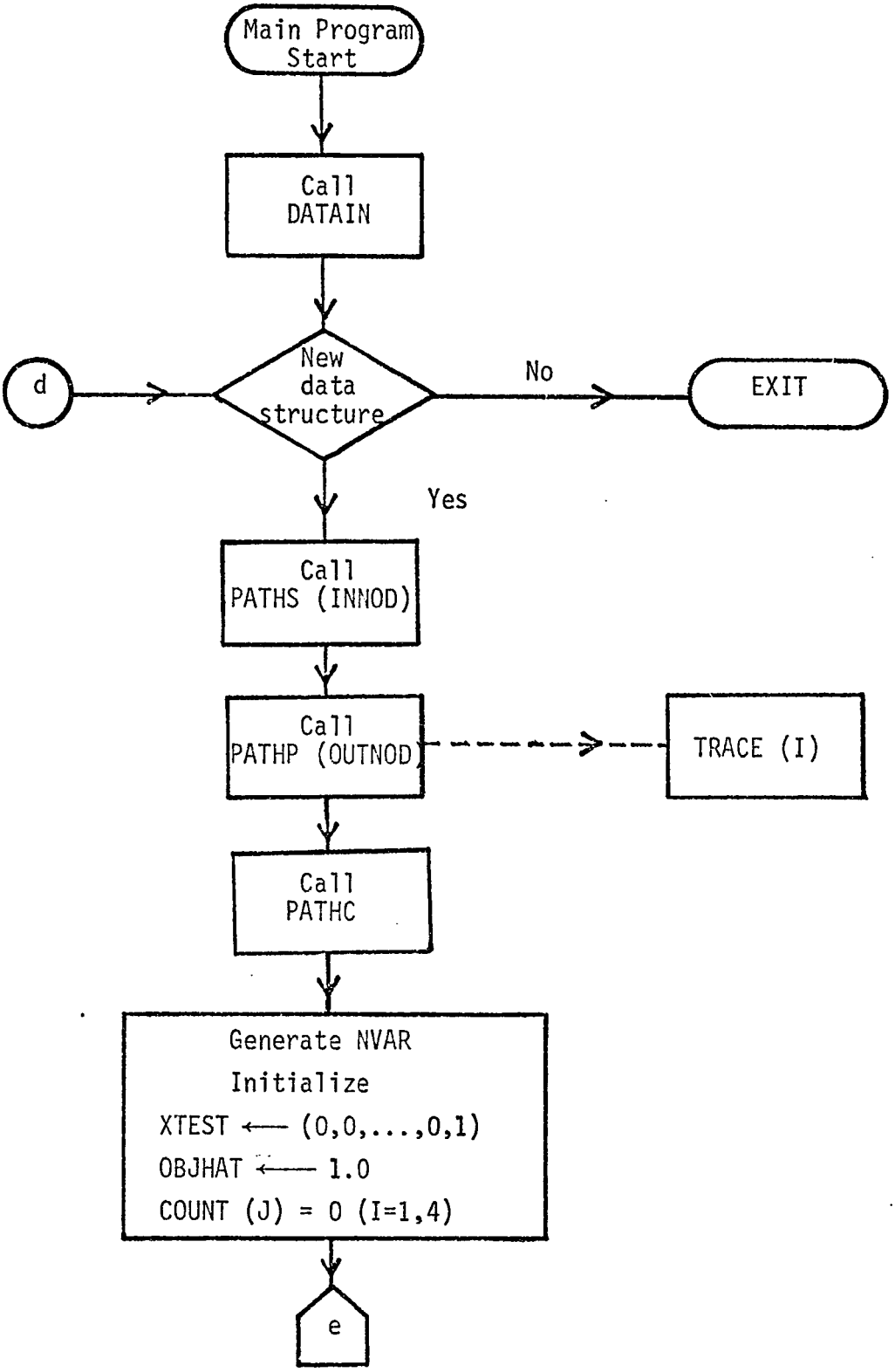
OBJFN - Given by (1. - RLBTY).

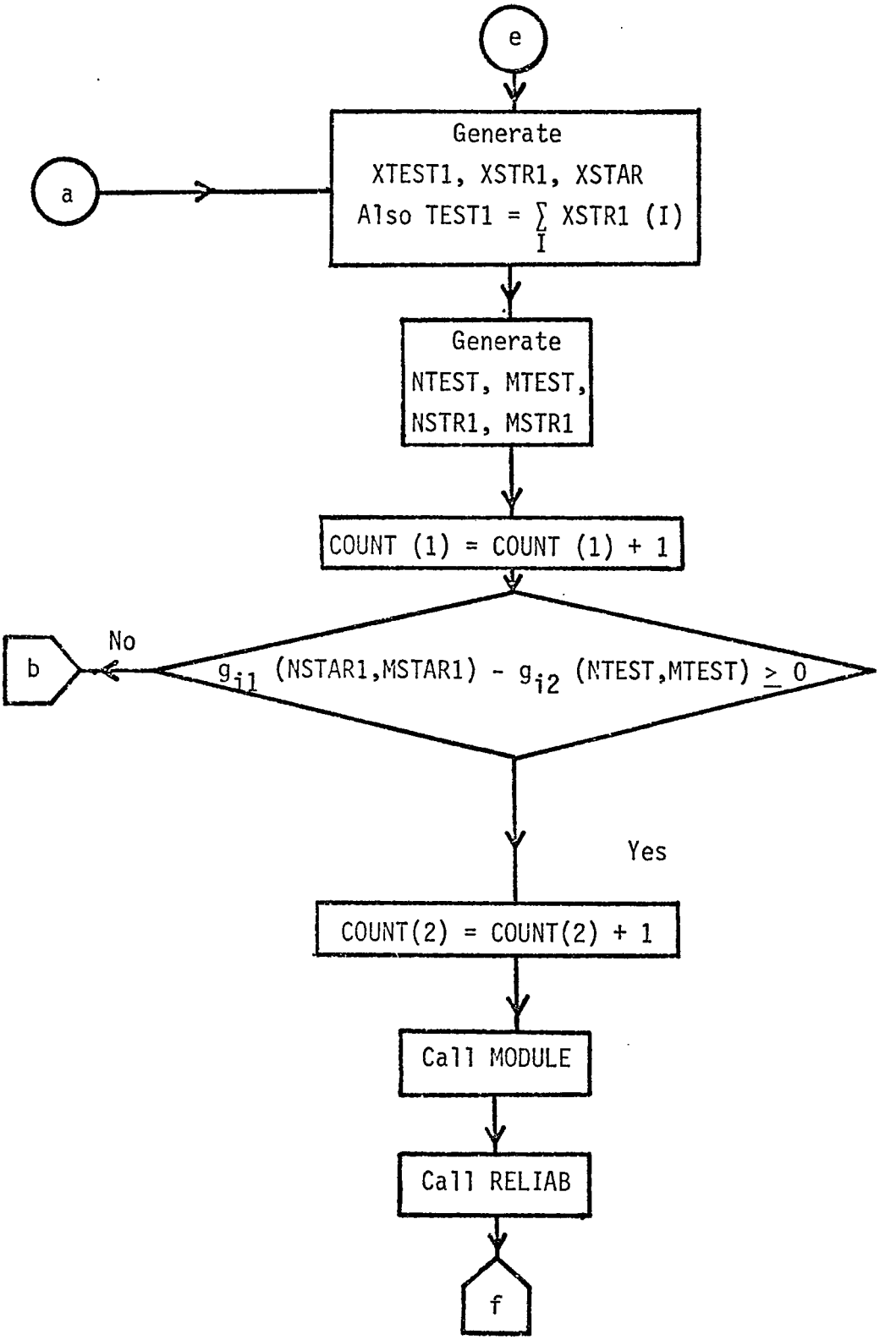
OBJHAT - Previous best value of OBJFN.

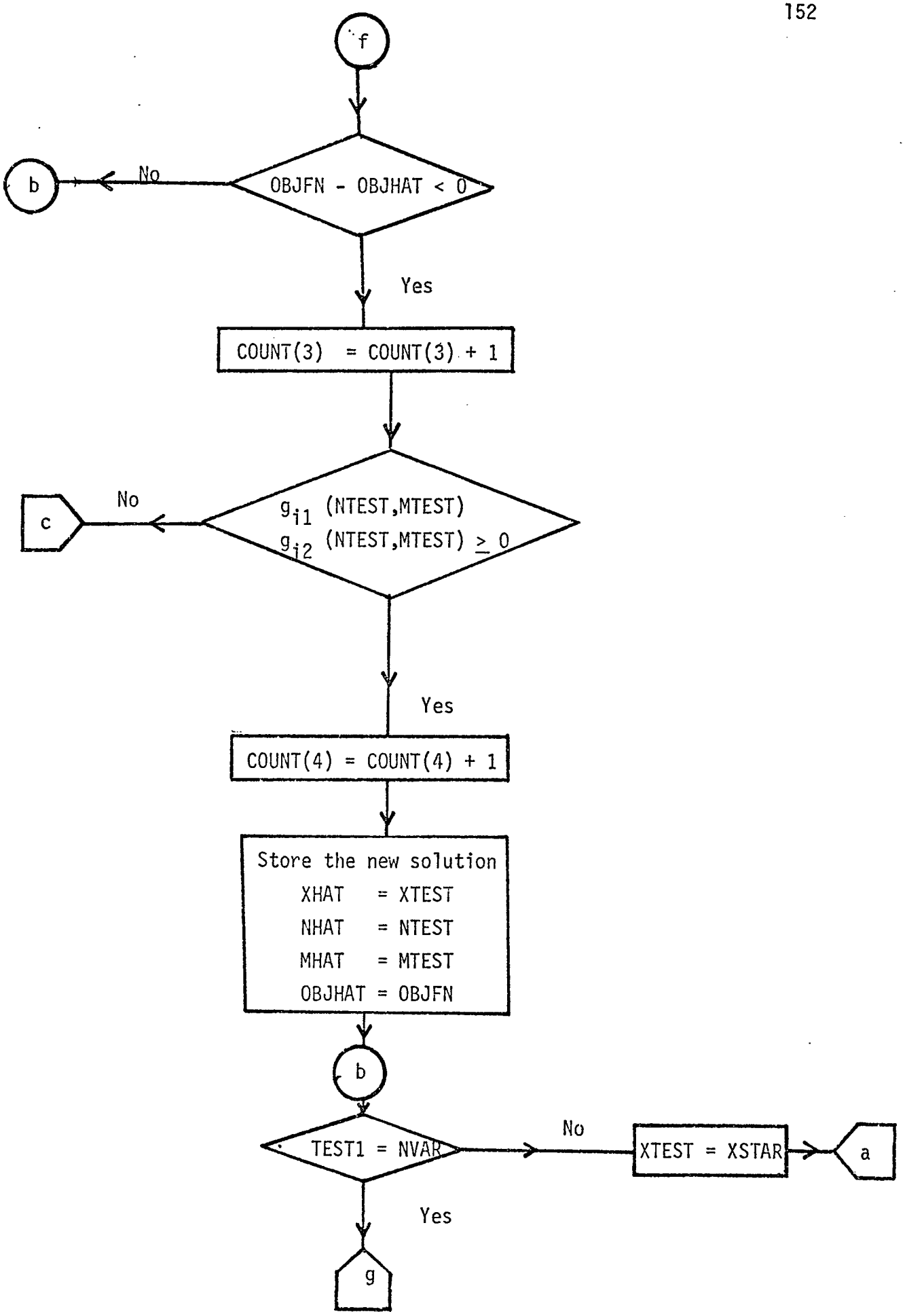
NVAR - Total number of binary system variables.

TEST1 and TEST2 - Variables to see if all the solution vectors have been examined.

COUNT - Vector of dimension four. Used for counting the number of operations at various stages of the computation.


Main Program Start
Call DATAIN
d
New data structure
No
EXIT
Yes
Call PATHS (INNOD)
Call PATHP (OUTNOD)
TRACE (I)
Call PATHC
Generate NVAR
Initialize
XTEST ← (0,0,...,0,1)
OBJHAT ← 1.0
COUNT (J) = 0 (I=1,4)
e


e
a
Generate
XTEST1, XSTR1, XSTAR
Also TEST1 = $\sum_I$ XSTR1 (I)
Generate
NTEST, MTEST,
NSTR1, MSTR1
COUNT (1) = COUNT (1) + 1
No
b
g_{i1} (NSTAR1,MSTAR1) - g_{i2} (NTEST,MTEST) $\geq$ 0
Yes
COUNT(2) = COUNT(2) + 1
Call MODULE
Call RELIAB
f


f
OBJFN - OBJHAT < 0
No
b
Yes
COUNT(3) = COUNT(3) + 1
g_{i1} (NTEST,MTEST)
g_{i2} (NTEST,MTEST) ≥ 0
No
c
Yes
COUNT(4) = COUNT(4) + 1
Store the new solution
XHAT = XTEST
NHAT = NTEST
MHAT = MTEST
OBJHAT = OBJFN
b
TEST1 = NVAR
No
XTEST = XSTAR
a
Yes
g

