

OPTIMIZATION OF A CLASS OF NONLINEAR
RELIABILITY PROBLEMS USING A
STRUCTURE ANALYSIS ALGORITHM

by

Jeffery G. Wilkinson

ProQuest Number: 10796405

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10796405

Published by ProQuest LLC (2019). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Mineral Economics).

Golden, Colorado

Date March 29, 1989

Signed: _____


Jeffery G. Wilkinson

Approved: _____


Dr. R.E.D. Woolsey
Thesis Advisor

Golden, Colorado

Date March 29, 1989

Dr. John E. Tilton
Professor and Head,
Mineral Economics Department

ABSTRACT

Three major barriers frustrate the use of current technology to improve the design of existing systems: (1) the physical parameters (i.e., weight and volume) imposed by a subsystem on the entire system, (2) the economic penalties created by the costs of modifying the existing system to accommodate subsystem changes, and (3) the determination of the optimum structural design for the new system. As a result, design engineers must consider more than just the typical constraints of reliability and cost. Presently used methods for optimum design of redundant component systems require that the system structure be defined first. This approach assumes that we are optimizing the best structure. This assumption is suboptimal in that we may meet the desired level of reliability, but at a greatly increased cost in available resources. This study shows that we no longer must be bound by this often suboptimal assumption. A method is developed for solving this class of multiple constraint reliability problems. This study's algorithm solves for the optimal system structure and the specifications for the system given that structure. A computer program based on the algorithm is developed as part of the dissertation.

TABLE OF CONTENTS

	<u>Page</u>
ABSTRACT.	iii
LIST OF FIGURES	v
ACKNOWLEDGMENTS	vi
Chapter	
1 INTRODUCTION	1
2 CURRENT METHODS OF SOLVING CONVENTIONAL RELIABILITY PROBLEMS.	9
3 OPTIMIZATION AND INTERRELATION OF STANDARD DESIGN STRUCTURES	16
Optimization of the Series-Parallel System	17
Optimization of the Parallel-Series System	20
Relationship Between the Systems	24
4 OPTIMIZATION OF THE GENERAL RELIABILITY PROBLEM USING A PARALLEL-SERIES-PARALLEL STRUCTURE	26
5 EMPIRICAL RESULTS FROM THE APPLICATION OF THE ALGORITHM TO SLIGHTLY DISSIMILAR PROBLEMS.	36
REFERENCES CITED.	56
APPENDIX	
A Proof.	57
B Program Flowchart.	60
C Computer Program of the Algorithm.	61
D Sample Program Output.	74

LIST OF FIGURES

<u>Figure</u>		<u>Page</u>
1.1	Series-Parallel System	3
1.2	Parallel-Series System	4
4.1	Parallel-Series-Parallel System.	28

ACKNOWLEDGMENTS

It is with sincere appreciation that I acknowledge the motivational force of Dr. R.E.D. Woolsey. Throughout my graduate studies he has provided guidance and assistance in all of my endeavors. I have profited immeasurably from his wisdom and candor.

My wife, Rebecca and my daughter, Heather may never really know how much their support and understanding have meant to me. Without this rock to lean on this work would not have been completed or possibly even contemplated. Thank you both for your enduring love and understanding.

Chapter 1

INTRODUCTION

In the general field of systems design and modification, the goal is to design a system that meets or exceeds specific multiple design criteria. These criteria may include a required level of reliability, a budget ceiling, maximum space available for the system/subsystem or maximum weight of the system/subsystem. The aerospace industry is confronted with this type of problem on a recurring basis. The technology available for the design and improvement of existing systems is changing all of the time. There is great incentive to take advantage of the new technologies in order to remain competitive. However, one cannot replace an existing airframe in order to "fit in" a new subsystem. Therefore, the design of new systems must be constrained by the weight and volume availability dictated by the design of the original system. The purpose of this dissertation is to show that the more complex version of this problem, one that addresses an unknown initial system structure, can be solved using a combination of optimizing concepts.

For purposes of clarity some definitions of terms used throughout this text are presented. The least complex

item used in a system is a component. A component has weight, volume, cost, and a level of reliability. A module is one or more components connected in parallel. A subsystem is one or more modules connected in series. Finally, a system is one or more subsystems connected in parallel. For a system to function at least one component in each module of one of the parallel subsystems must be operative.

Conventional systems design that optimizes reliability by use of redundant components can take one of two structural approaches: series-parallel systems (Figure 1.1), and parallel-series systems (Figure 1.2). The structural approach used by the engineer is determined a-priori and the system is optimized within this design using known linear constraints.

The reliability portion of the design criteria is mathematically stated as a nonlinear function. This stems from the definition of reliability. Reliability is the probability that the system will continue functioning properly for a specific period of time. Probabilities are usually depicted as a fraction decimal between 0 and 1; therefore, it is also correct to define reliability as 1 minus the probability of failure ($1 - P[\text{failure}]$).

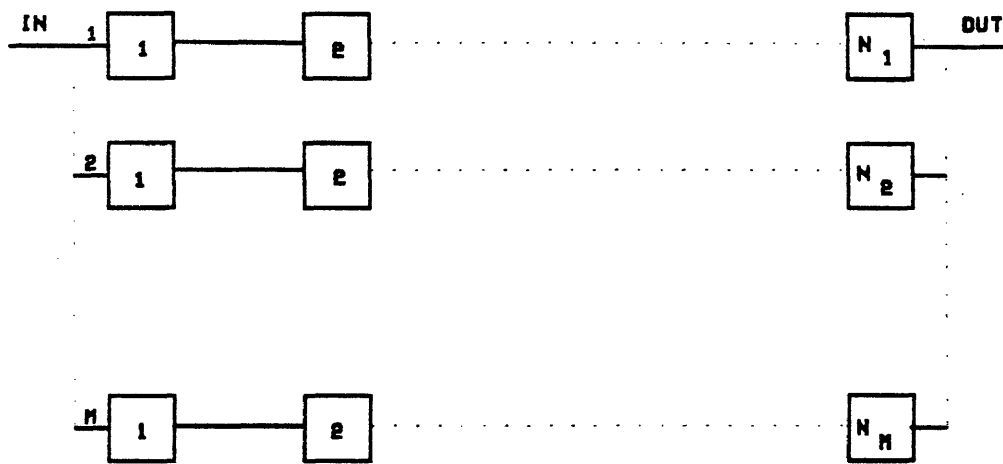


Figure 1.1
Series-Parallel System

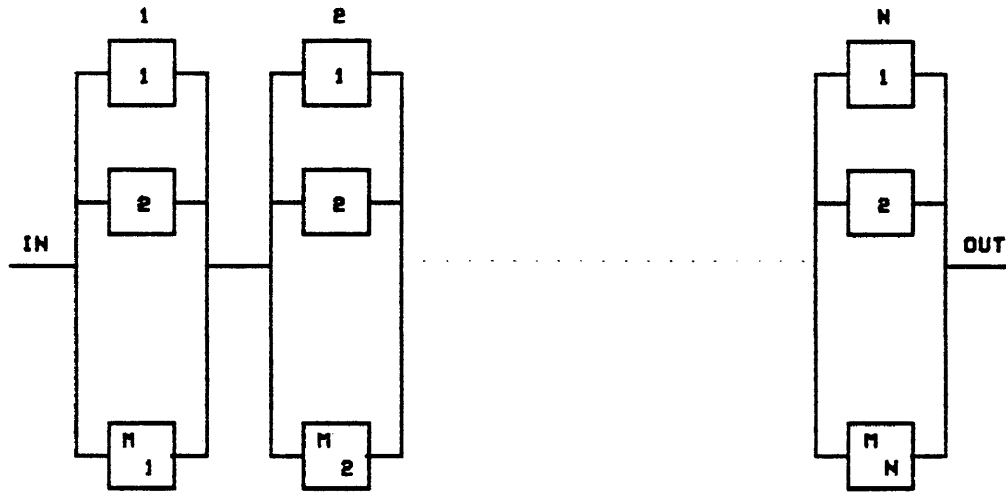


Figure 1.2
Parallel-Series System

If the system is comprised of multiple modules in series, then the system reliability is the product of the individual module reliabilities. Such a system is referred to as a parallel-series system. The mathematical relationship between the components connected in parallel is such that the module reliability becomes

$$1 - [P[\text{failure}]]^{\# \text{ of units.}}$$

Therefore, the multiplicative relationship between modules in series can be depicted in general terms as

$$(1 - P_1^{X_1})(1 - P_2^{X_2}) \dots (1 - P_n^{X_n}) = R$$

where

P_n is the probability of failure of component n ,
 R is the reliability of the system,
 X_i is the number of type i components.

If the system is comprised of two or more parallel subsystems, each having multiple single component modules in series, it is called a series-parallel system. The mathematical function that describes the reliability of this system is

$$1 - (1 - R_{\text{sub}})^M = R$$

where

R_{sub} is the reliability of each subsystem,
 M is the number of identical subsystems,
 R is the reliability of the system.