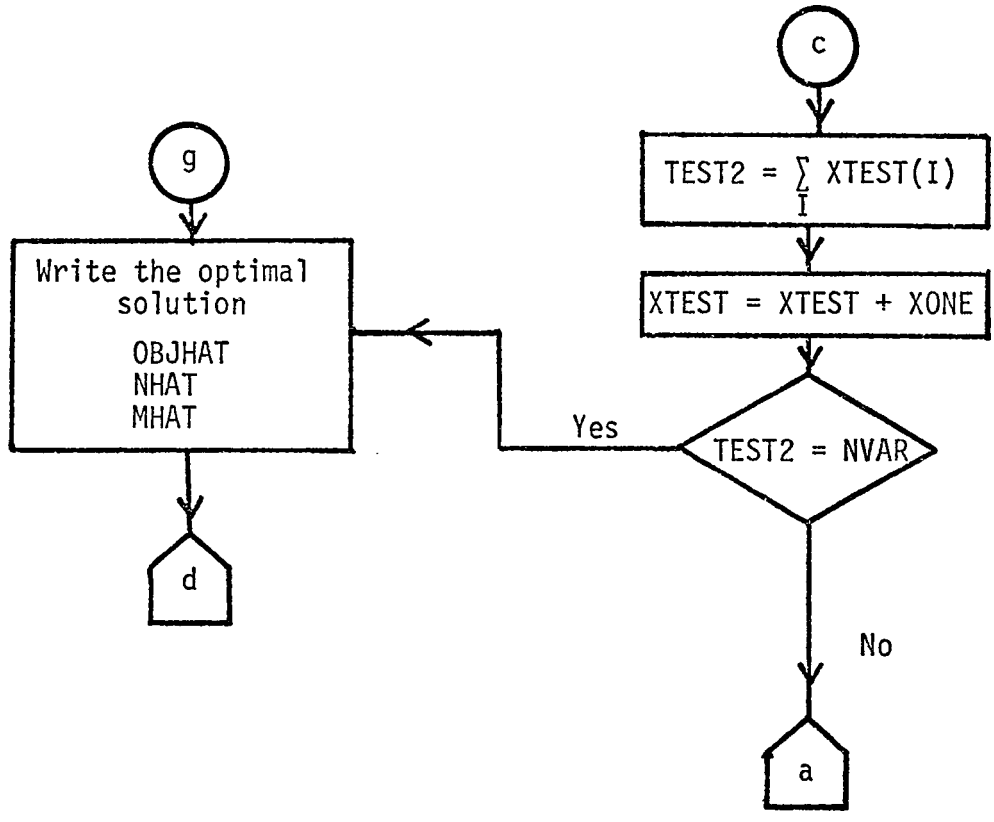


Figure G.3

Schematic diagram of Computation II

COMPUTER LISTING FOR COMPUTATION SCHEME I

SYSTEM RELIABILITY OPTIMIZATION

```
C     MAIN PROGRAM                                                      MAIN  10
C     SYSTEM RELIABILITY OPTIMIZATION                                   MAIN  20
C     MODIFIED INTEGER GRADIENT METHOD                                  MAIN  30
      IMPLICIT INTEGER*4 (A-Z)                                          MAIN  40
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  MAIN  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             MAIN  60
     2  PATH(300,50),NMOD,MPATH,PSIGN(300)                              MAIN  70
      INTEGER*2 P,PP,PATH                                               MAIN  80
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     MAIN  90
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 MAIN 100
      COMMON/OPTM2/CONST(10),COSTF(50),G(50),RGRADN(50),RIGRDN(50),     MAIN 110
     1  RUNIT(50),WUNIT(50),CMCOST,OBJFN,RLBTY,WEIGHT                   MAIN 120
      REAL*4 CONST,COSTF,G,RGRADN,RIGRDN,RUNIT,WUNIT,CMCOST,OBJFN,      MAIN 130
     1  RLBTY,WEIGHT                                                    MAIN 140
      REAL*4 YFL                                                        MAIN 150
C                                                                       MAIN 160
      PATHL = 300                                                       MAIN 170
      PLENTH = 1000                                                     MAIN 180
      PPL = 50                                                          MAIN 190
   10 CALL DATAIN                                                       MAIN 200
      READ(5,1000) INNOD,OUTNOD                                         MAIN 210
      CALL PATHS(INNOD)                                                 MAIN 220
      IF(STATUS) 20,10,20                                               MAIN 230
   20 CALL PATHP(OUTNOD)                                                MAIN 240
      WRITE(6,1100)                                                     MAIN 250
      DO 30 L=1,NPATH                                                   MAIN 260
      WRITE(6,1200) L,(PATH(L,M),M=1,NMOD  )                            MAIN 270
   30 CONTINUE                                                          MAIN 280
      CALL PATHC                                                        MAIN 290
      WRITE(6,1001) NPATH,MPATH                                         MAIN 300
      WRITE(6,2600)                                                     MAIN 310
      DO 35 L=1,MPATH                                                   MAIN 320
      WRITE(6,2700) L,PSIGN(L),(PATH(L,M),M=1,NMOD)                     MAIN 330
   35 CONTINUE                                                          MAIN 340
      KOUNT = 1                                                         MAIN 350
```

```
      ISEED = 984376521                                                 MAIN 360
      IX = ISEED                                                        MAIN 370
      DO 45 I=1,NMOD                                                    MAIN 380
   45 NUNIT(I) = 1                                                      MAIN 390
      GO TO 55                                                          MAIN 400
C     GENERATING A UNIFORMLY DISTRIBUTED RANDOM STARTING POINT          MAIN 410
   40 DO 50 I=1,NMOD                                                    MAIN 420
      CALL RANDU(IX,IY,YFL)                                             MAIN 430
      NUNIT(I) = YFL*(NMAX(I)- NMIN(I)+1) + NMIN(I)                     MAIN 440
      IX = IY                                                           MAIN 450
   50 CONTINUE                                                          MAIN 460
   55 WRITE (6,1300) KOUNT                                              MAIN 470
      WRITE(6,2100)                                                     MAIN 480
      WRITE(6,1400) (NUNIT(I),I=1,NMOD  )                               MAIN 490
C     LOCATING A FEASIBLE POINT                                         MAIN 500
      WRITE(6,2200)                                                     MAIN 510
      CALL FESIBL                                                       MAIN 520
      WRITE(6,1410) MOVE,INDEX                                          MAIN 530
      IF (MOVE .GT. 20  .OR.  INDEX .EQ. NMOD1 )  GO TO 90              MAIN 540
      GO TO 60                                                          MAIN 550
   60 WRITE(6,2400)                                                     MAIN 560
      WRITE(6,1400) (NUNIT(I),I=1,NMOD  )                               MAIN 570
      WRITE(6,1600)  CONST(1),CONST(2)                                  MAIN 580
C     SEARCHING FOR LOCAL OPTIMUM                                       MAIN 590
      WRITE(6,2300)                                                     MAIN 600
   65 CALL OPTIMZ                                                       MAIN 610
      IF(MOVE .EQ. 0) GO TO 70                                          MAIN 620
      GO TO 65                                                          MAIN 630
   70 WRITE(6,2500)                                                     MAIN 640
      WRITE(6,1400) (NUNIT(I),I=1,NMOD  )                               MAIN 650
      CALL MODULE                                                       MAIN 660
      CALL RELIAB                                                       MAIN 670
      CALL CONSTR                                                       MAIN 680
      WRITE(6,1500) RLBTY                                               MAIN 690
      WRITE(6,1600)  CONST(1),CONST(2)                                  MAIN 700
   90 KOUNT = KOUNT + 1                                                 MAIN 710
```

```
      IF (KOUNT .GT. 25) GO TO 10                                       MAIN 720
      GO TO 40                                                          MAIN 730
 1000 FORMAT (2I3)                                                      MAIN 740
 1001 FORMAT(1H0,//,30X,'NPATH = ',I3,//,30X,'MPATH = ',I5)             MAIN 750
 1100 FORMAT (1H0,//,T20,'PATHS IN BINARY CODE',/)                      MAIN 760
 1200 FORMAT (1H0,T16,'PATH',I3,' =  ',50I2)                            MAIN 770
 1300 FORMAT(1H1,40X,'KOUNT =',I3,//)                                   MAIN 780
 1400 FORMAT(1H0,/,10X,'NUNIT(I) =',20I5)                               MAIN 790
 1410 FORMAT (1H0,//,30X,'MOVE =',I4,10X,'INDEX =',I4)                  MAIN 800
 1500 FORMAT (1H0,//,30X,'SYSTEM RELIABILITY =',G13.6)                  MAIN 810
 1600 FORMAT(1H0,/,30X,'CONSTRAINT(1) =',G13.6,//,30X,'CONSTRAINT(2) =',MAIN 820
     1  G13.6)                                                          MAIN 830
 2100 FORMAT(1H0,50X,'THE INITIAL POINT IS')                            MAIN 840
 2200 FORMAT(1H0,50X,'SEARCHING FOR THE FEASIBLE POINT')                MAIN 850
 2300 FORMAT(1H0,50X,'SEARCHING FOR THE LOCAL OPTIMUM POINT')           MAIN 860
 2400 FORMAT(1H0,////,50X,'THE FEASIBLE POINT IS')                      MAIN 870
 2500 FORMAT(1H0,////,50X,'THE OPTIMAL POINT IS')                       MAIN 880
 2600 FORMAT(1H0,//,T20,'PATHS AND PATH COMBINATIONS IN BINARY CODE',/) MAIN 890
 2700 FORMAT(1H0,T16,'PATHC',I3,' = ',I2,4X,50I2)                       MAIN 900
      END                                                               MAIN 910
```

```
      SUBROUTINE DATAIN                                                 DATA  10
C     THIS SUBROUTINE READS AND ECHOCHECKS THE DATA                     DATA  20
      IMPLICIT INTEGER*4 (A-Z)                                          DATA  30
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  DATA  40
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             DATA  50
     2  PATH(300,50),NMOD,MPATH,PSIGN(300)                              DATA  60
      INTEGER*2 P,PP,PATH                                               DATA  70
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     DATA  80
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 DATA  90
      COMMON/OPTM2/CONST(10),COSTF(50),G(50),RGRADN(50),RIGRDN(50),     DATA 100
     1  RUNIT(50),WUNIT(50),CMCOST,OBJFN,RLBTY,WEIGHT                   DATA 110
      REAL*4 CONST,COSTF,G,RGRADN,RIGRDN,RUNIT,WUNIT,CMCOST,OBJFN,      DATA 120
     1  RLBTY,WEIGHT                                                    DATA 130
      DIMENSION TITLE(30)                                               DATA 140
      READ(5,999,END=100)( TITLE(I),I=1,30)                             DATA 150
      READ(5,1000) NNODES,NMOD,CMCOST,WEIGHT                            DATA 160
      DO 10 I=1,NMOD                                                    DATA 170
   10 READ(5,1007) KODE(I),NMAX(I),NMIN(I),MUNIT(I),COSTF(I),WUNIT(I),  DATA 180
     1  RUNIT(I)                                                        DATA 190
C                                                                       DATA 200
      FOR=0                                                             DATA 210
      REV=0                                                             DATA 220
      DO 40 I=1,1000                                                    DATA 230
      READ(5,1001)P(I,1),P(I,2),P(I,4)                                  DATA 240
      IF( P(I,1)-P(I,2) ) 20,50,30                                      DATA 250
   20 FOR=FOR + 1                                                       DATA 260
      GO TO 40                                                          DATA 270
   30 REV=REV + 1                                                       DATA 280
   40 CONTINUE                                                          DATA 290
   50 NEDGES=FOR + REV                                                  DATA 300
      NEDGS1=NEDGES + 1                                                 DATA 310
      FREV=FOR + 1                                                      DATA 320
      WRITE(6,1002) ( TITLE(I),I=1,30)                                  DATA 330
      WRITE(6,1003)                                                     DATA 340
      WRITE(6,1004)                                                     DATA 350
```

```
      WRITE(6,1003)                                                     DATA 360
      DO 60 I=1,NEDGES                                                  DATA 370
      WRITE(6,1005) I,P(I,1),P(I,2),P(I,4)                              DATA 380
      WRITE(6,1003)                                                     DATA 390
   60 CONTINUE                                                          DATA 400
      WRITE(6,1008)                                                     DATA 410
      WRITE(6,1011)                                                     DATA 420
      WRITE(6,1009)                                                     DATA 430
      WRITE(6,1011)                                                     DATA 440
      DO 70 I=1,NMOD                                                    DATA 450
      WRITE(6,1010) I,KODE(I),NMAX(I),NMIN(I),MUNIT(I),COSTF(I),        DATA 460
     1  WUNIT(I),RUNIT(I)                                               DATA 470
      WRITE(6,1011)                                                     DATA 480
   70 CONTINUE                                                          DATA 490
      WRITE(6,1006) CMCOST,WEIGHT                                       DATA 500
  999 FORMAT(15A4)                                                      DATA 510
 1000 FORMAT (2I10,2F10.0)                                              DATA 520
 1001 FORMAT (3I10)                                                     DATA 530
 1002 FORMAT(1H1,//,T10,30A4,//,T50,'INVENTORY OF BRANCHES',/)          DATA 540
 1003 FORMAT (1H+,T38,44('_'))                                          DATA 550
 1004 FORMAT (1H ,T38,'|  BRANCH  |','  ORIGIN  |','   END    |',       DATA 560
     1  '  MODULE  |')                                                 DATA 570
 1005 FORMAT (1H ,T38,4('|',4X,I2,4X),'|')                              DATA 580
 1006 FORMAT (1H0,T10,'CMCOST=',G13.6,/,T10,'WEIGHT=',G13.6)            DATA 590
 1007 FORMAT(4I10,3F10.0)                                               DATA 600
 1008 FORMAT(1H0,T50,'INVENTORY OF MODULES',/)                          DATA 610
 1009 FORMAT(1H ,T20,'|  MODULE  |','   KODE   |',' MAX-UNIT |',        DATA 620
     1  ' MIN-UNIT |','M OUT OF N|','UNIT COSTF|','UNIT WEGHT|',       DATA 630
     1  ' RELIABILITY |')                                               DATA 640
 1010 FORMAT(1H ,T20,5('|',4X,I2,4X),2('|',F10.3),'|',G13.6,'|')        DATA 650
 1011 FORMAT(1H+,T20,91('_'))                                           DATA 660
      RETURN                                                            DATA 670
  100 CALL EXIT                                                         DATA 680
      END                                                               DATA 690
```

```
      SUBROUTINE PATHS(INPUT)                                           PATH  10
C     THIS SUBROUTINE ACCEPTS AN INPUT NODE AND FINDS ALL PATHS         PATH  20
C     STARTING AT THIS NODE                                             PATH  30
      IMPLICIT INTEGER*4 (A-Z)                                          PATH  40
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  PATH  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             PATH  60
     2  PATH(300,50),NMOD,MPATH,PSIGN(300)                              PATH  70
      INTEGER*2 P,PP,PATH                                               PATH  80
C                                                                       PATH  90
      DO 5 I=1,NNODES                                                   PATH 100
    5 PP(I)=0                                                           PATH 110
      PP(INPUT)=1                                                       PATH 120
    7 A1=NEDGS1                                                         PATH 130
      CYCLE=0                                                           PATH 140
      STATUS=1                                                          PATH 150
      MARK=0                                                            PATH 160
      NSTART=A1                                                         PATH 170
   10 CYCLE=CYCLE + 1                                                   PATH 180
      DO 100 I=1,NEDGES                                                 PATH 190
      IF( (I.NE.1).AND.(I.NE.FREV) ) GO TO 20                           PATH 200
      IF( MARK.LT.0 ) GO TO 120                                         PATH 210
      ISTART=NSTART                                                     PATH 220
      NSTART=A1                                                         PATH 230
      MARK=-1                                                           PATH 240
   20 FIN=A1 - 1                                                        PATH 250
      IF( (CYCLE.GT.1).OR.(I.GT.FOR) ) GO TO 30                         PATH 260
      J=P(I,1)                                                          PATH 270
      IF( PP(J).EQ.0 ) GO TO 30                                         PATH 280
      P(A1,1)=P(I,1)                                                    PATH 290
      P(A1,2)=P(I,2)                                                    PATH 300
      P(A1,3)=A1                                                        PATH 310
      P(A1,4)=P(I,4)                                                    PATH 320
      A1=A1 + 1                                                         PATH 330
      MARK=0                                                            PATH 340
   30 A2=ISTART                                                         PATH 350
```

```
   40 IF( A2.GT.FIN ) GO TO 100                                    PATH 360
      IF( P(A2,2).NE.P(I,1) ) GO TO 90                             PATH 370
      IF( P(I,2)-P(A2,1) ) 90,90,50                                PATH 380
   50 IF( (CYCLE.EQ.1).AND.(I.LE.FOR) ) GO TO 80                   PATH 390
      A3=P(A2,3)                                                   PATH 400
   60 IF( P(A3,2).EQ.P(I,2) ) GO TO 90                             PATH 410
      IF( A3.EQ.P(A3,3) ) GO TO 80                                 PATH 420
      A3=P(A3,3)                                                   PATH 430
      GO TO 60                                                     PATH 440
   80 MARK=0                                                       PATH 450
      P(A1,1)=P(A2,1)                                              PATH 460
      P(A1,2)=P(I,2)                                               PATH 470
      P(A1,3)=A2                                                   PATH 480
      P(A1,4)=P(I,4)                                               PATH 490
      A1=A1 + 1                                                    PATH 500
      IF( A1.GT.PLENTH ) GO TO 110                                 PATH 510
   90 A2=A2 + 1                                                    PATH 520
      GO TO 40                                                     PATH 530
  100 CONTINUE                                                     PATH 540
      IF( REV ) 120,120,10                                         PATH 550
  110 WRITE(6,1000)                                                PATH 560
      STATUS=0                                                     PATH 570
  120 LEND=A1 - 1                                                  PATH 580
      RETURN                                                       PATH 590
 1000 FORMAT(1H1,35('*'),/,' DIAGNOSTIC - LONGER P ARRAY REQUIRED',  PATH 600
     1/,' ',35('*'))                                               PATH 610
      END                                                          PATH 620
```

```
      SUBROUTINE PATHP(OUTPUT)                                          PATH  10
C     THIS SUBROUTINE ITEMIZES ALL THE PATHS STARTING AT INPUT (DEFINED PATH  20
C     BY A CALL TO PATHS) AND ENDING AT THE OUTPUT NODE SPECIFIED.      PATH  30
      IMPLICIT INTEGER*4 (A-Z)                                          PATH  40
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  PATH  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             PATH  60
     2  PATH(300,50),NMOD,MPATH,PSIGN(300)                              PATH  70
      INTEGER*2 P,PP,PATH                                               PATH  80
C                                                                       PATH  90
      WRITE(6,1001) P(NEDGS1,1),OUTPUT                                  PATH 100
      WRITE(6,1002)                                                     PATH 110
      WRITE(6,1003)                                                     PATH 120
      WRITE(6,1002)                                                     PATH 130
      NPATH=0                                                           PATH 140
      DO 350 I=NEDGS1,LEND                                              PATH 150
      IF( (P(I,2).EQ.OUTPUT) ) GO TO 340                                PATH 160
      GO TO 350                                                         PATH 170
  340 CALL TRACE(I)                                                     PATH 180
      WRITE(6,1002)                                                     PATH 190
  350 CONTINUE                                                          PATH 200
      RETURN                                                            PATH 210
 1001 FORMAT('1',T35,'INVENTORY OF PATHS',//,T21,'THE FOLLOWING PATHS EXPATH 220
     1IST BETWEEN NODES',I3,' AND',I3,/)                                PATH 230
 1002 FORMAT(1H+,T16,62('_'))                                           PATH 240
 1003 FORMAT(' ',T16,'| ROW OF P |   TRACE  -->  NODES    |   TRACE  -->PATH 250
     1  MODULES  |')                                                   PATH 260
      END                                                               PATH 270
```

```
      SUBROUTINE TRACE(I)                                               TRAC  10
C     THIS SUBROUTINE PRINTS THE ELEMENTS ASSOCIATED WITH THE           TRAC  20
C     ITH ENTRY IN THE P-ARRAY.                                         TRAC  30
      IMPLICIT INTEGER*4 (A-Z)                                          TRAC  40
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  TRAC  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             TRAC  60
     2  PATH(300,50),NMOD,MPATH,PSIGN(300)                              TRAC  70
      INTEGER*2 P,PP,PATH                                               TRAC  80
      INTEGER*2 PI4(50)                                                 TRAC  90
C                                                                       TRAC 100
      J=I                                                               TRAC 110
      K=PPL                                                             TRAC 120
   10 PP(K)=P(J,2)                                                      TRAC 130
      PI4(K)=P(J,4)                                                     TRAC 140
      K=K - 1                                                           TRAC 150
      IF( P(J,3).EQ.J ) GO TO 20                                        TRAC 160
      J=P(J,3)                                                          TRAC 170
      GO TO 10                                                          TRAC 180
   20 PP(K)=P(J,1)                                                      TRAC 190
      FL=K                                                              TRAC 200
   30 NN=PPL - FL + 1                                                   TRAC 210
      IF( NN.EQ.0 ) GO TO 40                                            TRAC 220
      IF( NN.GT.4 ) NN=4                                                TRAC 230
      LL=FL + NN - 1                                                    TRAC 240
      WRITE(6,1000)( PP(J),J=FL,LL )                                    TRAC 250
      IF( FL.EQ.K ) GO TO 35                                            TRAC 260
      WRITE(6,1001) ( PI4(J),J=FL,LL )                                  TRAC 270
      GO TO 45                                                          TRAC 280
   35 FL1=FL+1                                                          TRAC 290
      WRITE(6,1002) I                                                   TRAC 300
      WRITE(6,1001) ( PI4(J),J=FL1,LL )                                 TRAC 310
   45 FL=LL+1                                                           TRAC 320
      GO TO 30                                                          TRAC 330
   40 NPATH=NPATH+1                                                     TRAC 340
      DO 50 J=1,NMOD                                                    TRAC 350
```

```
   50 PATH(NPATH,J) = 0                                                 TRAC 360
      K1 = K + 1                                                        TRAC 370
      DO 60 J=K1,PPL                                                    TRAC 380
      MM = PI4(J)                                                       TRAC 390
   60 PATH(NPATH,MM) = 1                                                TRAC 400
      RETURN                                                            TRAC 410
 1000 FORMAT(' ',T16,'|',T27,'|',T52,'|',T28,4(I3,'-->'))               TRAC 420
 1001 FORMAT('+',T77,'|',T53,4(I3,'-->'))                               TRAC 430
 1002 FORMAT('+',T20,I4)                                                TRAC 440
      END                                                               TRAC 450
```

```
      SUBROUTINE PATHC                                                  PATH  10
C     THIS SUBROUTINE FINDS ALL THE PATH COMBINATIONS IN                PATH  20
C     THEIR BINARY FORM.                                                PATH  30
      IMPLICIT INTEGER*4 (A-Z)                                          PATH  40
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  PATH  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             PATH  60
     2  PATH(300,50),NMOD,MPATH,PSIGN(300)                              PATH  70
      INTEGER*2 P,PP,PATH                                               PATH  80
      INTEGER*2 PTAIL(300)                                              PATH  90
C                                                                       PATH 100
      DO 10 I=1,NPATH                                                   PATH 110
      PTAIL(I) = I                                                      PATH 120
      PSIGN(I) = +1                                                     PATH 130
   10 CONTINUE                                                          PATH 140
      N1 = 1                                                            PATH 150
      N2 = NPATH                                                        PATH 160
   50 IF (PTAIL(N1) .EQ. NPATH) GO TO 100                               PATH 170
      K1 = PTAIL(N1) + 1                                                PATH 180
      DO 20 I=K1,NPATH                                                  PATH 190
      N2 = N2 + 1                                                       PATH 200
      IF (N2 .GT. PATHL) GO TO 200                                      PATH 210
      PTAIL(N2) = I                                                     PATH 220
      PSIGN(N2) = -PSIGN(N1)                                            PATH 230
      INDEX = 0                                                         PATH 240
      DO 30 J=1,NMOD                                                    PATH 250
      PATH(N2,J) = 0                                                    PATH 260
      IF(PATH(N1,J).EQ.1 .OR. PATH(I,J).EQ.1) GO TO 40                  PATH 270