One can readily see the nonlinear structure of each of

these functions.

The budget, weight, and volume portions of the design criteria are usually stated as linear functions. The design criteria may also include weight and volume constraints on any subsystem.

The "cost" of each type component multiplied by the number used is the "cost" attributed to that component. The sum of the "costs" of the individual components equals the "cost" of the entire system. The general structure of the linear system design criteria is

$$c_1X_1 + c_2X_2 + \dots + c_nX_n = \text{COST.}$$

As will be discussed in detail later, the specific mathematical structure of the system design criteria is crucial to the success of the algorithm developed in this dissertation. An example problem, indicative of the type discussed, is shown below.

$$\begin{array}{ll}
 \text{Reliability: Total System Reliability} & \geq .90 \\
 \text{System Cost:} & \$5M_1 + \$7M_2 \leq 60 \\
 \text{System Weight:} & 4M_1 + 5M_2 \leq 70 \\
 \text{System Volume:} & 2M_1 + 4M_2 \leq 40 \\
 \text{Subsystem Weight:} & 4M_1 + 5M_2 \leq 19 \\
 \text{Subsystem Volume:} & 2M_1 + 4M_2 \leq 13 \quad (1.1)
 \end{array}$$

The algorithm developed in this dissertation approaches the multiple criteria design problem from a different

perspective than standard optimizing techniques. Conventional approaches assume a system structure and solve the problem in such manner that one criterion is optimized subject to the constraints imposed by the remaining criteria. The method depicted in this paper takes advantage of all of the information provided to the design engineer. Each criterion has a specific maximum or minimum threshold it must achieve. Rather than strictly viewing these values as resource constraints, this algorithm looks upon them as goals. Each of these goals must be reached to have a feasible solution. The nonlinear nature of this class of problems and the existence of specific design goals can be exploited to determine optimum system structure and specifications. Optimality is defined as the maximization of the reliability of the system composed of K identical subsystems.

Chapter 2 discusses some of the more robust techniques currently being used to solve the conventionally structured problems. The specifics of the approach taken by this algorithm is explained in Chapter 3 and Chapter 4. A computer program was developed based on this algorithm. Chapter 5 discusses the algorithm in depth and the results obtained when a sample set of problems was solved. Areas

that should be considered for further research are also presented in Chapter 5.

Chapter 2
CURRENT METHODS OF SOLVING CONVENTIONAL
RELIABILITY PROBLEMS

Reliability problems of the type described in Chapter 1 are not addressed explicitly in the literature. This may be due, in part, to the structural requirements of existing algorithms. Standard optimizing algorithms require that a problem be stated such that one is maximizing a reliability objective function subject to a set of constraints. The form of the objective function is determined by the structure of the system one is optimizing. This structure is determined prior to the formulation of the problem. If the answer that results is within the parameters of the required goals, it is accepted. If the result is not within the goals, the parameters must be changed, the problem is reformulated and solved again. This process repeats until an acceptable solution is found. There are numerous approaches to solving this form of the problem. Some of the more robust methods will be discussed here.

The method of Lagrange multipliers and the Kuhn-Tucker conditions can be used to solve a parallel-series problem of

the form

Maximize reliability

$$\prod_{j=1}^N (1 - P_j X_j)$$

Subject to

$$\sum_{j=1}^N a_{ij} X_j \leq b_i$$

where

$$i = 1, 2, \dots, r \quad (2.1)$$

The simultaneous equations formed by the Kuhn-Tucker conditions can be solved using Newton's method. These solutions are substituted into the original equations to obtain the optimal system reliability and the redundancies for each module in the parallel-series system. There are two negative attributes of this technique of which one must be aware. First, every solution set that satisfies the Kuhn-Tucker conditions will not always be a point that corresponds to the system maximum. Conversely, the point at which the objective function is at a maximum will always satisfy the Kuhn-Tucker conditions. Second, solutions obtained by this approach will not necessarily be integer. The accepted method is to round noninteger results down to the nearest integer. This approach may not result in the

optimum integer solution (Tillman, Hwang, and Kuo 1980).

Problems of the form shown above may also be solved using geometric programming (GP). GP exploits the fact that the arithmetic mean is at least as great as the geometric mean therefore permitting one to solve the dual rather than the primal problem (Federowicz and Mazumdar 1968). When solving the dual, one finds that this particular type of problem has one degree of difficulty. This means that one cannot solve explicitly for all of the dual variables. Nonetheless, all of the dual variables can be expressed in terms of one particular dual variable. Oatney (1987) showed that the value of this variable can be found by using the Brent-Decker method.

The solution for the dual is also the primal solution; for that reason, one is presented with the optimal value of the objective function prior to knowing the point of optimality. Geometric programming is more robust than the method of Lagrange multipliers for this type structure, because one is guaranteed that the solution is the optimal one. However, it suffers from one of the same drawbacks: the solution may not be integer and therefore when rounding to the nearest integer, an optimal solution cannot be guaranteed.

Wilkinson (1987) showed that the parallel-series system

can be solved for the integer optimal solution by using a technique that determines absolute bounds on the variables. This approach is used as a subroutine in this paper and is explained in great detail in chapter 3.

Dynamic programming is a technique for solving conventional reliability problems with both linear and nonlinear constraints. This technique shifts the view of the problem from an N-variable decision problem to N single-variable problems. The problem is structured into decision stages and each stage is solved sequentially (Tillman, Hwang, and Kuo 1980). In this case, the problem is to determine the number of redundant components at each module of an N-module series system.

The impact of all possible numbers of redundant components are evaluated at each stage of the dynamic program. The values considered at each stage are restricted by those considered in the previous stages and by the constraints on the problem. The result is an algorithm that enumerates through integer combinations and chooses that combination that optimizes the objective function. This technique does determine the optimal integer combination. However, it may require the examination of n^n combinations. Such a total enumeration would require an inordinate amount of computer time for even relatively small values of n.

Jensen (1970) used a dynamic programming approach to solve a broader class of problems called series-parallel-series networks. He used dynamic programming to solve the unconstrained version of the general problem. The algorithm was extended using dominance and a variety of elimination techniques to handle multiple constraints. His algorithm, like the others, requires that the engineer decide ahead of time on the system structure.

The system reliability problem being considered requires integer solutions; therefore, as might be expected, some applications of integer programming (IP) techniques to solve both conventional structures of this problem have been published. In 1968 Mizukami explored the use of convex programming to linearly approximate the nonlinear reliability function. His method used the simplex technique to solve the system of linear approximations and then Gomory's method of integer forms to arrive at an integer solution. In Ghare and Taylor's article they describe a method using Balas' branch and bound technique to also solve the parallel-series type problem. The use of IP techniques, however, may require complex transformations of the equations in order that the structure required by the particular algorithm be met. This is the case for both Gomory's cutting plane method and Balas's branch and bound

method (Tillman, Hwang, and Kuo 1980).

Both of these methods will (theoretically) solve problems of the type being considered, given that the structure is known. However, the nature of IP optimizing algorithms is such that convergence is guaranteed but the time required to converge may be extraordinary. A similar drawback is found in those IP algorithms that use some form of partial enumeration strategy. These algorithms may be forced to enumerate all possible combinations, thus resulting again in exponential computation time.

All of the strategies discussed above require that the reliability problem be formulated with a conventional objective function and constraints. Each of these approaches assume that the structure of the system is known to be either series-parallel or parallel-series. As will be shown in Chapter 3 such assumptions can lead to suboptimal use of available resources. The nonlinear optimization techniques provide an optimal noninteger solution quickly, but conventional methods of rounding to the nearest integer cannot guarantee optimality. The strategies that provide integer solutions usually require complex restructuring of the problem, inordinate amounts of computer time, or both. The method outlined in Chapters 3 and 4 require that the goal of each system and subsystem

constraint be determined a-priori, which is reasonable given the nature of real system design problems. This requirement permits one to determine the optimal structure of the system first and then, given that structure, the optimal system specifications.

Chapter 3
OPTIMIZATION AND INTERRELATION
OF STANDARD DESIGN STRUCTURES

Most conventional approaches to the problem of optimization of system reliability presuppose that the structure of the system is known a priori. When system reliability is to be improved through the addition of redundant subsystems the structure of the final system is always known. These systems are usually structured as either parallel-series or series-parallel systems (Tillman, Hwang, and Kuo 1980).

When the reliability problem is stated in the form of multiple criteria goals as shown in Chapter 1 then, in this paper, we shall show that one can determine the optimal system structure and derive absolute bounds on all system variables. We shall show that this is true regardless of which system structure is used. An efficient algorithm that modifies the bounds as it iterates can quickly search within these bounds to find the optimal integer solution and determine if this solution meets all of the design criteria (Wilkinson 1987).