      GO TO 30                                                          PATH 280
   40 PATH(N2,J) = 1                                                    PATH 290
      INDEX = INDEX + 1                                                 PATH 300
   30 CONTINUE                                                          PATH 310
      IF(I.EQ.NPATH) GO TO 20                                           PATH 320
      IF (INDEX .EQ. NMOD) N2 = N2-1                                    PATH 330
   20 CONTINUE                                                          PATH 340
  100 N1 = N1 + 1                                                       PATH 350
```

```
      IF (N1 .EQ. N2) GO TO 500                                         PATH 360
      GO TO 50                                                          PATH 370
  200 WRITE(6,1000)                                                     PATH 380
      STATUS = 0                                                        PATH 390
  500 MPATH = N2                                                        PATH 400
      RETURN                                                            PATH 410
 1000 FORMAT(1H1,39('*'),/,'DIAGONOSTIC-LONGER PATH ARRAY REQUIRED',/,  PATH 420
     1  ' ',39('*'))                                                    PATH 430
      END                                                               PATH 440
```

```
      SUBROUTINE MODULE                                                 MODU  10
C     THIS SUBROUTINE COMPUTES THE MODULE RELIABILITY AND               MODU  20
C     SENSITIVITY OF MODULE RELIABILITY TO THE MODULE REDUNDANCY.       MODU  30
      IMPLICIT INTEGER*4 (A-Z)                                          MODU  40
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  MODU  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             MODU  60
     2  PATH(300,50),NMOD,MPATH,PSIGN(300)                              MODU  70
      INTEGER*2 P,PP,PATH                                               MODU  80
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     MODU  90
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 MODU 100
      COMMON/OPTM2/CONST(10),COSTF(50),G(50),RGRADN(50),RIGRDN(50),     MODU 110
     1  RUNIT(50),WUNIT(50),CMCOST,OBJFN,RLBTY,WEIGHT                   MODU 120
      REAL*4 CONST,COSTF,G,RGRADN,RIGRDN,RUNIT,WUNIT,CMCOST,OBJFN,      MODU 130
     1  RLBTY,WEIGHT                                                    MODU 140
      REAL*4 DUMMY1,DUMMY2,DUMMY3                                       MODU 150
C                                                                       MODU 160
      DO 200 K=1,NMOD                                                   MODU 170
      IF(KODE(K) .EQ. 0) GO TO 100                                      MODU 180
      DUMMY1 = RUNIT(K) ** MUNIT(K)                                     MODU 190
      DUMMY3 = - MUNIT(K) * ALOG(RUNIT(K))                              MODU 200
      G(K) = 1.0                                                        MODU 210
      DUMMY2 = 1.0                                                      MODU 220
      NM = NUNIT(K) - MUNIT(K)                                          MODU 230
      IF(NM .EQ. 0) GO TO 20                                            MODU 240
      DO 10 I=1,NM                                                      MODU 250
      DUMMY2 = DUMMY2 * DUMMY3/I                                        MODU 260
      G(K) = G(K) + DUMMY2                                              MODU 270
   10 CONTINUE                                                          MODU 280
   20 G(K) = G(K) * DUMMY1                                              MODU 290
      RIGRDN(K) = DUMMY1 * DUMMY2 * DUMMY3/(NM+1)                       MODU 300
      GO TO 200                                                         MODU 310
  100 DUMMY1 = RUNIT(K)                                                 MODU 320
      DUMMY2 = DUMMY1**NUNIT(K)                                         MODU 330
      G(K) = DUMMY2                                                     MODU 340
      NM = NUNIT(K) - MUNIT(K)                                          MODU 350
```

```
      IF(NM .EQ. 0) GO TO 120                                           MODU 360
      DO 110 I=1,NM                                                     MODU 370
      DUMMY2 = DUMMY2 * ((1.-DUMMY1)/DUMMY1) * (NUNIT(K)-I+1)/I         MODU 380
      G(K) = G(K) + DUMMY2                                              MODU 390
  110 CONTINUE                                                          MODU 400
  120 RIGRDN(K) = DUMMY2 * (1.-DUMMY1) * MUNIT(K)/(NM+1)                MODU 410
  200 CONTINUE                                                          MODU 420
      RETURN                                                            MODU 430
      END                                                               MODU 440
```

```
      SUBROUTINE RELIAB                                                 RELI  10
C     THIS SUBROUTINE CALCULATES THE SYSTEM RELIABILITY FROM THE        RELI  20
C     MODULE RELIABILITIES.                                             RELI  30
      IMPLICIT INTEGER*4 (A-Z)                                          RELI  40
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  RELI  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             RELI  60
     2  PATH(300,50),NMOD,MPATH,PSIGN(300)                              RELI  70
      INTEGER*2 P,PP,PATH                                               RELI  80
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     RELI  90
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 RELI 100
      COMMON/OPTM2/CONST(10),COSTF(50),G(50),RGRADN(50),RIGRDN(50),     RELI 110
     1  RUNIT(50),WUNIT(50),CMCOST,OBJFN,RLBTY,WEIGHT                   RELI 120
      REAL*4 CONST,COSTF,G,RGRADN,RIGRDN,RUNIT,WUNIT,CMCOST,OBJFN,      RELI 130
     1  RLBTY,WEIGHT                                                    RELI 140
      REAL*4 GPATH                                                      RELI 150
C                                                                       RELI 160
      RLBTY = 0.0                                                       RELI 170
      DO 10 I=1,MPATH                                                   RELI 180
      GPATH = PSIGN(I)                                                  RELI 190
      DO 20 J=1,NMOD                                                    RELI 200
      IF (PATH(I,J) .EQ. 0) GO TO 20                                    RELI 210
      GPATH = GPATH * G(J)                                              RELI 220
   20 CONTINUE                                                          RELI 230
      RLBTY = RLBTY + GPATH                                             RELI 240
   10 CONTINUE                                                          RELI 250
      RETURN                                                            RELI 260
      END                                                               RELI 270
```

```
      SUBROUTINE SENSE                                                  SENS  10
C     THIS SUBROUTINE DETERMINES THE SENSITIVITY OF THE SYSTEM          SENS  20
C     RELIABILITY TO MODULE REDUNDANCIES.                               SENS  30
      IMPLICIT INTEGER*4 (A-Z)                                          SENS  40
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  SENS  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             SENS  60
     2  PATH(300,50),NMOD,MPATH,PSIGN(300)                              SENS  70
      INTEGER*2 P,PP,PATH                                               SENS  80
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     SENS  90
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 SENS 100
      COMMON/OPTM2/CONST(10),COSTF(50),G(50),RGRADN(50),RIGRDN(50),     SENS 110
     1  RUNIT(50),WUNIT(50),CMCOST,OBJFN,RLBTY,WEIGHT                   SENS 120
      REAL*4 CONST,COSTF,G,RGRADN,RIGRDN,RUNIT,WUNIT,CMCOST,OBJFN,      SENS 130
     1  RLBTY,WEIGHT                                                    SENS 140
      REAL*4 DUMMY(50)                                                  SENS 150
C                                                                       SENS 160
      DO 10 I=1,NMOD                                                    SENS 170
   10 DUMMY(I) = G(I)                                                   SENS 180
      DO 100 I=1,NMOD                                                   SENS 190
      DO 90 J=1,NMOD                                                    SENS 200
   90 G(J) = DUMMY(J)                                                   SENS 210
      G(I) = 1.0                                                        SENS 220
      CALL RELIAB                                                       SENS 230
      RGRADN(I) = RLBTY                                                 SENS 240
      G(I) = 0.0                                                        SENS 250
      CALL RELIAB                                                       SENS 260
      RGRADN(I) = (RGRADN(I)-RLBTY) * RIGRDN(I)                         SENS 270
  100 CONTINUE                                                          SENS 280
      DO 150 I=1,NMOD                                                   SENS 290
  150 G(I) = DUMMY(I)                                                   SENS 300
      RETURN                                                            SENS 310
      END                                                               SENS 320
```

```
      SUBROUTINE CONSTR                                                 CONS  10
C     THIS SUBROUTINE COMPUTES THE SYSTEM CONSTRAINTS.                  CONS  20
      IMPLICIT INTEGER*4 (A-Z)                                          CONS  30
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  CONS  40
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             CONS  50
     2  PATH(300,50),NMOD,MPATH,PSIGN(300)                              CONS  60
      INTEGER*2 P,PP,PATH                                               CONS  70
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     CONS  80
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 CONS  90
      COMMON/OPTM2/CONST(10),COSTF(50),G(50),RGRADN(50),RIGRDN(50),     CONS 100
     1  RUNIT(50),WUNIT(50),CMCOST,OBJFN,RLBTY,WEIGHT                   CONS 110
      REAL*4 CONST,COSTF,G,RGRADN,RIGRDN,RUNIT,WUNIT,CMCOST,OBJFN,      CONS 120
     1  RLBTY,WEIGHT                                                    CONS 130
C                                                                       CONS 140
      CONST(1) = 0.0                                                    CONS 150
      CONST(2) = 0.0                                                    CONS 160
      DO 10 I=1,NMOD                                                    CONS 170
      CONST(1) = CONST(1) + NUNIT(I)*COSTF(I)                           CONS 180
      CONST(2) = CONST(2) + NUNIT(I)*WUNIT(I)                           CONS 190
   10 CONTINUE                                                          CONS 200
      CONST(1) = CONST(1) - CMCOST                                      CONS 210
      CONST(2) = CONST(2) - WEIGHT                                      CONS 220
      RETURN                                                            CONS 230
      END                                                               CONS 240
```

```
      SUBROUTINE FESIBL                                                 FESI  10
C     THIS SUBROUTINE FINDS THE FEASIBLE SOLUTION BY THE METHOD         FESI  20
C     OF WEIGHTED PERPENDICULARS.                                       FESI  30
      IMPLICIT INTEGER*4 (A-Z)                                          FESI  40
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  FESI  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             FESI  60
     2  PATH(300,50),NMOD,MPATH,PSIGN(300)                              FESI  70
      INTEGER*2 P,PP,PATH                                               FESI  80
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     FESI  90
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 FESI 100
      COMMON/OPTM2/CONST(10),COSTF(50),G(50),RGRADN(50),RIGRDN(50),     FESI 110
     1  RUNIT(50),WUNIT(50),CMCOST,OBJFN,RLBTY,WEIGHT                   FESI 120
      REAL*4 CONST,COSTF,G,RGRADN,RIGRDN,RUNIT,WUNIT,CMCOST,OBJFN,      FESI 130
     1  RLBTY,WEIGHT                                                    FESI 140
      REAL*4 Z(50),ZMIN,S,DUMMY1,ZMAX                                   FESI 150
C                                                                       FESI 160
      MOVE = 0                                                          FESI 170
      INDEX = 0                                                         FESI 180
    5 CALL CONSTR                                                       FESI 190
      WRITE(6,1400) (NUNIT(I),I=1,NMOD  )                               FESI 200
      WRITE(6,1600)  CONST(1),CONST(2)                                  FESI 210
      IF(CONST(1) .LE. 0.0  .AND.  CONST(2) .LE. 0.0) GO TO 1000        FESI 220
      IF(CONST(1) .LE. 0.0) GO TO 30                                    FESI 230
      S = CONST(1)                                                      FESI 240
      DUMMY1 = 0.0                                                      FESI 250
      DO 10 I=1,NMOD                                                    FESI 260
   10 DUMMY1 = DUMMY1 + NUNIT(I)*COSTF(I)                               FESI 270
      DO 20 I=1,NMOD                                                    FESI 280
   20 Z(I) = -CONST(1) * COSTF(I)                                       FESI 290
   30 IF(CONST(2) .LE. 0.0) GO TO 50                                    FESI 300
      S = S + CONST(2)                                                  FESI 310
      DO 40 I=1,NMOD                                                    FESI 320
   40 Z(I) = Z(I) - CONST(2) * WUNIT(I)                                 FESI 330
   50 DO 60 I=1,NMOD                                                    FESI 340
   60 Z(I) = +Z(I)/S                                                    FESI 350
```

```
      ZMAX = ABS(Z(1))                                                  FESI 360
      DO 70 J=2,NMOD                                                    FESI 370
      IF (ZMAX .LT. ABS(Z(J))) ZMAX = ABS(Z(J))                         FESI 380
   70 CONTINUE                                                          FESI 390
      DO 80 I=1,NMOD                                                    FESI 400
      IF(Z(I)) 90,80,100                                                FESI 410
   90 Z(I) = (Z(I)/ZMAX) * 2. - 0.5                                     FESI 420
      Z(I) = AINT(Z(I))                                                 FESI 430
      GO TO 80                                                          FESI 440
  100 Z(I) = (Z(I)/ZMAX) * 2. + 0.5                                     FESI 450
      Z(I) = AINT(Z(I))                                                 FESI 460
   80 CONTINUE                                                          FESI 470
      INDEX = 0                                                         FESI 480
      MOVE = MOVE + 1                                                   FESI 490
      DO 120 I=1,NMOD                                                   FESI 500
      NUNIT(I) = NUNIT(I) + Z(I)                                        FESI 510
      IF(NUNIT(I).LT. NMIN(I) .OR. NUNIT(I).GT.NMAX(I)) INDEX=INDEX+1   FESI 520
      IF(NUNIT(I) .LT.  NMIN(I)) NUNIT(I) =  NMIN(I)                    FESI 530
      IF(NUNIT(I) .GT. NMAX(I)) NUNIT(I) = NMAX(I)                      FESI 540
  120 CONTINUE                                                          FESI 550
      IF (MOVE .GT. 20  .OR.  INDEX .EQ. NMOD  )  GO TO 999             FESI 560
      GO TO 5                                                           FESI 570
  999 WRITE(6,2500)                                                     FESI 580
 1000 RETURN                                                            FESI 590
 1400 FORMAT(1H0,/,10X,'NUNIT(I) =',20I5)                               FESI 600
 1600 FORMAT(1H0,/,30X,'CONSTRAINT(1) =',G13.6,//,30X,'CONSTRAINT(2) =',FESI 610
     1  G13.6)                                                          FESI 620
 2500 FORMAT(1H0,////,30X,'FESIBL POINT CAN NOT BE LOCATED',////)       FESI 630
      END                                                               FESI 640
```

```
      SUBROUTINE SORT                                                   SORT  10
C     THIS SUBROUTINE ARRANGES THE GRADIENT VECTOR IN ASCENDING ORDER.  SORT  20
      IMPLICIT INTEGER*4 (A-Z)                                          SORT  30
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  SORT  40
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             SORT  50
     2  PATH(300,50),NMOD,MPATH,PSIGN(300)                              SORT  60
      INTEGER*2 P,PP,PATH                                               SORT  70
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     SORT  80
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 SORT  90
      COMMON/OPTM2/CONST(10),COSTF(50),G(50),RGRADN(50),RIGRDN(50),     SORT 100
     1  RUNIT(50),WUNIT(50),CMCOST,OBJFN,RLBTY,WEIGHT                   SORT 110
      REAL*4 CONST,COSTF,G,RGRADN,RIGRDN,RUNIT,WUNIT,CMCOST,OBJFN,      SORT 120
     1  RLBTY,WEIGHT                                                    SORT 130
      REAL*4 GRDMIN,DUMMY(50)                                           SORT 140
C                                                                       SORT 150
      DO 5 I=1,NMOD                                                     SORT 160
      DUMMY(I) = RGRADN(I)                                              SORT 170
    5 CONTINUE                                                          SORT 180
      NONZRO = 0                                                        SORT 190
      DO 10 I=1,NMOD                                                    SORT 200
      IF(ABS(DUMMY(I)) .LT. 1.E-15) GO TO 10                            SORT 210
      NONZRO = NONZRO + 1                                               SORT 220
   10 CONTINUE                                                          SORT 230
      DO 20 I=1,NONZRO                                                  SORT 240
      GRDMIN = 1.E30                                                    SORT 250
      DO 30 J=1,NMOD                                                    SORT 260
      IF(ABS(DUMMY(J)) .LT. 1.E-15) GO TO 30                            SORT 270
      I1 = I - 1                                                        SORT 280
      IF(I1 .EQ.0) GO TO 50                                             SORT 290
      DO 40 K=1,I1                                                      SORT 300
      IF(J .EQ. NSORT(K)) GO TO 30                                      SORT 310
   40 CONTINUE                                                          SORT 320
   50 IF(GRDMIN .LE. ABS(DUMMY (J))) GO TO 30                           SORT 330
      GRDMIN = ABS(DUMMY (J))                                           SORT 340
      NSORT(I) = J                                                      SORT 350
```

```
36:      30 CONTINUE
37:      20 CONTINUE                                          SORT 360
38:         RETURN                                            SORT 370
39:         END                                               SORT 380
                                                              SORT 390
```

```
      SUBROUTINE OPTIM                                                  OPTI  10
C     THIS SUBROUTINE CHECKS FOR AN IMPROVED FEASIBLE SOLUTION.         OPTI  20
      IMPLICIT INTEGER*4 (A-Z)                                          OPTI  30
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  OPTI  40
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             OPTI  50
     2  PATH(300,50),NMOD,MPATH,PSIGN(300)                              OPTI  60
      INTEGER*2 P,PP,PATH                                               OPTI  70
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     OPTI  80
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 OPTI  90
      COMMON/OPTM2/CONST(10),COSTF(50),G(50),RGRADN(50),RIGRDN(50),     OPTI 100
     1  RUNIT(50),WUNIT(50),CMCOST,OBJFN,RLBTY,WEIGHT                   OPTI 110
      REAL*4 CONST,COSTF,G,RGRADN,RIGRDN,RUNIT,WUNIT,CMCOST,OBJFN,      OPTI 120
     1  RLBTY,WEIGHT                                                    OPTI 130
      DIMENSION NUNIT1(50)                                              OPTI 140
C                                                                       OPTI 150
      MOVE = 0                                                          OPTI 160
    1 DO 10 I = 1,NMOD                                                  OPTI 170
      NUNIT1(I) = NUNIT(I)                                              OPTI 180
   10 NUNIT(I) = NUNIT(I) + RGRADN(I)                                   OPTI 190
      DO 20 I=1,NMOD                                                    OPTI 200
      IF(NUNIT(I).LT. NMIN(I) .OR. NUNIT(I).GT.NMAX(I)) GO TO 100       OPTI 210
   20 CONTINUE                                                          OPTI 220
      CALL CONSTR                                                       OPTI 230
      IF(CONST(1).GT. 0.0 .OR. CONST(2).GT. 0.0)  GO TO 100             OPTI 240
      CALL MODULE                                                       OPTI 250
      CALL RELIAB                                                       OPTI 260
      IF (RLBTY .LE. OBJFN) GO TO 100                                   OPTI 270
      MOVE = MOVE + 1                                                   OPTI 280
      OBJFN = RLBTY                                                     OPTI 290
      GO TO 999                                                         OPTI 300
  100 DO 30 I=1,NMOD                                                    OPTI 310
   30 NUNIT(I) = NUNIT1(I)                                              OPTI 320
  999 RETURN                                                            OPTI 330
      END                                                               OPTI 340
```

```
      SUBROUTINE OPTIMZ                                                 OPTI  10
C     THIS SUBROUTINE FINDS A LOCALLY OPTIMAL SOLUTION.                 OPTI  20
      IMPLICIT INTEGER*4 (A-Z)                                          OPTI  30
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  OPTI  40
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             OPTI  50
     2  PATH(300,50),NMOD,MPATH,PSIGN(300)                              OPTI  60
      INTEGER*2 P,PP,PATH                                               OPTI  70
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     OPTI  80
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 OPTI  90
      COMMON/OPTM2/CONST(10),COSTF(50),G(50),RGRADN(50),RIGRDN(50),     OPTI 100
     1  RUNIT(50),WUNIT(50),CMCOST,OBJFN,RLBTY,WEIGHT                   OPTI 110
      REAL*4 CONST,COSTF,G,RGRADN,RIGRDN,RUNIT,WUNIT,CMCOST,OBJFN,      OPTI 120
     1  RLBTY,WEIGHT                                                    OPTI 130
      REAL*4 DUMMY(50)                                                  OPTI 140
C                                                                       OPTI 150
      CALL MODULE                                                       OPTI 160
      CALL RELIAB                                                       OPTI 170
      OBJFN = RLBTY                                                     OPTI 180
      CALL SENSE                                                        OPTI 190
      CALL SORT                                                         OPTI 200
      CALL CONSTR                                                       OPTI 210
      WRITE(6,1400) (NUNIT(I),I=1,NMOD  )                               OPTI 220
      WRITE(6,1500) OBJFN                                               OPTI 230
      WRITE(6,1600)  CONST(1),CONST(2)                                  OPTI 240
      DO 5 I=1,NMOD                                                     OPTI 250
    5 DUMMY(I) = RGRADN(I)                                              OPTI 260
      DO 200 I=1,NONZRO                                                 OPTI 270
      DO 110 J=1,NMOD                                                   OPTI 280
  110 RGRADN(J) = 0.0                                                   OPTI 290
      I1 = NONZRO - I + 1                                               OPTI 300
      K = NSORT(I1)                                                     OPTI 310
      RGRADN(K) = DUMMY(K)/ABS(DUMMY(K))                                OPTI 320
      CALL OPTIM                                                        OPTI 330
      IF(MOVE .EQ. 0) GO TO 200                                         OPTI 340
      GO TO 1000                                                        OPTI 350
```

```
36:    200 CONTINUE                                                          OPTI 360
37:   1000 RETURN                                                            OPTI 370
38:   1400 FORMAT(1H0,/,10X,'NUNIT(I) =',20I5)                               OPTI 380
39:   1500 FORMAT (1H0,//,30X,'SYSTEM RELIABILITY =',G13.6)                  OPTI 390
40:   1600 FORMAT(1H0,/,30X,'CONSTRAINT(1) =',G13.6,//,30X,'CONSTRAINT(2) =',OPTI 400
41:       1  G13.6)                                                          OPTI 410
42:        END                                                               OPTI 420
```