The optimization method, given that the system

structure is already known for each of the conventional structures, is outlined below. In this paper, variables whose values are not known are annotated with a horizontal bar on top.

Optimization of the Series-Parallel System

The general form of the series-parallel system reliability goal is

$$1 - (1 - R_{s u b})^{\bar{K}} \geq R_g$$

where

$$\begin{aligned} R_{s u b} &= \text{reliability of each series subsystem} \\ \bar{K} &= \text{the unknown number of parallel} \\ &\quad \text{subsystems} \\ R_g &= \text{system reliability goal} \end{aligned} \quad (3.1)$$

The mathematical structure of this goal permits one to derive an absolute lower bound on the number of subsystems required to achieve the goal. The expression

$$(1 - R_{s u b})$$

is equal to the probability of failure of each subsystem

$$P_f$$

Therefore, by substitution, the new expression for the reliability of the system is

$$1 - P_f^{\bar{K}} \geq R_g$$

Because values for P_f and R_g are assumed to be known in all cases, one can use the above expression to solve for the minimum integer value of K that will satisfy this equation.

If

$$1 - P_f^{\bar{K}} \geq R_g$$

then

$$1 - R_g \geq P_f^{\bar{K}}$$

Take the natural logarithm of both sides:

$$\ln (1 - R_g) \geq \bar{K} \ln P_f$$

Solve for \bar{K} ($\ln P_f < 0.0$ since $0.0 < P_f < 1.0$)

$$\bar{K} \geq \frac{\ln (1 - R_g)}{\ln P_f}$$

\bar{K} is now the lower bound (K_{LB}) on the number of subsystems required to satisfy the reliability goal. If this minimum value for K_{LB} is not integer then one must round up to the nearest integer value to determine the minimum integer value that satisfies equation 3.1. Since, by definition, there is one component of each type in each subsystem of a series-parallel system, K_{LB} is also the minimum number of components of each type required to meet the reliability goal.

The constraining goals on this type of problem are linear in structure and can be used to determine the upper bounds on the system. In a series-parallel system the number of each type of component used in the system equals the number of subsystems, because all subsystems are the

same.

$$\bar{M}_1 = \bar{M}_2 = \dots = \bar{M}_n$$

Since all subsystems are the same, one can combine the cost, in resources, of components in a subsystem to determine the cost of each subsystem. The result is that the three constraining goals

$$\text{Cost: } c_1\bar{M}_1 + c_2\bar{M}_2 + \dots + c_n\bar{M}_n \leq C$$

$$\text{Weight: } w_1\bar{M}_1 + w_2\bar{M}_2 + \dots + w_n\bar{M}_n \leq W$$

$$\text{Volume: } v_1\bar{M}_1 + v_2\bar{M}_2 + \dots + v_n\bar{M}_n \leq V$$

where

- c_i = Cost per component i
- w_i = Weight per component i
- v_i = Volume per component i
- \bar{M}_i = Unknown number of component i
- C = Maximum budget available for system
- W = Maximum permitted weight of system
- V = Maximum permitted volume of system

become

$$\text{Cost: } \left(\sum_{i=1}^n c_i \right) \bar{M}_C \leq C$$

$$\text{Weight: } \left(\sum_{i=1}^n w_i \right) \bar{M}_W \leq W$$

$$\text{Volume: } \left(\sum_{i=1}^n v_i \right) \bar{M}_V \leq V$$

All of the variable values are known except for \bar{M}_C , \bar{M}_W , and \bar{M}_V . After substituting in the values that are known one can solve for these three variables. The result is an absolute upper bound on the number of subsystems that each type of

resource can support. These values must be rounded down to the nearest integer to provide an absolute integer bound. The absolute upper bound on the total system is the minimum of the set of values found:

$$K_{UB} = \min. (M_C, M_W, M_V)$$

If $K_{UB} < K_{LB}$ then the proposed system is infeasible. If $K_{UB} \geq K_{LB}$ then K_{UB} equals the number of subsystems that maximizes the reliability of the system within the constraining goals. However, this solution still may not achieve the required reliability threshold.

Optimization of the Parallel-Series System

The general form of the parallel-series system reliability goal is

$$\left[1 - P_{f1} \bar{M}_1\right] \left[1 - P_{f2} \bar{M}_2\right] \dots \left[1 - P_{fn} \bar{M}_n\right] \geq R_g$$

where

P_{fi} = probability of failure of component i
 \bar{M}_i = the unknown number of type i components
 R_g = system reliability goal

The multiplicative nature of this goal provides specific information on the required reliability of each subsystem. The total system cannot reach its goal unless the reliability of each module is also greater than or equal to R_g . The general expression that describes the

reliability of the j th module is

$$1 - P_{fj} \bar{M}_j$$

Therefore, the constraint on each module j is

$$1 - P_{fj} \bar{M}_j \geq R_g$$

Because values of P_{fj} and R_g are assumed to be known in all cases, one can use this relationship to solve explicitly for the minimum noninteger value of \bar{M}_j that will satisfy this equation.

If

$$1 - P_{fj} \bar{M}_j \geq R_g$$

then

$$1 - R_g \geq P_{fj} \bar{M}_j$$

Take the natural logarithm of both sides

$$\ln (1 - R_g) \geq \bar{M}_j \ln P_{fj}$$

solve for \bar{M}_j ($\ln P_{fj} < 0.0$ since $0.0 < P_{fj} < 1.0$)

$$\bar{M}_j \geq \frac{\ln (1 - R_g)}{\ln P_{fj}}$$

If the minimum value for \bar{M}_j is not integer then one must round up to the nearest integer value to determine the minimum integer value that satisfies the equation. Solving the minimum \bar{M}_j for all j will provide absolute lower bounds on all variables.

The structure of the linear goals and the lower bounds determined above can be exploited to establish specific

integer upper bounds on each variable. The general form of each of the three linear goals is the same as that for the conventional series-parallel system explained above.

$$\text{Cost: } c_1 \bar{M}_1 + c_2 \bar{M}_2 + \dots + c_n \bar{M}_n \leq C$$

$$\text{Weight: } w_1 \bar{M}_1 + w_2 \bar{M}_2 + \dots + w_n \bar{M}_n \leq W$$

$$\text{Volume: } v_1 \bar{M}_1 + v_2 \bar{M}_2 + \dots + v_n \bar{M}_n \leq V$$

One can solve for the upper bound on \bar{M}_i (\bar{M}_{UBi}), with respect to each resource constraint, by substituting the lower bounds, M_{LBj} for all $j \neq i$, into the goal and solving for the maximum \bar{M}_i that satisfies the equation. The upper bound on \bar{M}_1 (\bar{M}_{UB1}) is determined in the following manner.

If

$$c_1 \bar{M}_{UB1} + c_2 \bar{M}_2 + \dots + c_n \bar{M}_n \leq C$$

substitute in M_{LBj} for \bar{M}_j , for all $j \neq 1$

$$c_1 \bar{M}_{UB1} + c_2 M_{LB2} + \dots + c_n M_{LBn} \leq C$$

subtract $c_j M_{LBj}$, for all $j \neq 1$, from both sides

$$c_1 \bar{M}_{UB1} \leq C - c_2 M_{LB2} - \dots - c_n M_{LBn}$$

divide both sides by c_1

$$\bar{M}_{UB1} \leq \frac{C - c_2 M_{LB2} - \dots - c_n M_{LBn}}{c_1}$$

The upper bound on M_1 with respect to goal 2 is shown by this equation. To determine an integer upper bound on M_1 with respect to goal 2, one must round M_1 down to the nearest integer value. Applying this logic to the other two

goals will produce a set containing the upper bound on M_1 . The minimum value within this set is the absolute upper bound on M_1 .

Once the integer bounds on all of the variables have been computed, a total enumeration of all possible combinations will reveal the optimal solution. This optimal solution is not guaranteed to be feasible. The solution must be checked to see if it has violated the reliability goal. This approach, however, is far more time consuming than necessary. A more efficient approach is one that searches the combinations by looping through nested variables. The objective is to maximize reliability; therefore, the search will start at the upper bounds on each of the variables. If a particular combination violates the reliability goal or has a reliability that is less than the previous best feasible solution, then decrementing the most deeply nested variable will only cause a greater violation. Therefore, when either situation occurs, the decrementation of the innermost variable is stopped and the next variable is decremented by one and the process resumes.

The efficiency of the search can be increased by monitoring the current best feasible solution (CBFS). When the reliability of the CBFS is sufficiently greater than the last best solution then the search halts and new variable

bounds are computed. The search then continues with the new tighter variable bounds. The result is the integer solution that maximizes the reliability of the system. As stated before, this solution is not guaranteed to be feasible.

Relationship Between the Systems

The Parallel-Series and Series-Parallel systems are different structural approaches to solving the same general problem. Each increases the reliability of a system by use of redundancy. In the case of the Parallel-Series system the duplication occurs at the individual component level. One or more duplicate components are connected in parallel to the initial component, thus providing immediate backup in the case of component failure.