SYSTEM AVAILABILITY OPTIMIZATION

```
C     MAIN PROGRAM                                                      MAIN  10
C     SYSTEM AVAILABILITY OPTIMIZATION                                  MAIN  20
C     MODIFIED INTEGER GRADIENT METHOD                                  MAIN  30
      IMPLICIT INTEGER*4 (A-Z)                                          MAIN  40
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  MAIN  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             MAIN  60
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        MAIN  70
      INTEGER*2 P,PP,PATH                                               MAIN  80
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     MAIN  90
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 MAIN 100
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      MAIN 110
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   MAIN 120
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      MAIN 130
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       MAIN 140
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   MAIN 150
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,MAIN 160
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTMAIN 170
      REAL*4 YFL,FCOST,MCOST                                            MAIN 180
C                                                                       MAIN 190
      PATHL = 300                                                       MAIN 200
      PLENTH = 1000                                                     MAIN 210
      PPL = 50                                                          MAIN 220
   10 CALL DATAIN                                                       MAIN 230
      READ(5,1000) INNOD,OUTNOD                                         MAIN 240
      CALL PATHS(INNOD)                                                 MAIN 250
      IF(STATUS) 20,10,20                                               MAIN 260
   20 CALL PATHP(OUTNOD)                                                MAIN 270
      WRITE(6,1100)                                                     MAIN 280
      DO 30 L=1,NPATH                                                   MAIN 290
      WRITE(6,1200) L,(PATH(L,M),M=1,NMOD  )                            MAIN 300
   30 CONTINUE                                                          MAIN 310
      CALL PATHC                                                        MAIN 320
      WRITE(6,1001) NPATH,MPATH                                         MAIN 330
      WRITE(6,2600)                                                     MAIN 340
      DO 35 L=1,MPATH                                                   MAIN 350
```

```
      WRITE(6,2700) L,PSIGN(L),(PATH(L,M),M=1,NMOD)                     MAIN 360
   35 CONTINUE                                                          MAIN 370
      KOUNT = 1                                                         MAIN 380
      ISEED = 984376521                                                 MAIN 390
      IX = ISEED                                                        MAIN 400
      DO 45 I=1,NMOD                                                    MAIN 410
   45 NUNIT(I) = 1                                                      MAIN 420
      TM = TMIN + (TMAX-TMIN)/2.                                        MAIN 430
      GO TO 55                                                          MAIN 440
C     GENERATING A UNIFORMLY DISTRIBUTED RANDOM STARTING POINT          MAIN 450
   40 DO 50 I=1,NMOD                                                    MAIN 460
      CALL RANDU(IX,IY,YFL)                                             MAIN 470
      NUNIT(I) = YFL*(NMAX(I)- NMIN(I)+1) + NMIN(I)                     MAIN 480
      IX = IY                                                           MAIN 490
   50 CONTINUE                                                          MAIN 500
      CALL RANDU(IX,IY,YFL)                                             MAIN 510
      TM = YFL*(TMAX-TMIN) + TMIN                                       MAIN 520
      IX = IY                                                           MAIN 530
   55 WRITE (6,1300) KOUNT                                              MAIN 540
      WRITE(6,2100)                                                     MAIN 550
      WRITE(6,1400) (NUNIT(I),I=1,NMOD  )                               MAIN 560
      WRITE(6,1450) TM                                                  MAIN 570
C     LOCATING A FEASIBLE POINT                                         MAIN 580
      WRITE(6,2200)                                                     MAIN 590
      CALL FESIBL                                                       MAIN 600
      WRITE(6,1410) MOVE,INDEX                                          MAIN 610
      IF (MOVE .GT. 20  .OR.  INDEX .EQ. NMOD1 )  GO TO 90              MAIN 620
      GO TO 60                                                          MAIN 630
   60 WRITE(6,2400)                                                     MAIN 640
      WRITE(6,1400) (NUNIT(I),I=1,NMOD  )                               MAIN 650
      WRITE(6,1450) TM                                                  MAIN 660
      WRITE(6,1600)  CONST(1),CONST(2)                                  MAIN 670
C     SEARCHING FOR LOCAL OPTIMUM                                       MAIN 680
      WRITE(6,2300)                                                     MAIN 690
   65 CALL OPTIMZ                                                       MAIN 700
      IF(MOVE .EQ. 0) GO TO 70                                          MAIN 710
```

```
      GO TO 65                                                          MAIN 720
   70 WRITE(6,2500)                                                     MAIN 730
      WRITE(6,1400) (NUNIT(I),I=1,NMOD  )                               MAIN 740
      WRITE(6,1450) TM                                                  MAIN 750
      FLAG = 1                                                          MAIN 760
      CALL AVLBTY                                                       MAIN 770
      CALL CONSTR                                                       MAIN 780
      FCOST = 0.0                                                       MAIN 790
      DO 80 I=1,NMOD                                                    MAIN 800
   80 FCOST = FCOST + NUNIT(I)*COSTF(I)                                 MAIN 810
      MCOST = CONST(1) + CMCOST - FCOST                                 MAIN 820
      WRITE(6,1500) ASYS,PSYS,RSYS                                      MAIN 830
      WRITE(6,1600)  CONST(1),CONST(2)                                  MAIN 840
      WRITE(6,1700) FCOST,MCOST                                         MAIN 850
      WRITE(6,1900) MTBM,MDT                                            MAIN 860
   90 KOUNT = KOUNT + 1                                                 MAIN 870
      IF (KOUNT .GT. 25) GO TO 10                                       MAIN 880
      GO TO 40                                                          MAIN 890
 1000 FORMAT (2I3)                                                      MAIN 900
 1001 FORMAT(1H0,//,30X,'NPATH = ',I3,//,30X,'MPATH = ',I5)             MAIN 910
 1100 FORMAT (1H0,//,T20,'PATHS IN BINARY CODE',/)                      MAIN 920
 1200 FORMAT (1H0,T16,'PATH',I3,' =  ',50I2)                            MAIN 930
 1300 FORMAT(1H1,40X,'KOUNT =',I3,//)                                   MAIN 940
 1400 FORMAT(1H0,/,10X,'NUNIT(I) =',20I5)                               MAIN 950
 1410 FORMAT (1H0,//,30X,'MOVE =',I4,10X,'INDEX =',I4)                  MAIN 960
 1450 FORMAT (1H0,10X,'TM =',G13.5)                                     MAIN 970
 1500 FORMAT (1H0,//,30X,'SYSTEM AVAILABILITY =',G13.6,//,30X,          MAIN 980
     1  'SYSTEM PROFIT =',G13.6,//,30X,'SYSTEM RELIABILITY =',G13.6)    MAIN 990
 1600 FORMAT(1H0,/,30X,'CONSTRAINT(1) =',G13.6,//,30X,'CONSTRAINT(2) =',MAIN1000
     1  G13.6)                                                          MAIN1010
 1700 FORMAT(1H0,/,30X,'FCOST =',G13.6,//,30X,'MCOST =',G13.6)          MAIN1020
 1900 FORMAT(1H0,20X,'MTBM =',G13.6,20X,'MDT =',G13.6)                  MAIN1030
 2100 FORMAT(1H0,50X,'THE INITIAL POINT IS')                            MAIN1040
 2200 FORMAT(1H0,50X,'SEARCHING FOR THE FEASIBLE POINT')                MAIN1050
 2300 FORMAT(1H0,50X,'SEARCHING FOR THE LOCAL OPTIMUM POINT')           MAIN1060
 2400 FORMAT(1H0,////,50X,'THE FEASIBLE POINT IS')                      MAIN1070
```

```
 2500 FORMAT(1H0,////,50X,'THE OPTIMAL POINT IS')                          MAIN1080
 2600 FORMAT(1H0,//,T20,'PATHS AND PATH COMBINATIONS IN BINARY CODE',/) MAIN1090
 2700 FORMAT(1H0,T16,'PATHC',I3,' = ',I2,4X,50I2)                          MAIN1100
      END                                                                  MAIN1110
```

```
      SUBROUTINE DATAIN                                                 DATA  10
C     THIS SUBROUTINE READS AND ECHOCHECKS THE DATA                     DATA  20
      IMPLICIT INTEGER*4 (A-Z)                                          DATA  30
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  DATA  40
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             DATA  50
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        DATA  60
      INTEGER*2 P,PP,PATH                                               DATA  70
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     DATA  80
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 DATA  90
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      DATA 100
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   DATA 110
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      DATA 120
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       DATA 130
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   DATA 140
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,DATA 150
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTDATA 160
      DIMENSION TITLE(30)                                               DATA 170
C                                                                       DATA 180
      READ(5,999,END=100)( TITLE(I),I=1,30)                             DATA 190
      READ(5,1000) NNODES,NMOD,CMCOST,WEIGHT,INCOM,COSTMF,TLF,TEM,      DATA 200
     1  TSC,TMIN,TMAX,TSTEP,INTVL                                       DATA 210
      DO 10 I=1,NMOD                                                    DATA 220
   10 READ(5,1007) KODE(I),NMAX(I),NMIN(I),MUNIT(I),COSTF(I),WUNIT(I),  DATA 230
     1  LAMDA(I)                                                        DATA 240
      NMOD1 = NMOD + 1                                                  DATA 250
      FOR=0                                                             DATA 260
      REV=0                                                             DATA 270
      DO 40 I=1,1000                                                    DATA 280
      READ(5,1001)P(I,1),P(I,2),P(I,4)                                  DATA 290
      IF( P(I,1)-P(I,2) ) 20,50,30                                      DATA 300
   20 FOR=FOR + 1                                                       DATA 310
      GO TO 40                                                          DATA 320
   30 REV=REV + 1                                                       DATA 330
   40 CONTINUE                                                          DATA 340
   50 NEDGES=FOR + REV                                                  DATA 350
```

```
      NEDGS1=NEDGES + 1                                                 DATA 360
      FREV=FOR + 1                                                      DATA 370
      WRITE(6,1002) ( TITLE(I),I=1,30)                                  DATA 380
      WRITE(6,1003)                                                     DATA 390
      WRITE(6,1004)                                                     DATA 400
      WRITE(6,1003)                                                     DATA 410
      DO 60 I=1,NEDGES                                                  DATA 420
      WRITE(6,1005) I,P(I,1),P(I,2),P(I,4)                              DATA 430
      WRITE(6,1003)                                                     DATA 440
   60 CONTINUE                                                          DATA 450
      WRITE(6,1008)                                                     DATA 460
      WRITE(6,1011)                                                     DATA 470
      WRITE(6,1009)                                                     DATA 480
      WRITE(6,1011)                                                     DATA 490
      DO 70 I=1,NMOD                                                    DATA 500
      WRITE(6,1010) I,KODE(I),NMAX(I),NMIN(I),MUNIT(I),COSTF(I),        DATA 510
     1  WUNIT(I),LAMDA(I)                                               DATA 520
      WRITE(6,1011)                                                     DATA 530
   70 CONTINUE                                                          DATA 540
      WRITE(6,1006) CMCOST,WEIGHT,INCOM,COSTMF,TLF,TEM,TSC,TMIN,TMAX,   DATA 550
     1  TSTEP,INTVL                                                     DATA 560
  999 FORMAT(15A4)                                                      DATA 570
 1000 FORMAT (2I10,5F10.0/6F10.0)                                       DATA 580
 1001 FORMAT (3I10)                                                     DATA 590
 1002 FORMAT(1H1,//,T10,30A4,//,T50,'INVENTORY OF BRANCHES',/)          DATA 600
 1003 FORMAT (1H+,T38,44('_'))                                          DATA 610
 1004 FORMAT (1H ,T38,'|  BRANCH  |','  ORIGIN  |','   END    |',       DATA 620
     1  '  MODULE  |')                                                 DATA 630
 1005 FORMAT (1H ,T38,4('|',4X,I2,4X),'|')                              DATA 640
 1006 FORMAT (1H0,T10,'CMCOST=',G13.6,/,T10,'WEIGHT=',G13.6,/,T10,      DATA 650
     1  'INCOM =',G13.6,/,T10,'COSTMF=',G13.6,/,T10,'TLF   =',G13.6,/,  DATA 660
     2  T10,'TEM   =',G13.6,/,T10,'TSC   =',G13.6,/,T10,'TMIN  =',      DATA 670
     3  G13.6,/,T10,'TMAX  =',G13.6,/,T10,'TSTEP =',G13.6,/,T10,        DATA 680
     4  'INTVL =',G13.6)                                                DATA 690
 1007 FORMAT(4I10,2F10.0,E20.0)                                         DATA 700
 1008 FORMAT(1H0,T50,'INVENTORY OF MODULES',/)                          DATA 710
```

```
 1009 FORMAT(1H ,T20,'|  MODULE  |','   KODE   |',' MAX-UNIT |',        DATA 720
     1  ' MIN-UNIT |','M OUT OF N|','UNIT COSTF|','UNIT WEGHT|',        DATA 730
     1  ' FAILURE RATE |')                                               DATA 740
 1010 FORMAT(1H ,T20,5('|',4X,I2,4X),2('|',F10.3),'|',G14.5,'|')         DATA 750
 1011 FORMAT(1H+,T20,92('_'))                                            DATA 760
      RETURN                                                             DATA 770
  100 CALL EXIT                                                          DATA 780
      END                                                                DATA 790
```

```
      SUBROUTINE MODULE                                                 MODU  10
C     THIS SUBROUTINE COMPUTES THE MODULE RELIABILITY AND               MODU  20
C     SENSITIVITY OF MODULE RELIABILITY TO THE MODULE REDUNDANCY.       MODU  30
      IMPLICIT INTEGER*4 (A-Z)                                          MODU  40
       COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                 MODU  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             MODU  60
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        MODU  70
      INTEGER*2 P,PP,PATH                                               MODU  80
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     MODU  90
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 MODU 100
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      MODU 110
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   MODU 120
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      MODU 130
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       MODU 140
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   MODU 150
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,MODU 160
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTMODU 170
      REAL*4 DUMMY1,DUMMY2,DUMMY3                                       MODU 180
C                                                                       MODU 190
      DO 200 K=1,NMOD                                                   MODU 200
      IF(KODE(K) .EQ. 0) GO TO 100                                      MODU 210
      DUMMY1 = EXP(-LAMDA(K)*T*MUNIT(K))                                MODU 220
      DUMMY2 = 1.0                                                      MODU 230
      DUMMY3 = LAMDA(K) * T * MUNIT(K)                                  MODU 240
      G(K) = 1.0                                                        MODU 250
      NM = NUNIT(K) - MUNIT(K)                                          MODU 260
      IF(NM .EQ. 0) GO TO 20                                            MODU 270
      DO 10 I=1,NM                                                      MODU 280
      DUMMY2 = DUMMY2 * DUMMY3/I                                        MODU 290
      G(K) = G(K) + DUMMY2                                              MODU 300
   10 CONTINUE                                                          MODU 310
   20 G(K) = G(K) * DUMMY1                                              MODU 320
      RIGRDN(K) = DUMMY1 * DUMMY2 * DUMMY3/(NM+1)                       MODU 330
      GO TO 200                                                         MODU 340
  100 DUMMY1 = EXP(-LAMDA(K)*T)                                         MODU 350
```

```
      DUMMY2 = DUMMY1**NUNIT(K)                                         MODU 360
      G(K) = DUMMY2                                                     MODU 370
      NM = NUNIT(K) - MUNIT(K)                                          MODU 380
      IF(NM .EQ. 0) GO TO 120                                           MODU 390
      DO 110 I=1,NM                                                     MODU 400
      DUMMY2 = DUMMY2 * ((1.-DUMMY1)/DUMMY1) * (NUNIT(K)-I+1)/I         MODU 410
      G(K) = G(K) + DUMMY2                                              MODU 420
  110 CONTINUE                                                          MODU 430
  120 RIGRDN(K) = DUMMY2 * (1.-DUMMY1) * MUNIT(K)/(NM+1)                MODU 440
  200 CONTINUE                                                          MODU 450
      RETURN                                                            MODU 460
      END                                                               MODU 470
```

```
      SUBROUTINE RELIAB                                                 RELI  10
C     THIS SUBROUTINE CALCULATES THE SYSTEM RELIABILITY FROM THE        RELI  20
C     MODULE RELIABILITIES.                                             RELI  30
      IMPLICIT INTEGER*4 (A-Z)                                          RELI  40
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  RELI  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             RELI  60
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        RELI  70
      INTEGER*2 P,PP,PATH                                               RELI  80
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     RELI  90
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 RELI 100
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      RELI 110
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   RELI 120
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      RELI 130
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       RELI 140
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   RELI 150
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,RELI 160
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTRELI 170
      REAL*4 GPATH                                                      RELI 180
C                                                                       RELI 190
      RLBTY = 0.0                                                       RELI 200
      DO 10 I=1,MPATH                                                   RELI 210
      GPATH = PSIGN(I)                                                  RELI 220
      DO 20 J=1,NMOD                                                    RELI 230
      IF (PATH(I,J) .EQ. 0) GO TO 20                                    RELI 240
      GPATH = GPATH * G(J)                                              RELI 250
   20 CONTINUE                                                          RELI 260
      RLBTY = RLBTY + GPATH                                             RELI 270
   10 CONTINUE                                                          RELI 280
      RETURN                                                            RELI 290
      END                                                               RELI 300
```

```
      SUBROUTINE SENSE                                                  SENS  10
C     THIS SUBROUTINE DETERMINES THE SENSITIVITY OF THE SYSTEM          SENS  20
C     RELIABILITY TO MODULE REDUNDANCIES.                               SENS  30
      IMPLICIT INTEGER*4 (A-Z)                                          SENS  40
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  SENS  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             SENS  60
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        SENS  70
      INTEGER*2 P,PP,PATH                                               SENS  80
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     SENS  90
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 SENS 100
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      SENS 110
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   SENS 120
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      SENS 130
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       SENS 140
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   SENS 150
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,SENS 160
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTSENS 170
      REAL*4 DUMMY(50)                                                  SENS 180
C                                                                       SENS 190
      DO 10 I=1,NMOD                                                    SENS 200
   10 DUMMY(I) = G(I)                                                   SENS 210
      DO 100 I=1,NMOD                                                   SENS 220
      DO 90 J=1,NMOD                                                    SENS 230
   90 G(J) = DUMMY(J)                                                   SENS 240
      G(I) = 1.0                                                        SENS 250
      CALL RELIAB                                                       SENS 260
      RGRADN(I) = RLBTY                                                 SENS 270
      G(I) = 0.0                                                        SENS 280
      CALL RELIAB                                                       SENS 290
      RGRADN(I) = (RGRADN(I)-RLBTY) * RIGRDN(I)                         SENS 300
  100 CONTINUE                                                          SENS 310
      DO 150 I=1,NMOD                                                   SENS 320
  150 G(I) = DUMMY(I)                                                   SENS 330
      RETURN                                                            SENS 340
      END                                                               SENS 350
```

```
      SUBROUTINE AVLBTY                                                 AVLB  10
C     THIS SUBROUTINE DETERMINES SYSTEM AVAILABILITY, SYSTEM PROFIT,    AVLB  20
C     AND SENSITIVITY OF SYSTEM AVAILABILITY AND SYSTEM PROFIT TO EACH  AVLB  30
C     MODULE'S REDUNDANCY AND TO SYSTEM MAINTENANCE INTERVAL.           AVLB  40
      IMPLICIT INTEGER*4 (A-Z)                                          AVLB  50
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  AVLB  60
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             AVLB  70
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        AVLB  80
      INTEGER*2 P,PP,PATH                                               AVLB  90
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     AVLB 100
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 AVLB 110
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      AVLB 120
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   AVLB 130
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      AVLB 140
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       AVLB 150