In the Series-Parallel system the redundancy occurs at the system level. One or more duplicates of the total system are connected in parallel to the initial system. This provides an immediate backup system in the event that the initial system fails.

It has been rigorously proven that a parallel-series system, having the exact same number and types of components as a series-parallel system, will have a level of reliability that is greater than or equal to that of the series-parallel system (Appendix A). When there is no

component redundancy in the design these two structures are identical; thus, their levels of reliability are equal. In all other cases, that meet the initial criteria, the parallel-series system will have a higher level of reliability.

This relationship becomes self-evident when one considers the following example. Given two systems: one a parallel-series system and the other a series-parallel system. The initial system, prior to the addition of redundant components or systems, consists of five different components in series. If four additional components of each type are added to each example system the total number of components in each system is twenty-five. The series-parallel system will fail if just one of any type component fails in each subsystem. The parallel-series system will not fail until all of the components of a particular type fail. It is obvious that in the absence of any specific constraints one should choose a parallel-series type of design. Chapter 4 will discuss the optimization of structure when constraints that prohibit a strict parallel-series design are introduced into the problem.

Chapter 4

OPTIMIZATION OF THE GENERAL RELIABILITY PROBLEM
USING A PARALLEL-SERIES-PARALLEL STRUCTURE

The general form of the reliability problem, as described in Chapter 1, does not presuppose a particular conventional system structure. The objective is to define the system structure and specification that maximizes the reliability of the system, subject to meeting all of the design criteria.

The system design criteria consist of three resource constraints, two constraints on subsystem size, and a minimum required level of total system reliability. The constraining criteria are all linear in structure. The structure of the reliability criteria is not specified a priori. The general form of the problem is

Maximize \bar{R}_s

Subject to

Reliability: Total System Reliability $\geq R_g$

System Cost: $c_1\bar{M}_1 + c_2\bar{M}_2 + \dots + c_n\bar{M}_n \leq C$

System Weight: $w_1\bar{M}_1 + w_2\bar{M}_2 + \dots + w_n\bar{M}_n \leq W_{TS}$

System Volume: $v_1\bar{M}_1 + v_2\bar{M}_2 + \dots + v_n\bar{M}_n \leq V_{TS}$

Subsystem Weight: $w_1\bar{M}_{1j} + w_2\bar{M}_{2j} + \dots + w_n\bar{M}_{nj} \leq W_{SS}$

Subsystem Volume: $v_1\bar{M}_{1j} + v_2\bar{M}_{2j} + \dots + v_n\bar{M}_{nj} \leq V_{SS}$

where

\bar{R}_S = Total system reliability
 R_g = Minimum required level of reliability
 c_i = Cost per item i
 w_i = Weight per item i
 v_i = Volume per item i
 \bar{M}_i = Unknown total number of item i
 \bar{M}_{ij} = Unknown number of item i in subsystem j
 C = Maximum budget available for system
 W_{TS} = Maximum permitted weight of system
 V_{TS} = Maximum permitted volume of system
 W_{SS} = Maximum permitted weight of each subsystem
 V_{SS} = Maximum permitted volume of each subsystem

The reliability function is not specified in the general form of the problem because one cannot necessarily be certain which structure, series-parallel or parallel-series, is optimum for the system. However, one can specify a structure that encompasses both series-parallel systems and parallel-series systems. This composite structure is the parallel-series-parallel system (PSPS) (Figure 4.1). The reliability function for this system is

$$\bar{R}_S = 1 - [1 - \prod_{j=1}^N 1 - (1 - R_j)^{\bar{M}_j}]^{\bar{K}}$$

where

\bar{R}_S = Unknown total system reliability
 R_j = Reliability of the jth type component
 N = Number of different types of components in series in each of the subsystems
 \bar{M}_j = Unknown number of components in the jth module of each of the \bar{K} subsystems
 \bar{K} = Unknown number of identical parallel subsystems

(4.1)

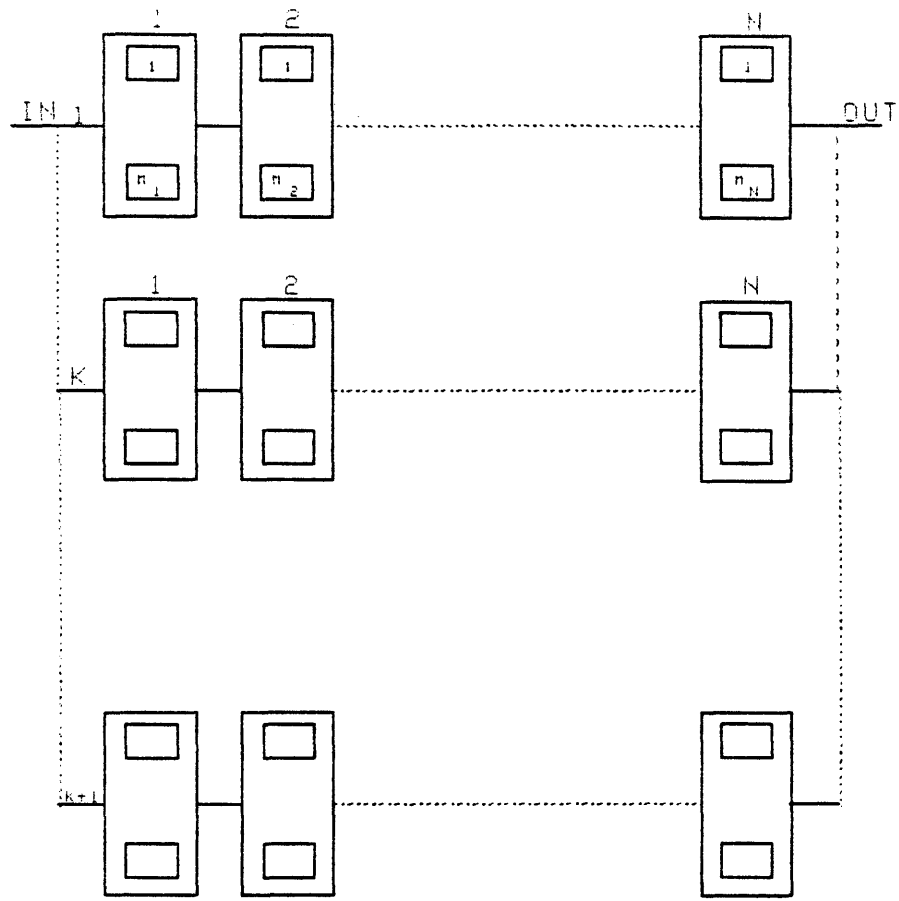


Figure 4.1
 Parallel-Series-Parallel System

A PSPS with a \bar{k} value of one is a parallel-series system. This can be shown by setting \bar{k} equal to 1 in equation 4.1 and simplifying. The result is equation 3.2, the reliability function for a parallel-series system. This substitution and simplification are shown below.

$$\bar{R}_S = 1 - \left[1 - \prod_{j=1}^N 1 - (1 - R_j)^{\bar{M}_j} \right]^1$$

equals

$$\bar{R}_S = 1 - \left[1 - \prod_{j=1}^N 1 - (1 - R_j)^{\bar{M}_j} \right]$$

Substitute P_{fj} (component probability of failure) for $(1 - R_j)$

$$\bar{R}_S = 1 - \left[1 - \prod_{j=1}^N 1 - P_{fj}^{\bar{M}_j} \right]$$

Expand the function

$$\bar{R}_S = 1 - \{ 1 - [(1 - P_{f1}^{\bar{M}_1})(1 - P_{f2}^{\bar{M}_2}) \dots (1 - P_{fN}^{\bar{M}_N})] \}$$

and simplify

$$\bar{R}_S = (1 - P_{f1}^{\bar{M}_1}) (1 - P_{f2}^{\bar{M}_2}) \dots (1 - P_{fN}^{\bar{M}_N})$$

This is equation 3.2.

Equivalently, a PSPS reliability function with \bar{M}_j equal to one for all values of j is the same as equation 3.1, the reliability function for a series-parallel system. This relationship is derived in the following manner.

$$\bar{R}_S = 1 - \left[1 - \prod_{j=1}^N 1 - (1 - R_j)^1 \right]^{\bar{k}}$$

equals

$$\bar{R}_s = 1 - [1 - \prod_{j=1}^N 1 - (1 - R_j)]^{\bar{K}}$$

Substituting R_{sub} for $\prod_{j=1}^N 1 - (1 - R_j)$

$$\bar{R}_s = 1 - (1 - R_{sub})^{\bar{K}}$$

This is equation 3.1.

It has been shown above that when appropriate the PSPS model can determine that the optimal structure is a strict series-parallel or parallel-series system. Given a particular set of design criteria neither of these conventional structures may be optimal and the PSPS model will show that the optimal structure is a PSPS. The algorithm for optimizing the PSPS is the same regardless of the resultant structure.