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   AVLB 160
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,AVLB 170
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTAVLB 180
      REAL*4 DELTA,DUMMY1                                               AVLB 190
C                                                                       AVLB 200
      EVEN = IFIX((TM/INTVL) + 1.0)                                     AVLB 210
      EVEN = (EVEN/2) * 2                                               AVLB 220
      IF (EVEN .EQ. 0) EVEN = 2                                         AVLB 230
      DELTA = TM/EVEN                                                   AVLB 240
      MTBM = 1.0                                                        AVLB 250
      DO 10 J=1,NMOD                                                    AVLB 260
   10 MTGRDN(J) = 0.0                                                   AVLB 270
      N = EVEN/2                                                        AVLB 280
      N1 = N-1                                                          AVLB 290
      DO 50 I=1,N                                                       AVLB 300
      T = DELTA * (2*I-1)                                               AVLB 310
      CALL MODULE                                                       AVLB 320
      CALL RELIAB                                                       AVLB 330
      RSYS = RLBTY                                                      AVLB 340
      MTBM = MTBM + 4.0 * RSYS                                          AVLB 350
```

```
      IF(FLAG .EQ. 1) GO TO 50                                          AVLB 360
      CALL SENSE                                                        AVLB 370
      DO 30 J=1,NMOD                                                    AVLB 380
   30 MTGRDN(J) = MTGRDN(J) + 4.0 * RGRADN(J)                           AVLB 390
   50 CONTINUE                                                          AVLB 400
      DO 100 I=1,N1                                                     AVLB 410
      IF (N1 .EQ. 0) GO TO 100                                          AVLB 420
      T = DELTA * 2 * I                                                 AVLB 430
      CALL MODULE                                                       AVLB 440
      CALL RELIAB                                                       AVLB 450
      RSYS = RLBTY                                                      AVLB 460
      MTBM = MTBM + 2. * RSYS                                           AVLB 470
      IF(FLAG .EQ. 1) GO TO 100                                         AVLB 480
      CALL SENSE                                                        AVLB 490
      DO 70 J=1,NMOD                                                    AVLB 500
   70 MTGRDN(J) = MTGRDN(J) + 2.0 * RGRADN(J)                           AVLB 510
  100 CONTINUE                                                          AVLB 520
      T = TM                                                            AVLB 530
      CALL MODULE                                                       AVLB 540
      CALL RELIAB                                                       AVLB 550
      RSYS = RLBTY                                                      AVLB 560
      MTBM = (MTBM+RSYS) * DELTA/3.                                     AVLB 570
      MDT = TEM * (1.-RSYS) + TSC * RSYS                                AVLB 580
      ASYS = MTBM/(MTBM+MDT)                                            AVLB 590
      DUMMY1 = 0.0                                                      AVLB 600
      DO 115 I=1,NMOD                                                   AVLB 610
  115 DUMMY1 = DUMMY1 + NUNIT(I) * COSTF(I)                             AVLB 620
      PSYS = INCOM*TLF*ASYS - DUMMY1*(1.+(1.-ASYS)*TLF*COSTMF/100.)     AVLB 630
      IF(FLAG .EQ. 1) GO TO 1000                                        AVLB 640
      CALL SENSE                                                        AVLB 650
      DO 120 I=1,NMOD                                                   AVLB 660
  120 MTGRDN(I) = (MTGRDN(I)+RGRADN(I)) * DELTA/3.                      AVLB 670
      RGRADT = 0.0                                                      AVLB 680
      DO 130 I=1,NMOD                                                   AVLB 690
      IF(KODE(I) .EQ. 0) GO TO 140                                      AVLB 700
      RGRADT = RGRADT - ((NUNIT(I)-MUNIT(I)+1)/TM) * RGRADN(I)          AVLB 710
```

```
      GO TO 130                                                         AVLB 720
  140 RGRADT = RGRADT - (LAMDA(I)*(NUNIT(I)-MUNIT(I)+1)/                AVLB 730
     1  (1.-EXP(-LAMDA(I)*TM))) * RGRADN(I)                             AVLB 740
  130 CONTINUE                                                          AVLB 750
      DO 150 I=1,NMOD                                                   AVLB 760
      AGRADN(I) = (MDT*MTGRDN(I) + MTBM*(TEM-TSC)*RGRADN(I))/           AVLB 770
     1  (MTBM+MDT)**2                                                   AVLB 780
      PGRADN(I) = INCOM*TLF*AGRADN(I) + DUMMY1*TLF*AGRADN(I)*COSTMF/100.AVLB 790
     1  - COSTF(I)*(1.+(1.-ASYS)*TLF*COSTMF/100.)                       AVLB 800
  150 CONTINUE                                                          AVLB 810
      AGRADT = (MDT*RSYS + MTBM*(TEM-TSC)*RGRADT)/(MTBM+MDT)**2         AVLB 820
      AGRADT = AGRADT * TSTEP                                           AVLB 830
      PGRADT = (INCOM + DUMMY1*COSTMF/100.) *TLF*AGRADT                 AVLB 840
 1000 RETURN                                                            AVLB 850
      END                                                               AVLB 860
```

```
      SUBROUTINE CONSTR                                                 CONS  10
C     THIS SUBROUTINE COMPUTES THE SYSTEM CONSTRAINTS.                  CONS  20
      IMPLICIT INTEGER*4 (A-Z)                                          CONS  30
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  CONS  40
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             CONS  50
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        CONS  60
      INTEGER*2 P,PP,PATH                                               CONS  70
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     CONS  80
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 CONS  90
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      CONS 100
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   CONS 110
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      CONS 120
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       CONS 130
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   CONS 140
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CONS 150
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTCONS 160
C                                                                       CONS 170
      CONST(1) = 0.0                                                    CONS 180
      CONST(2) = 0.0                                                    CONS 190
      DO 10 I=1,NMOD                                                    CONS 200
      CONST(1) = CONST(1) + NUNIT(I)*COSTF(I)                           CONS 210
      CONST(2) = CONST(2) + NUNIT(I)*WUNIT(I)                           CONS 220
   10 CONTINUE                                                          CONS 230
      CONST(1) = CONST(1) * (1.+(1.-ASYS)*TLF*COSTMF/100.)              CONS 240
      CONST(1) = CONST(1) - CMCOST                                      CONS 250
      CONST(2) = CONST(2) - WEIGHT                                      CONS 260
      RETURN                                                            CONS 270
      END                                                               CONS 280
```

```
      SUBROUTINE FESIBL                                                 FESI  10
C     THIS SUBROUTINE FINDS THE FEASIBLE SOLUTION BY THE METHOD         FESI  20
C     OF WEIGHTED PERPENDICULARS.                                       FESI  30
      IMPLICIT INTEGER*4 (A-Z)                                          FESI  40
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  FESI  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             FESI  60
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        FESI  70
      INTEGER*2 P,PP,PATH                                               FESI  80
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     FESI  90
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 FESI 100
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      FESI 110
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   FESI 120
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      FESI 130
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       FESI 140
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   FESI 150
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,FESI 160
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTFESI 170
      REAL*4 Z(50),ZMIN,S,DUMMY1,ZMAX                                   FESI 180
C                                                                       FESI 190
      MOVE = 0                                                          FESI 200
      INDEX = 0                                                         FESI 210
    5 FLAG = 0                                                          FESI 220
      CALL AVLBTY                                                       FESI 230
      CALL CONSTR                                                       FESI 240
      WRITE(6,1400) (NUNIT(I),I=1,NMOD  )                               FESI 250
      WRITE(6,1450) TM                                                  FESI 260
      WRITE(6,1600)  CONST(1),CONST(2)                                  FESI 270
      IF(CONST(1) .LE. 0.0  .AND.  CONST(2) .LE. 0.0) GO TO 1000        FESI 280
      IF(CONST(1) .LE. 0.0) GO TO 30                                    FESI 290
      S = CONST(1)                                                      FESI 300
      DUMMY1 = 0.0                                                      FESI 310
      DO 10 I=1,NMOD                                                    FESI 320
   10 DUMMY1 = DUMMY1 + NUNIT(I)*COSTF(I)                               FESI 330
      DO 20 I=1,NMOD                                                    FESI 340
   20 Z(I) = -CONST(1) * ( COSTF(I)*(1.+(1.-ASYS)*TLF*COSTMF/100.) -    FESI 350
```

```
      1  DUMMY1*AGRADN(I)*TLF*COSTMF/100. )                                FESI 360
      Z(NMOD1 ) = -CONST(1) * (-DUMMY1*AGRADT*TLF*COSTMF/100.)            FESI 370
   30 IF(CONST(2) .LE. 0.0) GO TO 50                                      FESI 380
      S = S + CONST(2)                                                    FESI 390
      DO 40 I=1,NMOD                                                      FESI 400
   40 Z(I) = Z(I) - CONST(2) * WUNIT(I)                                   FESI 410
      Z(NMOD1 ) = Z(NMOD1 ) - CONST(2)*0.0                                FESI 420
   50 DO 60 I=1,NMOD1                                                     FESI 430
   60 Z(I) = +Z(I)/S                                                      FESI 440
      ZMAX = ABS(Z(1))                                                    FESI 450
      DO 70 J=2,NMOD1                                                     FESI 460
      IF (ZMAX .LT. ABS(Z(J))) ZMAX = ABS(Z(J))                           FESI 470
   70 CONTINUE                                                            FESI 480
      DO 80 I=1,NMOD                                                      FESI 490
      IF(Z(I)) 90,80,100                                                  FESI 500
   90 Z(I) = (Z(I)/ZMAX) * 2. - 0.5                                       FESI 510
      Z(I) = AINT(Z(I))                                                   FESI 520
      GO TO 80                                                            FESI 530
  100 Z(I) = (Z(I)/ZMAX) * 2. + 0.5                                       FESI 540
      Z(I) = AINT(Z(I))                                                   FESI 550
   80 CONTINUE                                                            FESI 560
      Z(NMOD1) = (Z(NMOD1)/ZMAX) * 2.                                     FESI 570
      INDEX = 0                                                           FESI 580
      MOVE = MOVE + 1                                                     FESI 590
      DO 120 I=1,NMOD                                                     FESI 600
      NUNIT(I) = NUNIT(I) + Z(I)                                          FESI 610
      IF(NUNIT(I).LT. NMIN(I) .OR. NUNIT(I).GT.NMAX(I)) INDEX=INDEX+1     FESI 620
      IF(NUNIT(I) .LT.  NMIN(I)) NUNIT(I) =  NMIN(I)                      FESI 630
      IF(NUNIT(I) .GT. NMAX(I)) NUNIT(I) = NMAX(I)                        FESI 640
  120 CONTINUE                                                            FESI 650
      TM = TM + Z(NMOD1 ) * TSTEP                                         FESI 660
      IF(TM.LT.TMIN .OR. TM.GT.TMAX) INDEX = INDEX+1                      FESI 670
      IF(TM .LT. TMIN) TM = TMIN                                          FESI 680
      IF(TM .GT. TMAX) TM = TMAX                                          FESI 690
      IF (MOVE .GT. 20  .OR.  INDEX .EQ. NMOD1 )  GO TO 999               FESI 700
      GO TO 5                                                             FESI 710
```

```
  999 WRITE(6,2500)                                                     FESI 720
 1000 RETURN                                                            FESI 730
 1400 FORMAT(1H0,/,10X,'NUNIT(I) =',20I5)                               FESI 740
 1450 FORMAT (1H0,10X,'TM =',G13.5)                                     FESI 750
 1600 FORMAT(1H0,/,30X,'CONSTRAINT(1) =',G13.6,//,30X,'CONSTRAINT(2) =',FESI 760
     1  G13.6)                                                          FESI 770
 2500 FORMAT(1H0,////,30X,'FESIBL POINT CAN NOT BE LOCATED',////)       FESI 780
      END                                                               FESI 790
```

```
      SUBROUTINE SORT                                                   SORT  10
C     THIS SUBROUTINE ARRANGES THE GRADIENT VECTOR IN ASCENDING ORDER.  SORT  20
      IMPLICIT INTEGER*4 (A-Z)                                          SORT  30
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  SORT  40
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             SORT  50
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        SORT  60
      INTEGER*2 P,PP,PATH                                               SORT  70
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     SORT  80
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 SORT  90
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      SORT 100
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   SORT 110
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      SORT 120
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       SORT 130
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   SORT 140
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,SORT 150
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTSORT 160
      REAL*4 GRDMIN,DUMMY(50)                                           SORT 170
C                                                                       SORT 180
      DO 5 I=1,NMOD                                                     SORT 190
      DUMMY(I) = AGRADN(I)                                              SORT 200
    5 CONTINUE                                                          SORT 210
      DUMMY(NMOD1 ) = AGRADT                                            SORT 220
      NONZRO = 0                                                        SORT 230
      DO 10 I=1,NMOD                                                    SORT 240
      IF(ABS(DUMMY(I)) .LT. 1.E-15) GO TO 10                            SORT 250
      NONZRO = NONZRO + 1                                               SORT 260
   10 CONTINUE                                                          SORT 270
      DO 20 I=1,NONZRO                                                  SORT 280
      GRDMIN = 1.E30                                                    SORT 290
      DO 30 J=1,NMOD                                                    SORT 300
      IF(ABS(DUMMY(J)) .LT. 1.E-15) GO TO 30                            SORT 310
      I1 = I - 1                                                        SORT 320
      IF(I1 .EQ.0) GO TO 50                                             SORT 330
      DO 40 K=1,I1                                                      SORT 340
      IF(J .EQ. NSORT(K)) GO TO 30                                      SORT 350
```

```
   40 CONTINUE                                                          SORT 360
   50 IF(GRDMIN .LE. ABS(DUMMY (J))) GO TO 30                           SORT 370
      GRDMIN = ABS(DUMMY (J))                                           SORT 380
      NSORT(I) = J                                                      SORT 390
   30 CONTINUE                                                          SORT 400
   20 CONTINUE                                                          SORT 410
      RETURN                                                            SORT 420
      END                                                               SORT 430
```

```
      SUBROUTINE OPTIM                                                  OPTI  10
C     THIS SUBROUTINE CHECKS FOR AN IMPROVED FEASIBLE SOLUTION.         OPTI  20
      IMPLICIT INTEGER*4 (A-Z)                                          OPTI  30
       COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                 OPTI  40
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             OPTI  50
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        OPTI  60
      INTEGER*2 P,PP,PATH                                               OPTI  70
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     OPTI  80
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 OPTI  90
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      OPTI 100
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   OPTI 110
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      OPTI 120
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       OPTI 130
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   OPTI 140
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,OPTI 150
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTOPTI 160
      REAL*4 TM1                                                        OPTI 170
      DIMENSION NUNIT1(50)                                              OPTI 180
C                                                                       OPTI 190
      MOVE = 0                                                          OPTI 200
    1 DO 10 I = 1,NMOD                                                  OPTI 210
      NUNIT1(I) = NUNIT(I)                                              OPTI 220
   10 NUNIT(I) = NUNIT(I) + AGRADN(I)                                   OPTI 230
      TM1 = TM                                                          OPTI 240
      TM = TM + AGRADT * TSTEP                                          OPTI 250
      DO 20 I=1,NMOD                                                    OPTI 260
      IF(NUNIT(I).LT. NMIN(I) .OR. NUNIT(I).GT.NMAX(I)) GO TO 100       OPTI 270
   20 CONTINUE                                                          OPTI 280
      IF(TM.LT.TMIN .OR. TM.GT.TMAX) GO TO 100                          OPTI 290
      FLAG = 1                                                          OPTI 300
      CALL AVLBTY                                                       OPTI 310
      IF(ASYS .LE. OBJFN) GO TO 100                                     OPTI 320
      CALL CONSTR                                                       OPTI 330
      IF(CONST(1).GT. 0.0 .OR. CONST(2).GT. 0.0)  GO TO 100             OPTI 340
      MOVE = MOVE + 1                                                   OPTI 350
```

```
      OBJFN = ASYS                                                      OPTI 360
      GO TO 999                                                         OPTI 370
  100 DO 30 I=1,NMOD                                                    OPTI 380
   30 NUNIT(I) = NUNIT1(I)                                              OPTI 390
      TM = TM1                                                          OPTI 400
  999 RETURN                                                            OPTI 410
      END                                                               OPTI 420
```

```
      SUBROUTINE OPTIMZ                                                 OPTI  10
C     THIS SUBROUTINE FINDS A LOCALLY OPTIMAL SOLUTION.                 OPTI  20
      IMPLICIT INTEGER*4 (A-Z)                                          OPTI  30
       COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                 OPTI  40
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             OPTI  50
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        OPTI  60
      INTEGER*2 P,PP,PATH                                               OPTI  70
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     OPTI  80
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 OPTI  90
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      OPTI 100
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   OPTI 110
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      OPTI 120
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       OPTI 130
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   OPTI 140
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,OPTI 150
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTOPTI 160
      REAL*4 DUMMY(50)                                                  OPTI 170
C                                                                       OPTI 180
      FLAG = 0                                                          OPTI 190
      CALL AVLBTY                                                       OPTI 200
      CALL SORT                                                         OPTI 210
      CALL CONSTR                                                       OPTI 220
      WRITE(6,1400) (NUNIT(I),I=1,NMOD  )                               OPTI 230
      WRITE(6,1450) TM                                                  OPTI 240
      WRITE(6,1500) ASYS,PSYS,RSYS                                      OPTI 250
      WRITE(6,1600)  CONST(1),CONST(2)                                  OPTI 260
      OBJFN = ASYS                                                      OPTI 270
      DO 5 I=1,NMOD                                                     OPTI 280
    5 DUMMY(I) = AGRADN(I)                                              OPTI 290
      DUMMY(NMOD1 ) = AGRADT                                            OPTI 300
      DO 200 I=1,NONZRO                                                 OPTI 310
      DO 110 J=1,NMOD                                                   OPTI 320
  110 AGRADN(J) = 0.0                                                   OPTI 330
      I1 = NONZRO - I + 1                                               OPTI 340
      K = NSORT(I1)                                                     OPTI 350
```

```
      AGRADN(K) = DUMMY(K)/ABS(DUMMY(K))                                OPTI 360
      AGRADT = DUMMY(NMOD1 )/ABS(DUMMY(K))                              OPTI 370
      CALL OPTIM                                                        OPTI 380
      IF(MOVE .EQ. 0) GO TO 200                                         OPTI 390
      GO TO 1000                                                        OPTI 400
  200 CONTINUE                                                          OPTI 410
      DO 300 I=1,NMOD                                                   OPTI 420
  300 AGRADN(I) = 0.0                                                   OPTI 430
      AGRADT = (100./TSTEP) * DUMMY(NMOD1)/ABS(DUMMY(NMOD1))            OPTI 440
      CALL OPTIM                                                        OPTI 450
 1000 RETURN                                                            OPTI 460
 1400 FORMAT(1H0,/,10X,'NUNIT(I) =',20I5)                               OPTI 470
 1450 FORMAT (1H0,10X,'TM =',G13.5)                                     OPTI 480
 1500 FORMAT (1H0,//,30X,'SYSTEM AVAILABILITY =',G13.6,//,30X,          OPTI 490
     1  'SYSTEM PROFIT =',G13.6,//,30X,'SYSTEM RELIABILITY =',G13.6)    OPTI 500
 1600 FORMAT(1H0,/,30X,'CONSTRAINT(1) =',G13.6,//,30X,'CONSTRAINT(2) =',OPTI 510
     1  G13.6)                                                          OPTI 520
      END                                                               OPTI 530
```

SYSTEM PROFIT OPTIMIZATION

```
C     MAIN PROGRAM                                                      MAIN  10
C     SYSTEM PROFIT OPTIMIZATION                                        MAIN  20
C     MODIFIED INTEGER GRADIENT METHOD                                  MAIN  30
      IMPLICIT INTEGER*4 (A-Z)                                          MAIN  40
       COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                 MAIN  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             MAIN  60
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        MAIN  70
      INTEGER*2 P,PP,PATH                                               MAIN  80
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     MAIN  90
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 MAIN 100
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      MAIN 110
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   MAIN 120
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      MAIN 130
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       MAIN 140
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   MAIN 150
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,MAIN 160
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTMAIN 170
      REAL*4 YFL,FCOST,MCOST                                            MAIN 180
C                                                                       MAIN 190
      PATHL = 300                                                       MAIN 200
      PLENTH = 1000                                                     MAIN 210
      PPL = 50                                                          MAIN 220
   10 CALL DATAIN                                                       MAIN 230
      READ(5,1000) INNOD,OUTNOD                                         MAIN 240
      CALL PATHS(INNOD)                                                 MAIN 250
      IF(STATUS) 20,10,20                                               MAIN 260
   20 CALL PATHP(OUTNOD)                                                MAIN 270
      WRITE(6,1100)                                                     MAIN 280
      DO 30 L=1,NPATH                                                   MAIN 290
      WRITE(6,1200) L,(PATH(L,M),M=1,NMOD  )                            MAIN 300
   30 CONTINUE                                                          MAIN 310
      CALL PATHC                                                        MAIN 320
      WRITE(6,1001) NPATH,MPATH                                         MAIN 330
      WRITE(6,2600)                                                     MAIN 340
      DO 35 L=1,MPATH                                                   MAIN 350
```

```
      WRITE(6,2700) L,PSIGN(L),(PATH(L,M),M=1,NMOD)                     MAIN 360
   35 CONTINUE                                                          MAIN 370
      KOUNT = 1                                                         MAIN 380
      ISEED = 984376521                                                 MAIN 390
      IX = ISEED                                                        MAIN 400
      DO 45 I=1,NMOD                                                    MAIN 410
   45 NUNIT(I) = 1                                                      MAIN 420
      TM = TMIN + (TMAX-TMIN)/2.                                        MAIN 430
      GO TO 55                                                          MAIN 440
C     GENERATING A UNIFORMLY DISTRIBUTED RANDOM STARTING POINT          MAIN 450
   40 DO 50 I=1,NMOD                                                    MAIN 460
      CALL RANDU(IX,IY,YFL)                                             MAIN 470
      NUNIT(I) = YFL*(NMAX(I)- NMIN(I)+1) + NMIN(I)                     MAIN 480
      IX = IY                                                           MAIN 490
   50 CONTINUE                                                          MAIN 500
      CALL RANDU(IX,IY,YFL)                                             MAIN 510
      TM = YFL*(TMAX-TMIN) + TMIN                                       MAIN 520
      IX = IY                                                           MAIN 530
   55 WRITE (6,1300) KOUNT                                              MAIN 540
      WRITE(6,2100)                                                     MAIN 550
      WRITE(6,1400) (NUNIT(I),I=1,NMOD  )                               MAIN 560
      WRITE(6,1450) TM                                                  MAIN 570
C     LOCATING A FEASIBLE POINT                                         MAIN 580
      WRITE(6,2200)                                                     MAIN 590
      CALL FESIBL                                                       MAIN 600
      WRITE(6,1410) MOVE,INDEX                                          MAIN 610
      IF (MOVE .GT. 20  .OR.  INDEX .EQ. NMOD1 )  GO TO 90              MAIN 620
      GO TO 60                                                          MAIN 630
   60 WRITE(6,2400)                                                     MAIN 640
      WRITE(6,1400) (NUNIT(I),I=1,NMOD  )                               MAIN 650
      WRITE(6,1450) TM                                                  MAIN 660
      WRITE(6,1600)  CONST(1),CONST(2)                                  MAIN 670
C     SEARCHING FOR LOCAL OPTIMUM                                       MAIN 680
      WRITE(6,2300)                                                     MAIN 690
   65 CALL OPTIMZ                                                       MAIN 700
      IF(MOVE .EQ. 0) GO TO 70                                          MAIN 710
```

```
      GO TO 65                                                          MAIN 720
   70 WRITE(6,2500)                                                     MAIN 730
      WRITE(6,1400) (NUNIT(I),I=1,NMOD  )                               MAIN 740
      WRITE(6,1450) TM                                                  MAIN 750
      FLAG = 1                                                          MAIN 760
      CALL AVLBTY                                                       MAIN 770
      CALL CONSTR                                                       MAIN 780
      FCOST = CONST(1) + CMCOST                                         MAIN 790
      MCOST = FCOST * (1.-ASYS) * TLF * COSTMF/100.                     MAIN 800
      WRITE(6,1500) ASYS,PSYS,RSYS                                      MAIN 810
      WRITE(6,1600)  CONST(1),CONST(2)                                  MAIN 820
      WRITE(6,1700) FCOST,MCOST                                         MAIN 830
      WRITE(6,1900) MTBM,MDT                                            MAIN 840
   90 KOUNT = KOUNT + 1                                                 MAIN 850
      IF (KOUNT .GT. 25) GO TO 10                                       MAIN 860
      GO TO 40                                                          MAIN 870
 1000 FORMAT (2I3)                                                      MAIN 880
 1001 FORMAT(1H0,//,30X,'NPATH = ',I3,//,30X,'MPATH = ',I5)             MAIN 890
 1100 FORMAT (1H0,//,T20,'PATHS IN BINARY CODE',/)                      MAIN 900
 1200 FORMAT (1H0,T16,'PATH',I3,' =  ',50I2)                            MAIN 910
 1300 FORMAT(1H1,40X,'KOUNT =',I3,//)                                   MAIN 920
 1400 FORMAT(1H0,/,10X,'NUNIT(I) =',20I5)                               MAIN 930
 1410 FORMAT (1H0,//,30X,'MOVE =',I4,10X,'INDEX =',I4)                  MAIN 940
 1450 FORMAT (1H0,10X,'TM =',G13.5)                                     MAIN 950
 1500 FORMAT (1H0,//,30X,'SYSTEM AVAILABILITY =',G13.6,//,30X,          MAIN 960
     1  'SYSTEM PROFIT =',G13.6,//,30X,'SYSTEM RELIABILITY =',G13.6)    MAIN 970
 1600 FORMAT(1H0,/,30X,'CONSTRAINT(1) =',G13.6,//,30X,'CONSTRAINT(2) =',MAIN 980
     1  G13.6)                                                          MAIN 990
 1700 FORMAT(1H0,/,30X,'FCOST =',G13.6,//,30X,'MCOST =',G13.6)          MAIN1000
 1900 FORMAT(1H0,20X,'MTBM =',G13.6,20X,'MDT =',G13.6)                  MAIN1010
 2100 FORMAT(1H0,50X,'THE INITIAL POINT IS')                            MAIN1020
 2200 FORMAT(1H0,50X,'SEARCHING FOR THE FEASIBLE POINT')                MAIN1030
 2300 FORMAT(1H0,50X,'SEARCHING FOR THE LOCAL OPTIMUM POINT')           MAIN1040
 2400 FORMAT(1H0,////,50X,'THE FEASIBLE POINT IS')                      MAIN1050
 2500 FORMAT(1H0,////,50X,'THE OPTIMAL POINT IS')                       MAIN1060
 2600 FORMAT(1H0,//,T20,'PATHS AND PATH COMBINATIONS IN BINARY CODE',/) MAIN1070
```

```
 2700 FORMAT(1H0,T16,'PATHC',I3,' = ',I2,4X,50I2)                    MAIN1080
      END                                                            MAIN1090
```

```
      SUBROUTINE DATAIN                                                 DATA  10
C     THIS SUBROUTINE READS AND ECHOCHECKS THE DATA                     DATA  20
      IMPLICIT INTEGER*4 (A-Z)                                          DATA  30
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  DATA  40
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             DATA  50
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        DATA  60
      INTEGER*2 P,PP,PATH                                               DATA  70
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     DATA  80
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 DATA  90
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      DATA 100
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   DATA 110
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      DATA 120
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       DATA 130
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   DATA 140
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,DATA 150
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTDATA 160
      DIMENSION TITLE(30)                                               DATA 170
C                                                                       DATA 180
      READ(5,999,END=100)( TITLE(I),I=1,30)                             DATA 190
      READ(5,1000) NNODES,NMOD,CMCOST,WEIGHT,INCOM,COSTMF,TLF,TEM,      DATA 200
     1  TSC,TMIN,TMAX,TSTEP,INTVL                                       DATA 210
      DO 10 I=1,NMOD                                                    DATA 220
   10 READ(5,1007) KODE(I),NMAX(I),NMIN(I),MUNIT(I),COSTF(I),WUNIT(I),  DATA 230
     1  LAMDA(I)                                                        DATA 240
      NMOD1 = NMOD + 1                                                  DATA 250
      FOR=0                                                             DATA 260
      REV=0                                                             DATA 270
      DO 40 I=1,1000                                                    DATA 280
      READ(5,1001)P(I,1),P(I,2),P(I,4)                                  DATA 290
      IF( P(I,1)-P(I,2) ) 20,50,30                                      DATA 300
   20 FOR=FOR + 1                                                       DATA 310
      GO TO 40                                                          DATA 320
   30 REV=REV + 1                                                       DATA 330
   40 CONTINUE                                                          DATA 340
   50 NEDGES=FOR + REV                                                  DATA 350
```

```
      NEDGS1=NEDGES + 1                                                 DATA 360
      FREV=FOR + 1                                                      DATA 370
      WRITE(6,1002) ( TITLE(I),I=1,30)                                  DATA 380
      WRITE(6,1003)                                                     DATA 390
      WRITE(6,1004)                                                     DATA 400
      WRITE(6,1003)                                                     DATA 410
      DO 60 I=1,NEDGES                                                  DATA 420
      WRITE(6,1005) I,P(I,1),P(I,2),P(I,4)                              DATA 430
      WRITE(6,1003)                                                     DATA 440
   60 CONTINUE                                                          DATA 450
      WRITE(6,1008)                                                     DATA 460
      WRITE(6,1011)                                                     DATA 470
      WRITE(6,1009)                                                     DATA 480
      WRITE(6,1011)                                                     DATA 490
      DO 70 I=1,NMOD                                                    DATA 500
      WRITE(6,1010) I,KODE(I),NMAX(I),NMIN(I),MUNIT(I),COSTF(I),        DATA 510
     1  WUNIT(I),LAMDA(I)                                               DATA 520
      WRITE(6,1011)                                                     DATA 530
   70 CONTINUE                                                          DATA 540
      WRITE(6,1006) CMCOST,WEIGHT,INCOM,COSTMF,TLF,TEM,TSC,TMIN,TMAX,   DATA 550
     1  TSTEP,INTVL                                                     DATA 560
  999 FORMAT(15A4)                                                      DATA 570
 1000 FORMAT (2I10,5F10.0/6F10.0)                                       DATA 580
 1001 FORMAT (3I10)                                                     DATA 590
 1002 FORMAT(1H1,//,T10,30A4,//,T50,'INVENTORY OF BRANCHES',/)          DATA 600
 1003 FORMAT (1H+,T38,44('_'))                                          DATA 610
 1004 FORMAT (1H ,T38,'|  BRANCH  |','  ORIGIN  |','   END    |',       DATA 620
     1  '  MODULE  |')                                                 DATA 630
 1005 FORMAT (1H ,T38,4('|',4X,I2,4X),'|')                              DATA 640
 1006 FORMAT (1H0,T10,'CMCOST=',G13.6,/,T10,'WEIGHT=',G13.6,/,T10,      DATA 650
     1  'INCOM =',G13.6,/,T10,'COSTMF=',G13.6,/,T10,'TLF   =',G13.6,/,  DATA 660
     2  T10,'TEM   =',G13.6,/,T10,'TSC   =',G13.6,/,T10,'TMIN  =',      DATA 670
     3  G13.6,/,T10,'TMAX  =',G13.6,/,T10,'TSTEP =',G13.6,/,T10,        DATA 680
     4  'INTVL =',G13.6)                                                DATA 690
 1007 FORMAT(4I10,2F10.0,E20.0)                                         DATA 700
 1008 FORMAT(1H0,T50,'INVENTORY OF MODULES',/)                          DATA 710
```

```
 1009 FORMAT(1H ,T20,'|  MODULE  |','   KODE   |',' MAX-UNIT |',        DATA 720
     1  ' MIN-UNIT |','M OUT OF N|','UNIT COSTF|','UNIT WEGHT|',        DATA 730
     1  ' FAILURE RATE |')                                              DATA 740
 1010 FORMAT(1H ,T20,5('|',4X,I2,4X),2('|',F10.3),'|',G14.5,'|')        DATA 750
 1011 FORMAT(1H+,T20,92('_'))                                            DATA 760
      RETURN                                                             DATA 770
  100 CALL EXIT                                                          DATA 780
      END                                                                DATA 790
```

```
      SUBROUTINE MODULE                                                 MODU  10
C     THIS SUBROUTINE COMPUTES THE MODULE RELIABILITY AND               MODU  20
C     SENSITIVITY OF MODULE RELIABILITY TO THE MODULE REDUNDANCY.       MODU  30
      IMPLICIT INTEGER*4 (A-Z)                                          MODU  40
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  MODU  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             MODU  60
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        MODU  70
      INTEGER*2 P,PP,PATH                                               MODU  80
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     MODU  90
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 MODU 100
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      MODU 110
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   MODU 120
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      MODU 130
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       MODU 140
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   MODU 150
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,MODU 160
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTMODU 170
      REAL*4 DUMMY1,DUMMY2,DUMMY3                                       MODU 180
C                                                                       MODU 190
      DO 200 K=1,NMOD                                                   MODU 200
      IF(KODE(K) .EQ. 0) GO TO 100                                      MODU 210
      DUMMY1 = EXP(-LAMDA(K)*T*MUNIT(K))                                MODU 220
      DUMMY2 = 1.0                                                      MODU 230
      DUMMY3 = LAMDA(K) * T * MUNIT(K)                                  MODU 240
      G(K) = 1.0                                                        MODU 250
      NM = NUNIT(K) - MUNIT(K)                                          MODU 260
      IF(NM .EQ. 0) GO TO 20                                            MODU 270
      DO 10 I=1,NM                                                      MODU 280
      DUMMY2 = DUMMY2 * DUMMY3/I                                        MODU 290
      G(K) = G(K) + DUMMY2                                              MODU 300
   10 CONTINUE                                                          MODU 310
   20 G(K) = G(K) * DUMMY1                                              MODU 320
      RIGRDN(K) = DUMMY1 * DUMMY2 * DUMMY3/(NM+1)                       MODU 330
      GO TO 200                                                         MODU 340
  100 DUMMY1 = EXP(-LAMDA(K)*T)                                         MODU 350
```

```
      DUMMY2 = DUMMY1**NUNIT(K)                                         MODU 360
      G(K) = DUMMY2                                                     MODU 370
      NM = NUNIT(K) - MUNIT(K)                                          MODU 380
      IF(NM .EQ. 0) GO TO 120                                           MODU 390
      DO 110 I=1,NM                                                     MODU 400
      DUMMY2 = DUMMY2 * ((1.-DUMMY1)/DUMMY1) * (NUNIT(K)-I+1)/I         MODU 410
      G(K) = G(K) + DUMMY2                                              MODU 420
  110 CONTINUE                                                          MODU 430
  120 RIGRDN(K) = DUMMY2 * (1.-DUMMY1) * MUNIT(K)/(NM+1)                MODU 440
  200 CONTINUE                                                          MODU 450
      RETURN                                                            MODU 460
      END                                                               MODU 470
```

```
      SUBROUTINE RELIAB                                                 RELI  10
C     THIS SUBROUTINE CALCULATES THE SYSTEM RELIABILITY FROM THE        RELI  20
C     MODULE RELIABILITIES.                                             RELI  30
      IMPLICIT INTEGER*4 (A-Z)                                          RELI  40
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  RELI  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             RELI  60
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        RELI  70
      INTEGER*2 P,PP,PATH                                               RELI  80
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     RELI  90
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 RELI 100
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      RELI 110
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   RELI 120
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      RELI 130
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       RELI 140
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   RELI 150
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,RELI 160
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTRELI 170
      REAL*4 GPATH                                                      RELI 180
C                                                                       RELI 190
      RLBTY = 0.0                                                       RELI 200
      DO 10 I=1,MPATH                                                   RELI 210
      GPATH = PSIGN(I)                                                  RELI 220
      DO 20 J=1,NMOD                                                    RELI 230
      IF (PATH(I,J) .EQ. 0) GO TO 20                                    RELI 240
      GPATH = GPATH * G(J)                                              RELI 250
   20 CONTINUE                                                          RELI 260
      RLBTY = RLBTY + GPATH                                             RELI 270
   10 CONTINUE                                                          RELI 280
      RETURN                                                            RELI 290
      END                                                               RELI 300
```

```
      SUBROUTINE SENSE                                                  SENS  10
C     THIS SUBROUTINE DETERMINES THE SENSITIVITY OF THE SYSTEM          SENS  20
C     RELIABILITY TO MODULE REDUNDANCIES.                               SENS  30
      IMPLICIT INTEGER*4 (A-Z)                                          SENS  40
      COMMON/MASON/FUR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  SENS  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             SENS  60
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        SENS  70
      INTEGER*2 P,PP,PATH                                               SENS  80
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     SENS  90
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 SENS 100
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      SENS 110
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   SENS 120
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      SENS 130
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       SENS 140
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   SENS 150
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,SENS 160
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTSENS 170
      REAL*4 DUMMY(50)                                                  SENS 180
C                                                                       SENS 190
      DO 10 I=1,NMOD                                                    SENS 200
   10 DUMMY(I) = G(I)                                                   SENS 210
      DO 100 I=1,NMOD                                                   SENS 220
      DO 90 J=1,NMOD                                                    SENS 230
   90 G(J) = DUMMY(J)                                                   SENS 240
      G(I) = 1.0                                                        SENS 250
      CALL RELIAB                                                       SENS 260
      RGRADN(I) = RLBTY                                                 SENS 270
      G(I) = 0.0                                                        SENS 280
      CALL RELIAB                                                       SENS 290
      RGRADN(I) = (RGRADN(I)-RLBTY) * RIGRDN(I)                         SENS 300
  100 CONTINUE                                                          SENS 310
      DO 150 I=1,NMOD                                                   SENS 320
  150 G(I) = DUMMY(I)                                                   SENS 330
      RETURN                                                            SENS 340
      END                                                               SENS 350
```

```
      SUBROUTINE AVLBTY                                                 AVLB  10
C     THIS SUBROUTINE DETERMINES SYSTEM AVAILABILITY, SYSTEM PROFIT,    AVLB  20
C     AND SENSITIVITY OF SYSTEM AVAILABILITY AND SYSTEM PROFIT TO EACH  AVLB  30
C     MODULE'S REDUNDANCY AND TO SYSTEM MAINTENANCE INTERVAL.           AVLB  40
      IMPLICIT INTEGER*4 (A-Z)                                          AVLB  50
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  AVLB  60
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             AVLB  70
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        AVLB  80
      INTEGER*2 P,PP,PATH                                               AVLB  90
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     AVLB 100
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 AVLB 110
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      AVLB 120
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   AVLB 130