The goal is to optimize the reliability of the system subject to the specified design criteria. Only solutions that meet or exceed the minimum level of reliability are of any value. Therefore the PSPS reliability function value must be greater than or equal to the reliability goal.

$$1 - [1 - \prod_{j=1}^N 1 - (1 - R_j)^{\bar{M}_j}]^{\bar{K}} \geq R_g$$

Multiply both sides by -1 and add 1 to both sides:

$$[1 - \prod_{j=1}^N 1 - (1 - R_j)^{\bar{M}_j}]^{\bar{K}} \leq 1 - R_g$$

Take the \bar{k} th root of both sides:

$$1 - \prod_{j=1}^N 1 - (1 - R_j)^{\bar{M}_j} \leq \sqrt[\bar{k}]{1 - R_g}$$

Multiply both sides by -1 and adding 1 to both sides:

$$\prod_{j=1}^N 1 - (1 - R_j)^{\bar{M}_j} \geq 1 - \sqrt[\bar{k}]{1 - R_g} \quad (4.2)$$

In equation 4.2 there are only two unknowns: \bar{k} , which is the number of identical parallel subsystems, and \bar{M}_j for all j , which is the number of identical parallel components in each of the j modules. The optimum value of \bar{k} can be determined by examining the relationship between the subsystem constraints and the total system resource constraints.

The right hand side of the two subsystem constraints,

$$\text{Subsystem Weight: } w_1 \bar{M}_{1j} + w_2 \bar{M}_{2j} + \dots + w_n \bar{M}_{nj} \leq W_{SS}$$

$$\text{Subsystem Volume: } v_1 \bar{M}_{1j} + v_2 \bar{M}_{2j} + \dots + v_n \bar{M}_{nj} \leq V_{SS}$$

W_{SS} and V_{SS} , dictate the maximum amount of each resource that can be used in a subsystem. The right hand side of the corresponding system constraints,

$$\text{System Weight: } w_1 \bar{M}_1 + w_2 \bar{M}_2 + \dots + w_n \bar{M}_n \leq W_{TS}$$

$$\text{System Volume: } v_1 \bar{M}_1 + v_2 \bar{M}_2 + \dots + v_n \bar{M}_n \leq V_{TS}$$

W_{TS} and V_{TS} , dictate the amount of each resource available for the total system. The ratios (W_{TS}/W_{SS}) and (V_{TS}/V_{SS}) are noninteger upper bounds on the number of subsystems each

resource can support. The exact number of identical optimal parallel subsystems each resource can support is found by rounding each of the ratios down to the nearest integer.

The minimum of these two values

$$[\text{truncate}(W_{TS}/W_{SS}), \text{truncate}(V_{TS}/V_{SS})]$$

is the optimal value for \bar{K} , i.e. K^* .

Substituting K^* for \bar{K} in equation 4.2

$$\prod_{j=1}^N 1 - (1 - R_j)^{\bar{M}_j} \geq 1 - \sqrt[K^*]{1 - R_g} \quad (4.3)$$

All of the variables on the right hand side of equation 4.3 are known. Therefore a constant, R^* , where

$$R^* = 1 - \sqrt[K^*]{1 - R_g}$$

can be substituted into equation 4.3 resulting in

$$\prod_{j=1}^N 1 - (1 - R_j)^{\bar{M}_j} \geq R^* \quad (4.4)$$

R^* is the minimum level of reliability that each of the K^* parallel subsystems must meet to ensure that the PSPS meets the reliability goal. Equation 4.4 is now the reliability goal for each of the K^* subsystems in the PSPS.

The problem can now be formulated as an optimization of a parallel-series subsystem. The reliability criteria is as shown in equation 4.4. The subsystem weight and volume constraints are now directly applicable. The right hand side of the system cost constraint is divided by K^* thus

providing a cost constraint for the K^* identical subsystems.

$$\text{Subsystem Cost: } c_1 \bar{M}_1 + c_2 \bar{M}_2 + \dots + c_n \bar{M}_n \leq C/K^*$$

The subsystem problem is of the form

Maximize \bar{R}_S

Subject to

$$\text{Reliability Goal: } \prod_{j=1}^N 1 - (1 - R_j)^{\bar{M}_j} \geq R^*$$

$$\begin{aligned} \text{Subsystem Cost: } & c_1 \bar{M}_1 + c_2 \bar{M}_2 + \dots \\ & + c_n \bar{M}_n \leq C/K^* \end{aligned}$$

$$\begin{aligned} \text{Subsystem Weight: } & w_1 \bar{M}_{1j} + w_2 \bar{M}_{2j} + \dots \\ & + w_n \bar{M}_{nj} \leq W_{SS} \end{aligned}$$

$$\begin{aligned} \text{Subsystem Volume: } & v_1 \bar{M}_{1j} + v_2 \bar{M}_{2j} + \dots \\ & + v_n \bar{M}_{nj} \leq V_{SS} \quad (4.5) \end{aligned}$$

This problem can be optimized in the same manner as that described for the conventional parallel-series system in Chapter 3. If a solution is feasible, the solution of this new problem is the optimum specification for each of the K^* identical parallel subsystems in the PSPS.

At this point the current PSPS specification is optimal from the perspective of maximizing the reliability of the K^* subsystems. External design considerations may require that all of the subsystems be identical. Stopping at this point fulfills such a requirement. There may, however, be resources remaining that can be used to further enhance

system reliability. The strategy one uses to take advantage of the remaining resources is based on considerations beyond the scope of this work. These considerations can include: a desire to construct one more optimal subsystem using available resources, a need to have all subsystems identical, or a desire to minimize the absolute differences between subsystems used, while permitting two or more types of subsystems. In this work, the focus is to construct the optimal subsystem using the remaining resources.

The final step in the algorithm ensures an optimal use of any remaining resources. The resources used to build the K^* identical parallel subsystems are subtracted from the total amount of resources available. The remaining resources become the right hand side of each respective resource constraint. The right hand side of the reliability criteria function becomes the level of reliability associated with a series system composed of a single component of each type.

This final subsystem problem is constructed as a conventional parallel-series system. The result becomes the K^*+1 parallel subsystem in the PSPS. The design structure and specification now reflect the optimal solution, given a desire to maximize the number of identical parallel subsystems. The available resources have been divided

appropriately into subsystem resources and optimal subsystems have been designed. The parallel linking of these optimum subsystems results in the optimum design for the entire system.

Chapter 5 reports on the empirical results of the application of this algorithm to reliability problems whose optimum structures are series-parallel, parallel-series, and parallel-series-parallel.

Chapter 5
 EMPIRICAL RESULTS FROM THE APPLICATION
 OF THE ALGORITHM TO SLIGHTLY
 DISSIMILAR PROBLEMS

For explanatory purposes a specific problem, of the form shown in chapter 4, has been devised to empirically show the capability of this algorithm. This problem is limited to three types of components to prevent a plethora of detail detrimental to the discussion of the algorithm. The example problem is

$$\text{Maximize } \bar{R}_S = 1 - [1 - \prod_{i=1}^N 1 - (1 - R_i)^{\bar{M}_{ij}}] \bar{K}$$

Subject to

$$\text{Reliability: Total System Reliability} \geq .95$$

$$\text{System Cost: } \$1.2\bar{M}_1 + \$2.3\bar{M}_2 + \$3.4\bar{M}_3 \leq C$$

$$\text{System Weight: } 5\bar{M}_1 + 4\bar{M}_2 + 8\bar{M}_3 \leq W_{TS}$$

$$\text{System Volume: } 10\bar{M}_1 + 15\bar{M}_2 + 34\bar{M}_3 \leq V_{TS}$$

$$\text{Subsystem Weight: } 5\bar{M}_{1j} + 4\bar{M}_{2j} + 8\bar{M}_{3j} \leq W_{SS}$$

$$\text{Subsystem Volume: } 10\bar{M}_{1j} + 15\bar{M}_{2j} + 34\bar{M}_{3j} \leq V_{SS}$$

$$R_1 = \text{Reliability of Component 1: } .80$$

$$R_2 = \text{Reliability of Component 2: } .70$$

$$R_3 = \text{Reliability of Component 3: } .75$$

where

\bar{R}_s = Total system reliability
 \bar{K} = Unknown number of parallel subsystems
 N = Number of types of components
 R_i = Reliability of the i th type component
 \bar{M}_i = Unknown total number of component i
 \bar{M}_{ij} = Unknown number of component i in subsystem j
 C = Maximum budget available for system
 W_{TS} = Maximum permitted weight of system
 V_{TS} = Maximum permitted volume of system
 W_{SS} = Maximum permitted weight of each subsystem
 V_{SS} = Maximum permitted volume of each subsystem

This general problem will be used to demonstrate that with differing resource constraints the optimal system structure and design specification will change. The algorithm will determine the optimum system structure and design for each case.

In the first example the resource constraining values are

$$\begin{aligned}
 C &= \$45.00 \\
 W_{TS} &= 115.00 \\
 V_{TS} &= 475.00 \\
 W_{SS} &= 19.00 \\
 V_{SS} &= 65.00
 \end{aligned}$$

The problem that results is

$$\text{Maximize } 1 - \left[1 - \prod_{i=1}^N 1 - (1 - R_i)^{\bar{M}_{ij}} \right]^{\bar{K}}$$

Subject to

$$\begin{aligned}
 \text{Reliability:} \quad & \text{Total System Reliability} \geq .95 \\
 \text{System Cost:} \quad & \$1.2\bar{M}_1 + \$2.3\bar{M}_2 + \$3.4\bar{M}_3 \leq \$45.00 \\
 \text{System Weight:} \quad & 5\bar{M}_1 + 4\bar{M}_2 + 8\bar{M}_3 \leq 115.00 \\
 \text{System Volume:} \quad & 10\bar{M}_1 + 15\bar{M}_2 + 34\bar{M}_3 \leq 475.00 \\
 \text{Subsystem Weight:} \quad & 5\bar{M}_{1j} + 4\bar{M}_{2j} + 8\bar{M}_{3j} \leq 19.00 \\
 \text{Subsystem Volume:} \quad & 10\bar{M}_{1j} + 15\bar{M}_{2j} + 34\bar{M}_{3j} \leq 65.00 \\
 R_1 = & 0.80 \\
 R_2 = & 0.70 \\
 R_3 = & 0.75
 \end{aligned}$$

We know from equation 4.2 that the objective function reduces to

$$\prod_{i=1}^N 1 - (1 - R_i)^{\bar{M}_{i,j}} \geq 1 - \sqrt[\bar{K}]{1 - .95}$$

The optimum value for \bar{K} is the minimum of

$$[\text{truncate}(W_{TS}/W_{SS}), \text{truncate}(V_{TS}/V_{SS})]$$

substituting in the values from the example problem

$$[\text{truncate}(115.00/19.00), \text{truncate}(475.00/65.00)]$$

or

$$(6.00, 7.00)$$

The minimum value is 6.00 therefore

$$K^* = 6.00$$

The optimal number of identical parallel subsystems is six.

Substituting K^* for \bar{K}

$$\prod_{i=1}^N 1 - (1 - R_i)^{\bar{M}_i} \geq 1 - \sqrt[6]{1 - .95}$$

Reduce the right hand side

$$\prod_{i=1}^N 1 - (1 - R_i)^{\bar{M}_i} \geq .39 \quad (5.1)$$

Equation 5.1 is the reliability goal for each of the identical parallel subsystems. The budget constraint for this parallel-series subsystem is

$$\text{Subsystem Cost: } \$1.2\bar{M}_1 + \$2.3\bar{M}_2 + \$3.4\bar{M}_3 \leq (\$45.00/K^*)$$

Therefore the resource constraints are

$$\text{Subsystem Cost: } \$1.2\bar{M}_1 + \$2.3\bar{M}_2 + \$3.4\bar{M}_3 \leq \$7.50$$

$$\text{Subsystem Weight: } 5\bar{M}_{1j} + 4\bar{M}_{2j} + 8\bar{M}_{3j} \leq 19.00$$

$$\text{Subsystem Volume: } 10\bar{M}_{1j} + 15\bar{M}_{2j} + 34\bar{M}_{3j} \leq 65.00$$

Expand equation 5.1 and substitute in the component reliabilities

$$[1 - (1 - .80)^{\bar{M}_1}] \cdot [1 - (1 - .70)^{\bar{M}_2}] \cdot [1 - (1 - .75)^{\bar{M}_3}] \geq .39$$

Simplify the equation

$$(1 - .20^{\bar{M}_1}) \cdot (1 - .30^{\bar{M}_2}) \cdot (1 - .25^{\bar{M}_3}) \geq .39$$

We know that in order to meet the reliability goal the value of each term in the left hand side of this equation must be greater than or equal to the right hand side value, .39.

The lower bound on \bar{M}_i can be found explicitly by

establishing this relationship and solving for \bar{M}_i . The lower bound on \bar{M}_1 is found by setting

$$(1 - .20^{\bar{M}_1}) \geq .39$$

Solve for \bar{M}_1

$$\bar{M}_1 \geq \frac{\ln(1-.39)}{\ln .20}$$

The noninteger lower bound on \bar{M}_1 is 0.31. The integer lower bound is found by rounding up to the nearest integer, 1.0.

Therefore

$$\bar{M}_{1LB} = 1.0$$

Similarly one can solve for \bar{M}_{2LB} and \bar{M}_{3LB}

$$\bar{M}_{2LB} = 0.41$$

$$\bar{M}_{3LB} = 0.36$$

The integer lower bound on all of the variables is 1.0.

The upper bound on variable \bar{M}_i is found by substituting \bar{M}_{jLB} for all $j \neq i$ into each of the resource constraints and solving for \bar{M}_{iUB} for each constraint. The minimum value of \bar{M}_{iUB} for each constraint is the noninteger upper bound on that variable. Rounding down to the nearest integer provides the integer upper bound on that variable.

In this example problem we can substitute \bar{M}_{2LB} and \bar{M}_{3LB} into each of the resource constraints

$$\text{Subsystem Cost: } \$1.2\bar{M}_{1UB} + \$2.3(1.0) + \$3.4(1.0) \leq \$7.50$$

$$\text{Subsystem Weight: } 5\bar{M}_{1UB} + 4(1.0) + 8(1.0) \leq 19.00$$

$$\text{Subsystem Volume: } 10\bar{M}_{1UB} + 15(1.0) + 34(1.0) \leq 65.00$$

Solve for \bar{M}_{1UB} with respect to each constraint

$$\text{Cost: } \bar{M}_{1UB} = 1.50$$

$$\text{Weight: } \bar{M}_{1UB} = 1.40$$

$$\text{Volume: } \bar{M}_{1UB} = 1.60$$

The minimum value is 1.40. Rounding this value down to the nearest integer results in an integer upper bound of 1.0 for \bar{M}_1 . Similarly we can solve for the upper bounds on each of the other variables.

$$\bar{M}_{2UB} = 1.0$$

$$\bar{M}_{3UB} = 1.0$$

In this particular case the upper and lower bounds on all variables equal 1.0. The optimum subsystem must be one with a single component of each type in series. In this unique situation, the only possible configuration is a strict series-parallel system. We know that the optimum system will contain K^* , 6.0, of these systems in parallel.

The final step in determining the optimum specification for this system is to compute the remaining resources and optimize a final series subsystem using the remaining resources. In our example problem the remaining resources

are

Cost: \$3.60

Weight: 13.00

Volume: 121.00

The minimum required resources for a series system having one component of each type are

Cost: \$6.90

Weight: 17.00

Volume: 59.00

There are insufficient resources to support an additional subsystem therefore the six identical parallel subsystems designed above constitute the optimal system. The reliability of this system is

$$1 - [1 - \prod_{i=1}^3 1 - (1 - R_i)^1]^6 = 0.9619$$

The structure of this optimum system is equivalent to a conventional series-parallel system. There are six identical series subsystems in parallel. None of the modules within each subsystem contain any redundant components in parallel.

If we modify the right hand side values of the Weight and Volume constraints on the subsystems to

$$W_{SS} = 105.00$$

$$V_{SS} = 400.00$$

we create a problem with quite different characteristics.

The new problem is

$$\text{Maximize } 1 - [1 - \prod_{i=1}^N 1 - (1 - R_i)^{\bar{M}_{i,j}}]^{\bar{K}}$$

Subject to

$$\text{Reliability: Total System Reliability} \geq .95$$

$$\text{System Cost: } \$1.2\bar{M}_1 + \$2.3\bar{M}_2 + \$3.4\bar{M}_3 \leq \$45.00$$

$$\text{System Weight: } 5\bar{M}_1 + 4\bar{M}_2 + 8\bar{M}_3 \leq 115.00$$

$$\text{System Volume: } 10\bar{M}_1 + 15\bar{M}_2 + 34\bar{M}_3 \leq 475.00$$

$$\text{Subsystem Weight: } 5\bar{M}_{1j} + 4\bar{M}_{2j} + 8\bar{M}_{3j} \leq 105.00$$

$$\text{Subsystem Volume: } 10\bar{M}_{1j} + 15\bar{M}_{2j} + 34\bar{M}_{3j} \leq 400.00$$

$$R_1 = 0.80$$

$$R_2 = 0.70$$

$$R_3 = 0.75$$

We know from equation 4.2 that the objective function

reduces to

$$\prod_{i=1}^N 1 - (1 - R_i)^{\bar{M}_{i,j}} \geq 1 - \sqrt[\bar{K}]{1 - .95}$$

and

$$\bar{K} = \min[\text{truncate}(W_{TS}/W_{SS}), \text{truncate}(V_{TS}/V_{SS})]$$

or

$$\min(1.00, 4.00)$$

The minimum value is 1.00 therefore

$$K^* = 1.00$$

The optimal system will not have any redundant identical parallel subsystems.