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      AVLB 140
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       AVLB 150
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   AVLB 160
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,AVLB 170
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTAVLB 180
      REAL*4 DELTA,DUMMY1                                               AVLB 190
C                                                                       AVLB 200
      EVEN = IFIX((TM/INTVL) + 1.0)                                     AVLB 210
      EVEN = (EVEN/2) * 2                                               AVLB 220
      IF (EVEN .EQ. 0) EVEN = 2                                         AVLB 230
      DELTA = TM/EVEN                                                   AVLB 240
      MTBM = 1.0                                                        AVLB 250
      DO 10 J=1,NMOD                                                    AVLB 260
   10 MTGRDN(J) = 0.0                                                   AVLB 270
      N = EVEN/2                                                        AVLB 280
      N1 = N-1                                                          AVLB 290
      DO 50 I=1,N                                                       AVLB 300
      T = DELTA * (2*I-1)                                               AVLB 310
      CALL MODULE                                                       AVLB 320
      CALL RELIAB                                                       AVLB 330
      RSYS = RLBTY                                                      AVLB 340
      MTBM = MTBM + 4.0 * RSYS                                          AVLB 350
```

```
      IF(FLAG .EQ. 1) GO TO 50                                          AVLB 360
      CALL SENSE                                                        AVLB 370
      DO 30 J=1,NMOD                                                    AVLB 380
   30 MTGRDN(J) = MTGRDN(J) + 4.0 * RGRADN(J)                           AVLB 390
   50 CONTINUE                                                          AVLB 400
      DO 100 I=1,N1                                                     AVLB 410
      IF (N1 .EQ. 0) GO TO 100                                          AVLB 420
      T = DELTA * 2 * I                                                 AVLB 430
      CALL MODULE                                                       AVLB 440
      CALL RELIAB                                                       AVLB 450
      RSYS = RLBTY                                                      AVLB 460
      MTBM = MTBM + 2. * RSYS                                           AVLB 470
      IF(FLAG .EQ. 1) GO TO 100                                         AVLB 480
      CALL SENSE                                                        AVLB 490
      DO 70 J=1,NMOD                                                    AVLB 500
   70 MTGRDN(J) = MTGRDN(J) + 2.0 * RGRADN(J)                           AVLB 510
  100 CONTINUE                                                          AVLB 520
      T = TM                                                            AVLB 530
      CALL MODULE                                                       AVLB 540
      CALL RELIAB                                                       AVLB 550
      RSYS = RLBTY                                                      AVLB 560
      MTBM = (MTBM+RSYS) * DELTA/3.                                     AVLB 570
      MDT = TEM * (1.-RSYS) + TSC * RSYS                                AVLB 580
      ASYS = MTBM/(MTBM+MDT)                                            AVLB 590
      DUMMY1 = 0.0                                                      AVLB 600
      DO 115 I=1,NMOD                                                   AVLB 610
  115 DUMMY1 = DUMMY1 + NUNIT(I) * COSTF(I)                             AVLB 620
      PSYS = INCOM*TLF*ASYS - DUMMY1*(1.+(1.-ASYS)*TLF*COSTMF/100.)     AVLB 630
      IF(FLAG .EQ. 1) GO TO 1000                                        AVLB 640
      CALL SENSE                                                        AVLB 650
      DO 120 I=1,NMOD                                                   AVLB 660
  120 MTGRDN(I) = (MTGRDN(I)+RGRADN(I)) * DELTA/3.                      AVLB 670
      RGRADT = 0.0                                                      AVLB 680
      DO 130 I=1,NMOD                                                   AVLB 690
      IF(KODE(I) .EQ. 0) GO TO 140                                      AVLB 700
      RGRADT = RGRADT - ((NUNIT(I)-MUNIT(I)+1)/TM) * RGRADN(I)          AVLB 710
```

```
      GO TO 130                                                         AVLB 720
  140 RGRADT = RGRADT - (LAMDA(I)*(NUNIT(I)-MUNIT(I)+1)/                AVLB 730
     1  (1.-EXP(-LAMDA(I)*TM))) * RGRADN(I)                             AVLB 740
  130 CONTINUE                                                          AVLB 750
      DO 150 I=1,NMOD                                                   AVLB 760
      AGRADN(I) = (MDT*MTGRDN(I) + MTBM*(TEM-TSC)*RGRADN(I))/           AVLB 770
     1  (MTBM+MDT)**2                                                   AVLB 780
      PGRADN(I) = INCOM*TLF*AGRADN(I) + DUMMY1*TLF*AGRADN(I)*COSTMF/100.AVLB 790
     1  - COSTF(I)*(1.+(1.-ASYS)*TLF*COSTMF/100.)                       AVLB 800
  150 CONTINUE                                                          AVLB 810
      AGRADT = (MDT*RSYS + MTBM*(TEM-TSC)*RGRADT)/(MTBM+MDT)**2         AVLB 820
      AGRADT = AGRADT * TSTEP                                           AVLB 830
      PGRADT = (INCOM + DUMMY1*COSTMF/100.) *TLF*AGRADT                 AVLB 840
 1000 RETURN                                                            AVLB 850
      END                                                               AVLB 860
```

```
      SUBROUTINE CONSTR                                                 CONS  10
C     THIS SUBROUTINE COMPUTES THE SYSTEM CONSTRAINTS.                  CONS  20
      IMPLICIT INTEGER*4 (A-Z)                                          CONS  30
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  CONS  40
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             CONS  50
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        CONS  60
      INTEGER*2 P,PP,PATH                                               CONS  70
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     CONS  80
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 CONS  90
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      CONS 100
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   CONS 110
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      CONS 120
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       CONS 130
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   CONS 140
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CONS 150
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTCONS 160
C                                                                       CONS 170
      CONST(1) = 0.0                                                    CONS 180
      CONST(2) = 0.0                                                    CONS 190
      DO 10 I=1,NMOD                                                    CONS 200
      CONST(1) = CONST(1) + NUNIT(I)*COSTF(I)                           CONS 210
      CONST(2) = CONST(2) + NUNIT(I)*WUNIT(I)                           CONS 220
   10 CONTINUE                                                          CONS 230
      CONST(1) = CONST(1) - CMCOST                                      CONS 240
      CONST(2) = CONST(2) - WEIGHT                                      CONS 250
      RETURN                                                            CONS 260
      END                                                               CONS 270
```

```
      SUBROUTINE FESIBL                                                 FESI  10
C     THIS SUBROUTINE FINDS THE FEASIBLE SOLUTION BY THE METHOD         FESI  20
C     OF WEIGHTED PERPENDICULARS.                                       FESI  30
      IMPLICIT INTEGER*4 (A-Z)                                          FESI  40
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  FESI  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             FESI  60
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        FESI  70
      INTEGER*2 P,PP,PATH                                               FESI  80
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     FESI  90
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 FESI 100
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      FESI 110
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   FESI 120
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      FESI 130
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       FESI 140
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   FESI 150
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,FESI 160
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTFESI 170
      REAL*4 Z(50),ZMIN,S,DUMMY1,ZMAX                                   FESI 180
C                                                                       FESI 190
      MOVE = 0                                                          FESI 200
      INDEX = 0                                                         FESI 210
    5 CALL CONSTR                                                       FESI 220
      WRITE(6,1400) (NUNIT(I),I=1,NEDGES)                               FESI 230
      WRITE(6,1450) TM                                                  FESI 240
      WRITE(6,1600)  CONST(1),CONST(2)                                  FESI 250
      IF(CONST(1) .LE. 0.0  .AND.  CONST(2) .LE. 0.0) GO TO 1000        FESI 260
      IF(CONST(1) .LE. 0.0) GO TO 30                                    FESI 270
      S = CONST(1)                                                      FESI 280
      DO 20 I=1,NMOD                                                    FESI 290
   20 Z(I) = -CONST(1) * COSTF(I)                                       FESI 300
      Z(NMOD1 ) = -CONST(1) * 0.0                                       FESI 310
   30 IF(CONST(2) .LE. 0.0) GO TO 50                                    FESI 320
      S = S + CONST(2)                                                  FESI 330
      DO 40 I=1,NMOD                                                    FESI 340
   40 Z(I) = Z(I) - CONST(2) * WUNIT(I)                                 FESI 350
```

```
      Z(NMOD1 ) = Z(NMOD1 ) - CONST(2)*0.0                              FESI 360
   50 DO 60 I=1,NMOD1                                                   FESI 370
   60 Z(I) = +Z(I)/S                                                    FESI 380
      ZMAX = ABS(Z(1))                                                  FESI 390
      DO 70 J=2,NMOD1                                                   FESI 400
      IF (ZMAX .LT. ABS(Z(J))) ZMAX = ABS(Z(J))                         FESI 410
   70 CONTINUE                                                          FESI 420
      DO 80 I=1,NMOD                                                    FESI 430
      IF(Z(I)) 90,80,100                                                FESI 440
   90 Z(I) = (Z(I)/ZMAX) * 2. - 0.5                                     FESI 450
      Z(I) = AINT(Z(I))                                                 FESI 460
      GO TO 80                                                          FESI 470
  100 Z(I) = (Z(I)/ZMAX) * 2. + 0.5                                     FESI 480
      Z(I) = AINT(Z(I))                                                 FESI 490
   80 CONTINUE                                                          FESI 500
      Z(NMOD1) = (Z(NMOD1)/ZMAX) * 2.                                   FESI 510
      INDEX = 0                                                         FESI 520
      MOVE = MOVE + 1                                                   FESI 530
      DO 120 I=1,NMOD                                                   FESI 540
      NUNIT(I) = NUNIT(I) + Z(I)                                        FESI 550
      IF(NUNIT(I).LT. NMIN(I) .OR. NUNIT(I).GT.NMAX(I)) INDEX=INDEX+1   FESI 560
      IF(NUNIT(I) .LT.  NMIN(I)) NUNIT(I) =  NMIN(I)                    FESI 570
      IF(NUNIT(I) .GT. NMAX(I)) NUNIT(I) = NMAX(I)                      FESI 580
  120 CONTINUE                                                          FESI 590
      TM = TM + Z(NMOD1 ) * TSTEP                                       FESI 600
      IF(TM.LT.TMIN .OR. TM.GT.TMAX) INDEX = INDEX+1                    FESI 610
      IF(TM .LT. TMIN) TM = TMIN                                        FESI 620
      IF(TM .GT. TMAX) TM = TMAX                                        FESI 630
      IF (MOVE .GT. 20  .OR.  INDEX .EQ. NMOD1 )  GO TO 999             FESI 640
      GO TO 5                                                           FESI 650
  999 WRITE(6,2500)                                                     FESI 660
 1000 RETURN                                                            FESI 670
 1400 FORMAT(1H0,/,10X,'NUNIT(I) =',20I5)                               FESI 680
 1450 FORMAT (1H0,10X,'TM =',G13.5)                                     FESI 690
 1600 FORMAT(1H0,/,30X,'CONSTRAINT(1) =',G13.6,//,30X,'CONSTRAINT(2) =',FESI 700
     1  G13.6)                                                          FESI 710
```

```
 2500 FORMAT(1H0,////,30X,'FESIBL POINT CAN NOT BE LOCATED',////)        FESI 720
      END                                                                FESI 730
```

```
      SUBROUTINE SORT                                                   SORT  10
C     THIS SUBROUTINE ARRANGES THE GRADIENT VECTOR IN ASCENDING ORDER.  SORT  20
      IMPLICIT INTEGER*4 (A-Z)                                          SORT  30
      COMMON/MASON/FUR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  SORT  40
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             SORT  50
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        SORT  60
      INTEGER*2 P,PP,PATH                                               SORT  70
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     SORT  80
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 SORT  90
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      SORT 100
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   SORT 110
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      SORT 120
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       SORT 130
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   SORT 140
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,SORT 150
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTSORT 160
      REAL*4 GRDMIN,DUMMY(50)                                           SORT 170
C                                                                       SORT 180
      DO 5 I=1,NMOD                                                     SORT 190
      DUMMY(I) = PGRADN(I)                                              SORT 200
    5 CONTINUE                                                          SORT 210
      DUMMY(NMOD1 ) = PGRADT                                            SORT 220
      NONZRO = 0                                                        SORT 230
      DO 10 I=1,NMOD                                                    SORT 240
      IF(ABS(DUMMY(I)) .LT. 1.E-15) GO TO 10                            SORT 250
      NONZRO = NONZRO + 1                                               SORT 260
   10 CONTINUE                                                          SORT 270
      DO 20 I=1,NONZRO                                                  SORT 280
      GRDMIN = 1.E30                                                    SORT 290
      DO 30 J=1,NEDGES                                                  SORT 300
      IF(ABS(DUMMY(J)) .LT. 1.E-15) GO TO 30                            SORT 310
      I1 = I - 1                                                        SORT 320
      IF(I1 .EQ.0) GO TO 50                                             SORT 330
      DO 40 K=1,I1                                                      SORT 340
      IF(J .EQ. NSORT(K)) GO TO 30                                      SORT 350
```

```
   40 CONTINUE                                                          SORT 360
   50 IF(GRDMIN .LE. ABS(DUMMY (J))) GO TO 30                           SORT 370
      GRDMIN = ABS(DUMMY (J))                                           SORT 380
      NSORT(I) = J                                                      SORT 390
   30 CONTINUE                                                          SORT 400
   20 CONTINUE                                                          SORT 410
      RETURN                                                            SORT 420
      END                                                               SORT 430
```

```
      SUBROUTINE OPTIM                                                  OPTI  10
C     THIS SUBROUTINE CHECKS FOR AN IMPROVED FEASIBLE SOLUTION.         OPTI  20
      IMPLICIT INTEGER*4 (A-Z)                                          OPTI  30
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  OPTI  40
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             OPTI  50
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        OPTI  60
      INTEGER*2 P,PP,PATH                                               OPTI  70
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     OPTI  80
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 OPTI  90
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      OPTI 100
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   OPTI 110
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      OPTI 120
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       OPTI 130
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   OPTI 140
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,OPTI 150
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTOPTI 160
      REAL*4 TM1                                                        OPTI 170
      DIMENSION NUNIT1(50)                                              OPTI 180
C                                                                       OPTI 190
      MOVE = 0                                                          OPTI 200
    1 DO 10 I = 1,NMOD                                                  OPTI 210
      NUNIT1(I) = NUNIT(I)                                              OPTI 220
   10 NUNIT(I) = NUNIT(I) + PGRADN(I)                                   OPTI 230
      TM1 = TM                                                          OPTI 240
      TM = TM + PGRADT * TSTEP                                          OPTI 250
      DO 20 I=1,NMOD                                                    OPTI 260
      IF(NUNIT(I).LT. NMIN(I) .OR. NUNIT(I).GT.NMAX(I)) GO TO 100       OPTI 270
   20 CONTINUE                                                          OPTI 280
      IF(TM.LT.TMIN .OR. TM.GT.TMAX) GO TO 100                          OPTI 290
      FLAG = 1                                                          OPTI 300
      CALL AVLBTY                                                       OPTI 310
      IF(PSYS .LE. OBJFN) GO TO 100                                     OPTI 320
      CALL CONSTR                                                       OPTI 330
      IF(CONST(1).GT. 0.0 .OR. CONST(2).GT. 0.0)  GO TO 100             OPTI 340
      MOVE = MOVE + 1                                                   OPTI 350
```

```
      OBJFN = PSYS                                                      OPTI 360
      GO TO 999                                                         OPTI 370
  100 DO 30 I=1,NMOD                                                    OPTI 380
   30 NUNIT(I) = NUNIT1(I)                                              OPTI 390
      TM = TM1                                                          OPTI 400
  999 RETURN                                                            OPTI 410
      END                                                               OPTI 420
```

```
      SUBROUTINE OPTIMZ                                                 OPTI  10
C     THIS SUBROUTINE FINDS A LOCALLY OPTIMAL SOLUTION.                 OPTI  20
      IMPLICIT INTEGER*4 (A-Z)                                          OPTI  30
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  OPTI  40
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             OPTI  50
     2  PATH(300,50),NMOD,NMOD1,MPATH,PSIGN(300)                        OPTI  60
      INTEGER*2 P,PP,PATH                                               OPTI  70
      COMMON/OPTM1/NUNIT(50),NMAX(50),MUNIT(50),KODE(50),NSORT(50),     OPTI  80
     1  NMIN(50),FLAG,INDEX,MOVE,NONZRO                                 OPTI  90
      COMMON/OPTM2/AGRADN(50),CONST(10),COSTF(50),G(50),LAMDA(50),      OPTI 100
     1  MTGRDN(50),PGRADN(50),RGRADN(50),RIGRDN(50),WUNIT(50),AGRADT,   OPTI 110
     2  ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,CMCOST,      OPTI 120
     3  RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHT       OPTI 130
      REAL*4  AGRADN,CONST,COSTF,G,LAMDA,MTGRDN,PGRADN,RGRADN,RIGRDN,   OPTI 140
     1  WUNIT,AGRADT,ASYS,COSTMF,INCOM,INTVL,MTBM,MDT,OBJFN,PGRADT,PSYS,OPTI 150
     2  CMCOST,RLBTY,RGRADT,RSYS,T,TEM,TLF,TM,TMIN,TMAX,TSC,TSTEP,WEIGHTOPTI 160
      REAL*4 DUMMY(50)                                                  OPTI 170
C                                                                       OPTI 180
      FLAG = 0                                                          OPTI 190
      CALL AVLBTY                                                       OPTI 200
      CALL SORT                                                         OPTI 210
      CALL CONSTR                                                       OPTI 220
      WRITE(6,1400) (NUNIT(I),I=1,NMOD  )                               OPTI 230
      WRITE(6,1450) TM                                                  OPTI 240
      WRITE(6,1500) ASYS,PSYS,RSYS                                      OPTI 250
      WRITE(6,1600)  CONST(1),CONST(2)                                  OPTI 260
      OBJFN = PSYS                                                      OPTI 270
      DO 5 I=1,NMOD                                                     OPTI 280
    5 DUMMY(I) = PGRADN(I)                                              OPTI 290
      DUMMY(NMOD1 ) = PGRADT                                            OPTI 300
      DO 200 I=1,NONZRO                                                 OPTI 310
      DO 110 J=1,NMOD                                                   OPTI 320
  110 PGRADN(J) = 0.0                                                   OPTI 330
      I1 = NONZRO - I + 1                                               OPTI 340
      K = NSORT(I1)                                                     OPTI 350
```

```
      PGRADN(K) = DUMMY(K)/ABS(DUMMY(K))                                OPTI 360
      PGRADT = DUMMY(NMOD1 )/ABS(DUMMY(K))                              OPTI 370
      CALL OPTIM                                                        OPTI 380
      IF(MOVE .EQ. 0) GO TO 200                                         OPTI 390
      GO TO 1000                                                        OPTI 400
  200 CONTINUE                                                          OPTI 410
      DO 300 I=1,NMOD                                                   OPTI 420
  300 PGRADN(I) = 0.0                                                   OPTI 430
      PGRADT = (100./TSTEP) * DUMMY(NMOD1)/ABS(DUMMY(NMOD1))            OPTI 440
      CALL OPTIM                                                        OPTI 450
 1000 RETURN                                                            OPTI 460
 1400 FORMAT(1H0,/,10X,'NUNIT(I) =',20I5)                               OPTI 470
 1450 FORMAT (1H0,10X,'TM =',G13.5)                                     OPTI 480
 1500 FORMAT (1H0,//,30X,'SYSTEM AVAILABILITY =',G13.6,//,30X,          OPTI 490
     1  'SYSTEM PROFIT =',G13.6,//,30X,'SYSTEM RELIABILITY =',G13.6)    OPTI 500
 1600 FORMAT(1H0,/,30X,'CONSTRAINT(1) =',G13.6,//,30X,'CONSTRAINT(2) =',OPTI 510
     1  G13.6)                                                          OPTI 520
      END                                                               OPTI 530
```