Substituting K^* for \bar{K}

$$\prod_{i=1}^N 1 - (1 - R_i)^{\bar{M}_i} \geq 1 - \sqrt[1]{1 - .95}$$

and reducing the right hand side

$$\prod_{i=1}^N 1 - (1 - R_i)^{\bar{M}_i} \geq .95 \quad (5.3)$$

gives us the reliability goal for the single parallel-series subsystem. The resource constraints for this subsystem are

$$\text{Subsystem Cost: } \$1.2\bar{M}_1 + \$2.3\bar{M}_2 + \$3.4\bar{M}_3 \leq \$45.00$$

$$\text{Subsystem Weight: } 5\bar{M}_{1j} + 4\bar{M}_{2j} + 8\bar{M}_{3j} \leq 105.00$$

$$\text{Subsystem Volume: } 10\bar{M}_{1j} + 15\bar{M}_{2j} + 34\bar{M}_{3j} \leq 400.00$$

If we expand equation 5.3, substitute in the component reliabilities

$$[1 - (1 - .80)^{\bar{M}_1}] \cdot [1 - (1 - .70)^{\bar{M}_2}] \cdot [1 - (1 - .75)^{\bar{M}_3}] \geq .95$$

and simplify the equation

$$(1 - .20^{\bar{M}_1}) \cdot (1 - .30^{\bar{M}_2}) \cdot (1 - .25^{\bar{M}_3}) \geq .95$$

the lower bound on \bar{M}_1 is found by setting

$$(1 - .20^{\bar{M}_1}) \geq .95$$

We can now solve for \bar{M}_1

$$\bar{M}_1 \geq \frac{\ln(1 - .95)}{\ln .20}$$

The noninteger lower bound on \bar{M}_1 is 1.86. The integer lower bound is 2.0.

Therefore,

$$\bar{M}_{1LB} = 2.0$$

Similarly, one can solve for \bar{M}_{2LB} and \bar{M}_{3LB}

$$\bar{M}_{2LB} = 3.0$$

$$\bar{M}_{3LB} = 3.0$$

The possible integer upper bounds on variable \bar{M}_i are found by substituting \bar{M}_{jLB} for all $j \neq i$ into each of the resource constraints and solving for \bar{M}_{iUB} for each constraint. The possible integer upper bounds on each variable are

$$\bar{M}_{1UB} = \min(23, 13, 25)$$

$$\bar{M}_{2UB} = \min(14, 17, 18)$$

$$\bar{M}_{3UB} = \min(10, 10, 9)$$

Therefore,

$$\bar{M}_{1UB} = 13$$

$$\bar{M}_{2UB} = 14$$

$$\bar{M}_{3UB} = 9$$

We now have absolute upper and lower bounds on each of the variables. A smart search within these bounds, as discussed in chapter 3, will quickly converge on the optimum parallel-series subsystem specification.

$$\bar{M}_{11}^* = 5$$

$$\bar{M}_{21}^* = 8$$

$$\bar{M}_{31}^* = 6$$

The reliability of a parallel-series subsystem with these specifications is 0.9994.

The final step in determining the optimum specification for this system is to compute the remaining resources and optimize a final parallel-series subsystem using the remaining resources. In our example problem the remaining resources are

Cost: \$0.20
 Weight: 10.00
 Volume: 101.00

The minimum required resources for a series system having one component of each type are

Cost: \$6.90
 Weight: 17.00
 Volume: 59.00

There are insufficient resources to support an additional subsystem therefore the single parallel-series subsystem designed above constitutes the optimal system. The reliability of this system is

$$1 - \left[1 - \prod_{i=1}^3 1 - (1 - R_i)^{\bar{M}_i} \right]^1 = 0.9994$$

If we modify the right hand side values of the weight

and volume constraints on the subsystems a final time to

$$W_{SS} = 40$$

$$V_{SS} = 150$$

we will see that the optimum structure is different from either of the previous examples.

The new problem is

$$\text{Maximize } 1 - [1 - \prod_{i=1}^N 1 - (1 - R_i)^{\bar{M}_{i,j}}]^{\bar{K}}$$

Subject to

$$\text{Reliability: Total System Reliability} \geq .95$$

$$\text{System Cost: } \$1.2\bar{M}_1 + \$2.3\bar{M}_2 + \$3.4\bar{M}_3 \leq \$45.00$$

$$\text{System Weight: } 5\bar{M}_1 + 4\bar{M}_2 + 8\bar{M}_3 \leq 115.00$$

$$\text{System Volume: } 10\bar{M}_1 + 15\bar{M}_2 + 34\bar{M}_3 \leq 475.00$$

$$\text{Subsystem Weight: } 5\bar{M}_{1j} + 4\bar{M}_{2j} + 8\bar{M}_{3j} \leq 40.00$$

$$\text{Subsystem Volume: } 10\bar{M}_{1j} + 15\bar{M}_{2j} + 34\bar{M}_{3j} \leq 150.00$$

$$R_1 = 0.80$$

$$R_2 = 0.70$$

$$R_3 = 0.75$$

We know from equation 4.2 that the objective function reduces to

$$\prod_{i=1}^N 1 - (1 - R_i)^{\bar{M}_{i,j}} \geq 1 - \sqrt[\bar{K}]{1 - .95}$$

and solve for \bar{K}

$$\bar{K} = \min[\text{truncate}(W_{TS}/W_{SS}), \text{truncate}(V_{TS}/V_{SS})]$$

or

$$\min(2.00, 3.00)$$

The minimum value is 2.00 therefore

$$K^* = 2.00$$

The optimal system will have two identical parallel subsystems.

Substituting K^* for \bar{K}

$$\prod_{i=1}^N 1 - (1 - R_i)^{\bar{M}_i} \geq 1 - \sqrt[2]{1 - .95}$$

and reducing the right hand side

$$\prod_{i=1}^N 1 - (1 - R_i)^{\bar{M}_i} \geq .7764 \quad (5.4)$$

gives us the reliability goal for each of the identical parallel-series subsystems. The budget constraint for these subsystems is

$$\text{Subsystem Cost: } \$1.2\bar{M}_1 + \$2.3\bar{M}_2 + \$3.4\bar{M}_3 \leq (\$45.00/K^*)$$

The new resource constraints on the subsystems are

$$\text{Subsystem Cost: } \$1.2\bar{M}_1 + \$2.3\bar{M}_2 + \$3.4\bar{M}_3 \leq \$22.50$$

$$\text{Subsystem Weight: } 5\bar{M}_{1j} + 4\bar{M}_{2j} + 8\bar{M}_{3j} \leq 40.00$$

$$\text{Subsystem Volume: } 10\bar{M}_{1j} + 15\bar{M}_{2j} + 34\bar{M}_{nj} \leq 150.00$$

If we expand equation 5.4 and substitute in the component reliabilities

$$[1 - (1 - .80)^{\bar{M}_1}] \cdot [1 - (1 - .70)^{\bar{M}_2}] \cdot [1 - (1 - .75)^{\bar{M}_3}] \geq .7764$$

and simplify the equation

$$(1 - .20^{\bar{M}_1}) \cdot (1 - .30^{\bar{M}_2}) \cdot (1 - .25^{\bar{M}_3}) \geq .7764$$

We know that the lower bound on \bar{M}_1 is found by setting

$$(1 - .20^{\bar{M}_1}) \geq .7764$$

We can solve for \bar{M}_1

$$\bar{M}_1 \geq \frac{\ln(1 - .7764)}{\ln .20}$$

The noninteger lower bound on \bar{M}_1 is 0.9304. The integer lower bound is 1.0.

Therefore,

$$\bar{M}_{1LB} = 1.0$$

Similarly, one can solve for \bar{M}_{2LB} and \bar{M}_{3LB}

$$\bar{M}_{2LB} = 2.0$$

$$\bar{M}_{3LB} = 2.0$$

The integer upper bounds on variable \bar{M}_i is found by substituting \bar{M}_{jLB} for all $j \neq i$ into each of the resource constraints and solving for \bar{M}_{iUB} for each constraint. The integer upper bounds on each variable for each constraining goal are

$$\bar{M}_{1UB} = \min(9, 3, 5)$$

$$\bar{M}_{2UB} = \min(6, 4, 4)$$

$$\bar{M}_{3UB} = \min(4, 3, 3)$$

Therefore,

$$\bar{M}_{1UB} = 3$$

$$\bar{M}_{2UB} = 4$$

$$\bar{M}_{3UB} = 3$$

We now have absolute upper and lower bounds on each of the variables. The search, within these bounds, converged on these optimum feasible parallel-series subsystem specifications.

$$\bar{M}_{11}^* = 2$$

$$\bar{M}_{21}^* = 3$$

$$\bar{M}_{31}^* = 2$$

The reliability of a parallel-series subsystem with these specifications is 0.8775.