COMPUTER LISTING FOR COMPUTATION SCHEME II

SYSTEM RELIABILITY OPTIMIZATION

```
C     MAIN PROGRAM                                                      MAIN  10
C     SYSTEM RELIABILITY OPTIMIZATION                                   MAIN  20
C     PARTIAL ENUMERATION METHOD                                        MAIN  30
      IMPLICIT INTEGER*4 (A-Z)                                          MAIN  40
      INTEGER*2 XTEST(40),XTEST1(40),XONE(40),XSTAR(40),XSTR1(40),      MAIN  50
     1  NSTAR1(40),MSTAR1(40),XHAT(40),NHAT(40),MHAT(40),NXTEST(40,5),  MAIN  60
     2  MXTEST(40,5),NXSTR1(40,5),MXSTR1(40,5)                          MAIN  70
      DIMENSION COUNT(4)                                                MAIN  80
      INTEGER*2 P,PP,PATH                                               MAIN  90
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  MAIN 100
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             MAIN 110
     2  PATH(300,50),NMOD,MPATH,PSIGN(300)                              MAIN 120
      COMMON/OPTM1/RKODE(40),NKODE(40),MKODE(40),NTEST(40),MTEST(40)    MAIN 130
      COMMON/OPTM2/R(40),G(40),GHAT(40),KINDEX(40),SINDEX(40),          MAIN 140
     1  CONSTR(40),COST,COSTHT,RLBTY,OBJFN,OBJHAT                       MAIN 150
      REAL*4 R,G,GHAT,KINDEX,SINDEX,CONSTR,COST,COSTHT,RLBTY,OBJFN,     MAIN 160
     1  OBJHAT                                                          MAIN 170
C                                                                       MAIN 180
      PLENTH = 1000                                                     MAIN 190
      PPL = 50                                                          MAIN 200
      PATHL = 300                                                       MAIN 210
   10 CALL DATAIN                                                       MAIN 220
      READ(5,1000) INNOD,OUTNOD                                         MAIN 230
      CALL PATHS(INNOD)                                                 MAIN 240
      IF(STATUS) 20,10,20                                               MAIN 250
   20 CALL PATHP(OUTNOD)                                                MAIN 260
      WRITE(6,1002)                                                     MAIN 270
      DO 30  L= 1,NPATH                                                 MAIN 280
      WRITE(6,1003) L,(PATH(L,M),M=1,NMOD  )                            MAIN 290
   30 CONTINUE                                                          MAIN 300
      CALL PATHC                                                        MAIN 310
      WRITE(6,1001) NPATH,MPATH                                         MAIN 320
      WRITE(6,1004)                                                     MAIN 330
      DO 35 L=1,MPATH                                                   MAIN 340
      WRITE(6,1005) L,PSIGN(L),(PATH(L,M),M=1,NMOD)                     MAIN 350
```

```
   35 CONTINUE                                                          MAIN 360
      OBJHAT = 1.0                                                      MAIN 370
      DO 1  I = 1,4                                                     MAIN 380
    1 COUNT(I) = 0                                                      MAIN 390
C     GENERATES TOTAL NUMBER OF VARIABLES IN TNE PROBLEM                MAIN 400
      NVAR = 0                                                          MAIN 410
      DO 40  I = 1,NMOD                                                 MAIN 420
      NVAR = NVAR + NKODE(I) + MKODE(I)                                 MAIN 430
   40 CONTINUE                                                          MAIN 440
C     GENERATES INITIAL VALUES OF XTEST(I)                              MAIN 450
      DO 50 I = 1,NVAR                                                  MAIN 460
      XTEST(I) = 0                                                      MAIN 470
      XONE(I) = 0                                                       MAIN 480
   50 CONTINUE                                                          MAIN 490
      XONE(1) = 1                                                       MAIN 500
      XTEST(1) = 1                                                      MAIN 510
C     GENERATE XSTAR                                                    MAIN 520
  100 CARRY = 0                                                         MAIN 530
      DO 140  I=1,NVAR                                                  MAIN 540
      XTEST1(I) = XTEST(I) + 1 + CARRY                                  MAIN 550
      IF (XTEST1(I).LE.1) GO TO 150                                     MAIN 560
      CARRY = 1                                                         MAIN 570
      XTEST1(I) = XTEST1(I) - 2                                         MAIN 580
      GO TO 140                                                         MAIN 590
  150 CARRY = 0                                                         MAIN 600
  140 CONTINUE                                                          MAIN 610
      TEST1 = 0                                                         MAIN 620
      DO 200 I=1,NVAR                                                   MAIN 630
      IF((XTEST(I).EQ.1) .OR. (XTEST1(I).EQ.1)) GO TO 210               MAIN 640
      XSTR1(I) = 0                                                      MAIN 650
      GO TO 200                                                         MAIN 660
  210 XSTR1(I) = 1                                                      MAIN 670
      TEST1 = TEST1 + XSTR1(I)                                          MAIN 680
  200 CONTINUE                                                          MAIN 690
      CARRY = 0                                                         MAIN 700
      DO 220 I=1,NVAR                                                   MAIN 710
```

```
      XSTAR(I) = XSTR1(I) + XONE(I) + CARRY                              MAIN 720
      IF(XSTAR(I).LE.1) GO TO 230                                        MAIN 730
      CARRY = 1                                                          MAIN 740
      XSTAR(I) = XSTAR(I) - 2                                            MAIN 750
      GO TO 220                                                          MAIN 760
  230 CARRY = 0                                                          MAIN 770
  220 CONTINUE                                                           MAIN 780
C     GENERATE VALUES FOR NTEST(I),MTEST(I),NSTAR1(I),AND MSTAR1(I)      MAIN 790
      K = 1                                                              MAIN 800
      DO 60  J = 1,5                                                     MAIN 810
      DO 70  I=1,NMOD                                                    MAIN 820
      IF (J.GT.NKODE(I)) GO TO 70                                        MAIN 830
      NXTEST(I,J) = XTEST(K)                                             MAIN 840
      NXSTR1(I,J) = XSTR1(K)                                             MAIN 850
      K = K + 1                                                          MAIN 860
      IF (J.GT.MKODE(I)) GO TO 70                                        MAIN 870
      MXTEST(I,J) = XTEST(K)                                             MAIN 880
      MXSTR1(I,J) = XSTR1(K)                                             MAIN 890
      K = K + 1                                                          MAIN 900
   70 CONTINUE                                                           MAIN 910
   60 CONTINUE                                                           MAIN 920
      DO 80  I=1,NMOD                                                    MAIN 930
      NTEST(I) = 1                                                       MAIN 940
      MTEST(I) = 1                                                       MAIN 950
      NSTAR1(I) = 1                                                      MAIN 960
      MSTAR1(I) = 1                                                      MAIN 970
      L = NKODE(I)                                                       MAIN 980
      IF (L .EQ. 0) GO TO 95                                             MAIN 990
      DO 90 J=1,L                                                        MAIN1000
      NTEST(I) = NTEST(I)  + (2**(J-1))*(1-NXTEST(I,J))                  MAIN1010
      NSTAR1(I) = NSTAR1(I) + (2**(J-1))*(1-NXSTR1(I,J))                 MAIN1020
   90 CONTINUE                                                           MAIN1030
   95 L = MKODE(I)                                                       MAIN1040
      IF (L .EQ. 0) GO TO 80                                             MAIN1050
      DO 110 J=1,L                                                       MAIN1060
      MTEST(I) = MTEST(I)  + (2**(J-1))*MXTEST(I,J)                      MAIN1070
```

```
      MSTAR1(I) = MSTAR1(I) + (2**(J-1))*MXSTR1(I,J)                    MAIN1080
  110 CONTINUE                                                          MAIN1090
   80 CONTINUE                                                          MAIN1100
C     TEST FOR GI1(X*-1) - GI2(X)                                       MAIN1110
      COUNT(1) = COUNT(1) + 1                                           MAIN1120
      DO 310 I=1,NMOD                                                   MAIN1130
      CONSTR(I) = NTEST(I) - MTEST(I)                                   MAIN1140
      IF(CONSTR(I).GE.0.0) GO TO 310                                    MAIN1150
      GO TO 400                                                         MAIN1160
  310 CONTINUE                                                          MAIN1170
      M = NMOD   + 1                                                    MAIN1180
      CONSTR(M) = 0.0                                                   MAIN1190
      DO 320 I=1,NMOD                                                   MAIN1200
  320 CONSTR(M) = CONSTR(M) + NSTAR1(I)*KINDEX(I)*(1./MSTAR1(I))**      MAIN1210
     1SINDEX(I)                                                         MAIN1220
      CONSTR(M) = COST - CONSTR(M)                                      MAIN1230
      IF(CONSTR(M).GE.0.0) GO TO 330                                    MAIN1240
      GO TO 400                                                         MAIN1250
C     TEST FOR GO(X) - GO(XHAT).                                        MAIN1260
  330 COUNT(2) = COUNT(2) + 1                                           MAIN1270
      CALL MODULE                                                       MAIN1280
      CALL RELIAB                                                       MAIN1290
      OBJFN = 1. - RLBTY                                                MAIN1300
      IF(OBJFN.LT.OBJHAT) GO TO 250                                     MAIN1310
  400 IF(TEST1.EQ.NVAR) GO TO 500                                       MAIN1320
      DO 260 I=1,NVAR                                                   MAIN1330
      XTEST(I) = XSTAR(I)                                               MAIN1340
  260 CONTINUE                                                          MAIN1350
      GO TO 100                                                         MAIN1360
C     TEST FOR GI1(X)-GI2(X)                                            MAIN1370
  250 COUNT(3) = COUNT(3) + 1                                           MAIN1380
      M = NMOD   + 1                                                    MAIN1390
      CONSTR(M) = 0.0                                                   MAIN1400
      DO 340 I=1,NMOD                                                   MAIN1410
  340 CONSTR(M) = CONSTR(M) + NTEST (I)*KINDEX(I)*(1./MTEST (I))**      MAIN1420
     1SINDEX(I)                                                         MAIN1430
```

```
      CONSTR(M) = COST - CONSTR(M)                                      MAIN1440
      IF(CONSTR(M).GE.0.0) GO TO 350                                    MAIN1450
      CARRY = 0                                                         MAIN1460
      TEST2 = 0                                                         MAIN1470
      DO 360 I=1,NVAR                                                   MAIN1480
      TEST2 = TEST2 + XTEST(I)                                          MAIN1490
      XTEST(I) = XTEST(I) + XONE(I) + CARRY                             MAIN1500
      IF(XTEST(I).LE.1) GO TO 370                                       MAIN1510
      CARRY = 1                                                         MAIN1520
      XTEST(I) = XTEST(I) - 2                                           MAIN1530
      GO TO 360                                                         MAIN1540
  370 CARRY = 0                                                         MAIN1550
  360 CONTINUE                                                          MAIN1560
      IF(TEST2.EQ.NVAR) GO TO 500                                       MAIN1570
      GO TO 100                                                         MAIN1580
  350 COUNT(4) = COUNT(4) + 1                                           MAIN1590
      DO 380 I=1,NVAR                                                   MAIN1600
  380 XHAT(I) = XTEST(I)                                                MAIN1610
      OBJHAT = OBJFN                                                    MAIN1620
      DO 390 I=1,NMOD                                                   MAIN1630
      NHAT(I) = NTEST(I)                                                MAIN1640
      MHAT(I) = MTEST(I)                                                MAIN1650
      GHAT(I) = G(I)                                                    MAIN1660
  390 CONTINUE                                                          MAIN1670
      M = NMOD   + 1                                                    MAIN1680
      COSTHT = COST - CONSTR(M)                                         MAIN1690
      GO TO 400                                                         MAIN1700
  500 RLBTY = 1. - OBJHAT                                               MAIN1710
      WRITE(6,1006)                                                     MAIN1720
      WRITE(6,1007)                                                     MAIN1730
      WRITE(6,1008)                                                     MAIN1740
      WRITE(6,1007)                                                     MAIN1750
      DO 410 I=1,NMOD                                                   MAIN1760
      WRITE(6,1009) I,NHAT(I),MHAT(I),GHAT(I)                           MAIN1770
      WRITE(6,1007)                                                     MAIN1780
  410 CONTINUE                                                          MAIN1790
```

```
      WRITE(6,1010) RLBTY,COSTHT,(J,COUNT(J),J=1,4)                     MAIN1800
      GO TO 10                                                          MAIN1810
 1000 FORMAT(2I3)                                                       MAIN1820
 1001 FORMAT(1H0,//,30X,'NPATH = ',I3,//,30X,'MPATH = ',I5)             MAIN1830
 1002 FORMAT(1H0,//,T20,'PATHS IN BINARY CODE',/)                       MAIN1840
 1003 FORMAT(1H0,T16,'PATH',I3,' =  ',50I2)                             MAIN1850
 1004 FORMAT(1H0,//,T20,'PATHS AND PATH COMBINATIONS IN BINARY CODE',/) MAIN1860
 1005 FORMAT(1H0,T16,'PATHC',I3,' = ',I2,4X,50I2)                       MAIN1870
 1006 FORMAT(1H1,//,T40,'THE OPTIMAL POINT IS',//)                      MAIN1880
 1007 FORMAT(1H+,T30,55('_'))                                           MAIN1890
 1008 FORMAT(1H ,T30,'|  MODULE  |','  N-OPTM  |','  M-OPTM  |',        MAIN1900
     1  ' MODULE RELIABILITY |')                                        MAIN1910
 1009 FORMAT(1H ,T30,3('|',4X,I2,4X),'|',D20.10,'|')                    MAIN1920
 1010 FORMAT(1H0,/,30X,'OPTIMIZED SYSTEM RELIABILITY = ',D20.10,//,30X, MAIN1930
     1  'SYSTEM COST = ',G13.5,//,(30X,'COUNT',I3,'  =',I8,//))        MAIN1940
      END                                                               MAIN1950
```

```
      SUBROUTINE DATAIN                                                 DATA  10
C     THIS SUBROUTINE READS AND ECHOCHECKS THE DATA                     DATA  20
      IMPLICIT INTEGER*4 (A-Z)                                          DATA  30
      INTEGER*2 P,PP,PATH                                               DATA  40
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  DATA  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             DATA  60
     2  PATH(300,50),NMOD,MPATH,PSIGN(300)                              DATA  70
      COMMON/OPTM1/RKODE(40),NKODE(40),MKODE(40),NTEST(40),MTEST(40)    DATA  80
      COMMON/OPTM2/R(40),G(40),GHAT(40),KINDEX(40),SINDEX(40),          DATA  90
     1  CONSTR(40),COST,COSTHT,RLBTY,OBJFN,OBJHAT                       DATA 100
      REAL*4 R,G,GHAT,KINDEX,SINDEX,CONSTR,COST,COSTHT,RLBTY,OBJFN,     DATA 110
     1  OBJHAT                                                          DATA 120
      DIMENSION TITLE(30)                                               DATA 130
C                                                                       DATA 140
      READ(5,999,END=70) ( TITLE(I),I=1,30)                             DATA 150
      READ(5,1000)NNODES,NMOD,COST                                      DATA 160
      DO 10 I=1,NMOD                                                    DATA 170
   10 READ(5,1006) RKODE(I),NKODE(I),MKODE(I),SINDEX(I),KINDEX(I),R(I)  DATA 180
      FOR=0                                                             DATA 190
      REV=0                                                             DATA 200
      DO 40 I=1,1000                                                    DATA 210
      READ(5,1001) P(I,1),P(I,2),P(I,4)                                 DATA 220
      IF( P(I,1)-P(I,2) ) 20,50,30                                      DATA 230
   20 FOR=FOR + 1                                                       DATA 240
      GO TO 40                                                          DATA 250
   30 REV=REV + 1                                                       DATA 260
   40 CONTINUE                                                          DATA 270
   50 NEDGES=FOR + REV                                                  DATA 280
      NEDGS1=NEDGES + 1                                                 DATA 290
      FREV=FOR + 1                                                      DATA 300
      WRITE(6,1002) ( TITLE(I),I=1,30)                                  DATA 310
      WRITE(6,1003)                                                     DATA 320
      WRITE(6,1004)                                                     DATA 330
      WRITE(6,1003)                                                     DATA 340
      DO 60 I=1,NEDGES                                                  DATA 350
```

```
      WRITE(6,1005) I,P(I,1),P(I,2),P(I,4)                              DATA 360
      WRITE(6,1003)                                                     DATA 370
   60 CONTINUE                                                          DATA 380
      WRITE(6,1007) COST                                                DATA 390
      WRITE(6,1008)                                                     DATA 400
      WRITE(6,1009)                                                     DATA 410
      WRITE(6,1010)                                                     DATA 420
      WRITE(6,1009)                                                     DATA 430
      DO 80 I=1,NMOD                                                    DATA 440
      WRITE(6,1011)I,RKODE(I),NKODE(I),MKODE(I),SINDEX(I),KINDEX(I),R(I)DATA 450
      WRITE(6,1009)                                                     DATA 460
   80 CONTINUE                                                          DATA 470
      RETURN                                                            DATA 480
   70 CALL EXIT                                                         DATA 490
  999 FORMAT(15A4)                                                      DATA 500
 1000 FORMAT(2I10,F10.0)                                                DATA 510
 1001 FORMAT(3I10)                                                      DATA 520
 1002 FORMAT(1H1,//,T10,30A4,//,T50,'INVENTORY OF BRANCHES',/)          DATA 530
 1003 FORMAT(1H+,T38,44('_'))                                           DATA 540
 1004 FORMAT (1H ,T38,'|  BRANCH  |','  ORIGIN  |','   END    |',       DATA 550
     1  '  MODULE  |')                                                  DATA 560
 1005 FORMAT (1H ,T38,4('|',4X,I2,4X),'|')                              DATA 570
 1006 FORMAT(3I10,3F10.0)                                               DATA 580
 1007 FORMAT(1H0,//,T40,'COST CONSTRAINT =',F15.3,//)                   DATA 590
 1008 FORMAT(1H0,T50,'INVENTORY OF MODULES',/)                          DATA 600
 1009 FORMAT(1H+,T20,83('_'))                                           DATA 610
 1010 FORMAT(1H ,T20,'|  MODULE  |','  R-KODE  |','  N-KODE  |',        DATA 620
     1  '  M-KODE  |','COST-COEFF|','  COST/UNIT  |','  UNIT-REL   |')  DATA 630
 1011 FORMAT(1H ,T20,4('|',4X,I2,4X),2('|',F10.3),3X,'|',G13.6,'|')     DATA 640
      END                                                               DATA 650
```

```
      SUBROUTINE MODULE                                                 MODU  10
C     THIS SUBROUTINE COMPUTES THE MODULE RELIABILITY.                  MODU  20
      IMPLICIT INTEGER*4 (A-Z)                                          MODU  30
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  MODU  40
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             MODU  50
     2  PATH(300,50),NMOD,MPATH,PSIGN(300)                              MODU  60
      INTEGER*2 P,PP,PATH                                               MODU  70
      COMMON/OPTM1/RKODE(40),NKODE(40),MKODE(40),NTEST(40),MTEST(40)    MODU  80
      COMMON/OPTM2/R(40),G(40),GHAT(40),KINDEX(40),SINDEX(40),          MODU  90
     1  CONSTR(40),COST,COSTHT,RLBTY,OBJFN,OBJHAT                       MODU 100
      REAL*4 R,G,GHAT,KINDEX,SINDEX,CONSTR,COST,COSTHT,RLBTY,OBJFN,     MODU 110
     1  OBJHAT                                                          MODU 120
      REAL*4 DUMMY1,DUMMY2,DUMMY3                                       MODU 130
C                                                                       MODU 140
      DO 200 K=1,NMOD                                                   MODU 150
      IF(RKODE(K) .EQ. 0) GO TO 100                                     MODU 160
      DUMMY1 = R(K) ** MTEST(K)                                         MODU 170
      DUMMY3 = - MTEST(K) * ALOG(R(K))                                  MODU 180
      G(K) = 1.0                                                        MODU 190
      DUMMY2 = 1.0                                                      MODU 200
      NM = NTEST(K) - MTEST(K)                                          MODU 210
      IF(NM .EQ. 0) GO TO 20                                            MODU 220
      DO 10 I=1,NM                                                      MODU 230
      DUMMY2 = DUMMY2 * DUMMY3/I                                        MODU 240
      G(K) = G(K) + DUMMY2                                              MODU 250
   10 CONTINUE                                                          MODU 260
   20 G(K) = G(K) * DUMMY1                                              MODU 270
      GO TO 200                                                         MODU 280
  100 DUMMY1 = R(K)                                                     MODU 290
      DUMMY2 = DUMMY1**NTEST(K)                                         MODU 300
      G(K) = DUMMY2                                                     MODU 310
      NM = NTEST(K) - MTEST(K)                                          MODU 320
      IF(NM .EQ. 0) GO TO 200                                           MODU 330
      DO 110 I=1,NM                                                     MODU 340
      DUMMY2 = DUMMY2 * ((1.-DUMMY1)/DUMMY1) * (NTEST(K)-I+1)/I         MODU 350
```

```
      G(K) = G(K) + DUMMY2                                              MODU 360
  110 CONTINUE                                                          MODU 370
  200 CONTINUE                                                          MODU 380
      RETURN                                                            MODU 390
      END                                                               MODU 400
```

```
      SUBROUTINE RELIAB                                                 RELI  10
C     THIS SUBROUTINE CALCULATES THE SYSTEM RELIABILITY FROM THE        RELI  20
C     MODULE RELIABILITIES.                                             RELI  30
      IMPLICIT INTEGER*4 (A-Z)                                          RELI  40
      COMMON/MASON/FOR,FREV,LEND,NEDGES,NEDGS1,NNODES,                  RELI  50
     1  NPATH,PATHL,PLENTH,PPL,REV,STATUS,P(1000,4),PP(50),             RELI  60
     2  PATH(300,50),NMOD,MPATH,PSIGN(300)                              RELI  70
      INTEGER*2 P,PP,PATH                                               RELI  80
      COMMON/OPTM1/RKODE(40),NKODE(40),MKODE(40),NTEST(40),MTEST(40)    RELI  90
      COMMON/OPTM2/R(40),G(40),GHAT(40),KINDEX(40),SINDEX(40),          RELI 100
     1  CONSTR(40),COST,COSTHT,RLBTY,OBJFN,OBJHAT                       RELI 110
      REAL*4 R,G,GHAT,KINDEX,SINDEX,CONSTR,COST,COSTHT,RLBTY,OBJFN,     RELI 120
     1  OBJHAT                                                          RELI 130
      REAL*4 GPATH                                                      RELI 140
C                                                                       RELI 150
      RLBTY = 0.0                                                       RELI 160
      DO 10 I=1,MPATH                                                   RELI 170
      GPATH = PSIGN(I)                                                  RELI 180
      DO 20 J=1,NMOD                                                    RELI 190
      IF (PATH(I,J) .EQ. 0) GO TO 20                                    RELI 200
      GPATH = GPATH * G(J)                                              RELI 210
   20 CONTINUE                                                          RELI 220
      RLBTY = RLBTY + GPATH                                             RELI 230
   10 CONTINUE                                                          RELI 240
      RETURN                                                            RELI 250
      END                                                               RELI 260
```