The final step in determining the optimum specification for this system is to compute the remaining resources and optimize a final parallel-series subsystem using the remaining resources. In this example problem the remaining resources are

Cost: \$12.80

Weight: 39.00

Volume: 209.00

There are sufficient resources to support an additional subsystem therefore we must determine the specifications of the final parallel-series subsystem. This is done by

optimizing within the remaining resources. The reliability goal of this final subsystem is the reliability of a subsystem containing just one of each type component. In this case, the reliability goal is 0.42.

The formulation of this problem is

$$\text{Maximize } 1 - \left[1 - \prod_{i=1}^N 1 - (1 - R_i)^{\bar{M}_{ij}} \right]^{\bar{K}}$$

Subject to

$$\text{Reliability: Total System Reliability} \geq .42$$

$$\text{Subsystem Cost: } \$1.2\bar{M}_1 + \$2.3\bar{M}_2 + \$3.4\bar{M}_3 \leq \$12.80$$

$$\text{Subsystem Weight: } 5\bar{M}_{1j} + 4\bar{M}_{2j} + 8\bar{M}_{3j} \leq 39.00$$

$$\text{Subsystem Volume: } 10\bar{M}_{1j} + 15\bar{M}_{2j} + 34\bar{M}_{3j} \leq 209.00$$

$$R_1 = 0.80$$

$$R_2 = 0.70$$

$$R_3 = 0.75$$

Solving this subsystem problem in the same manner as that described for the larger system yields the specifications for the final parallel-series subsystem. The optimal solution to this subsystem problem is

$$\bar{M}_1 = 2$$

$$\bar{M}_2 = 3$$

$$\bar{M}_3 = 1$$

The reliability of this subsystem is 0.70056. As one would

expect, the remaining resources

$$C = \$0.10$$

$$W_{TS} = 9.00$$

$$V_{TS} = 110.00$$

are not sufficient to support the addition of any components.

This algorithm has optimized the specifications of each subsystem and as a result has optimized the structure and specifications of the total system. The optimal structure is a parallel-series-parallel system with two identical parallel-series systems in parallel with one smaller parallel-series system. The total system reliability function value is

$$1 - \left\{ \left[1 - \prod_{i=1}^3 \left(1 - (1 - R_i)^{\bar{M}_{ij}^2} \right) \right] \left[1 - \prod_{i=1}^3 \left(1 - (1 - R_i)^{\bar{M}_{i3}} \right) \right] \right\} = 0.9955$$

where $j = (1,2)$

In this chapter we have shown empirically that the algorithm will determine the optimal system structure and design specifications regardless if that structure is a series-parallel system, a parallel-series system, or a parallel-series-parallel system.

Presently used methods for optimum design of redundant

component systems require that the system structure be defined first. That approach assumes that we are optimizing the best structure. This dissertation shows that we no longer must be bound by that, often suboptimal, assumption. The assumption is suboptimal in that we may have met the desired level of reliability, but at a greatly increased cost in available resources. What this means to the overall field of systems optimization, is that we can really do it right the first time. Designation of the system structure does not have to be done if all of the pertinent design criteria are available. If all of the design criteria are not known then perhaps the problem formulation is not complete.

The algorithm of this dissertation lets the systems designer use all of the design criteria available. The structure of the system does NOT have to be established prior to the optimization process, as was the case before this work. This algorithm decides if a feasible system is even possible that can meet all of the design criteria required. It establishes what the proper design structure is, and then maximizes the reliability within that structure. This ensures optimality in a way that conventional approaches cannot do at the structure level.

This algorithm determines the system with the maximum

level of reliability. In a parallel-series-parallel system, the $K+1$ subsystem will rarely have the same specifications as the other K identical subsystems. The result is a system that is optimal from the perspective of reliability but not necessarily optimal from other viewpoints. The designer may wish to have the optimal level of reliability, given that all subsystems are identical. This particular situation is one that may be a better choice in the eyes of a manufacturer. We may take advantage of economies of scale in production if all of the products are identical. Such a system, if it meets the minimum level of reliability, may be the "optimal" solution from the perspective of some. The work presented here has provided an initial position for future research in the growing complexity of this question.

Another important area for future research is that of broadening the algorithm to permit the inclusion of nonlinear resource constraints. It is usual for weight or cost equations to be more accurately depicted in a nonlinear equation. A technique that permits the bounding of variables when the problem structure includes nonlinear resource constraints would be of great benefit to this field of work.

The main contribution of this dissertation is the provision of a simple-to-use, generalized algorithm for

optimally solving a large class of nonlinear reliability problems, without the painful assumption of a priori structure. This unique technique first determines the optimal system structure and then optimizes the reliability within that structure. Such an approach ensures total optimality not just the best specifications given a particular structure.

REFERENCES CITED

- Federowicz, A. J. and M. Mazumdar. 1968. "Use of Geometric Programming to Maximize Reliability Achieved by Redundancy." Operations Research 15: 948-54.
- Ghare, P. M. and R. E. Taylor. 1969. "Optimal Redundancy for Reliability in Series Systems." Operations Research 17: 838-47.
- Kaufman, A., D. Grouchko, R. Cruon. 1977. Mathematical Models for the Study of the Reliability of Systems. New York: Academic Press.
- Jensen, P. A., 1970. "Optimization of Series-Parallel-Series Networks." Operations Research 18: 471-82.
- Mizukami, K. 1968. "Optimal Redundancy for Maximum System Reliability by the Method of Convex and Integer Programming." Operations Research 16: 392-406.
- Oatney, C. 1987. "An Algorithm for Solving a Class of Reliability Problems Using Geometric Programming." Master's thesis, Colorado School of Mines, Golden, Colorado.
- Tillman, F. A., C. Hwang, W. Kuo. 1980. Optimization of Systems Reliability. New York: Dekker.
- Wilkinson, J. G. 1987. "An Algorithm for Solving a Class of Multiple Goal Reliability Problems." Master's thesis, Colorado School of Mines, Golden, Colorado.

Appendix A

Proof

The following is a detailed proof that shows that a parallel-series system with the identical number and types of components as a series-parallel system will always have a level of reliability that is greater than or equal to that of the series-parallel system. This proof, in its entirety, is taken from Mathematical Models for the Study of the Reliability of Systems, Kaufman, Grouchko, and Cruon, 1977.

Let $\varphi(x_1, x_2, \dots, x_n)$ be a structure function with reliability function $h(p_1, p_2, \dots, p_n)$ associated with it.

Let:

S be the system obtained by placing in parallel k identical modules $\varphi(x_1^{(j)}, \dots, x_n^{(j)})$, $j=1, \dots, k$, with the reliabilities of the homologous components $e_i^{(1)}, \dots, e_i^{(k)}$ being equal,

$$p_i^{(j)} = p_i, \quad j = 1, \dots, k ;$$

C be the system obtained by replacing each component e_i of the structure φ by k distinct components in parallel $e_i^{(1)}, \dots, e_i^{(k)}$, with the same reliability p_i .

Then if the structure φ is monotone, the systems S and C have monotone structures and the reliability of C is greater than or equal to that of S.

The system S is obtained by composition of k modules identical to φ in a parallel structure:

$$\gamma(x^{(1)}, \dots, x^{(k)}) = 1 - \prod_{j=1}^k (1 - x^{(j)}).$$

Its structure function is therefore

$$\varphi_S(x_1^{(1)}, \dots, x_n^{(1)}, x_1^{(2)}, \dots, x_n^{(k)}) = \gamma[\varphi(x_1^{(1)}, \dots, x_n^{(1)}, x_1^{(k)}, \dots, x_n^{(k)})].$$

The system C is obtained by composing n modules identical to γ in the structure φ . Its structure function is therefore

$$\varphi_C(x_1^{(1)}, \dots, x_n^{(k)}, x_1^{(1)}, \dots, x_n^{(k)}) = \varphi[\gamma(x_1^{(1)}, \dots, x_n^{(k)}, x_1^{(1)}, \dots, x_n^{(k)})].$$

If the structure φ is monotone, it is representable by a reliability network; it is then the same for φ_S and φ_C . On the other hand, let $h(p_1, \dots, p_n)$ be the reliability function associated with the structure φ , and

$$l(p_i) = 1 - (1 - p_i)^k$$

the reliability function associated with the structure γ . Whenever all the homologous components have the same reliability p_i , the reliability functions of the systems S and C are, respectively, $l[h(p_1, \dots, p_n)]$ and $h[l(p_1), \dots, l(p_n)]$, and the theorem implies that

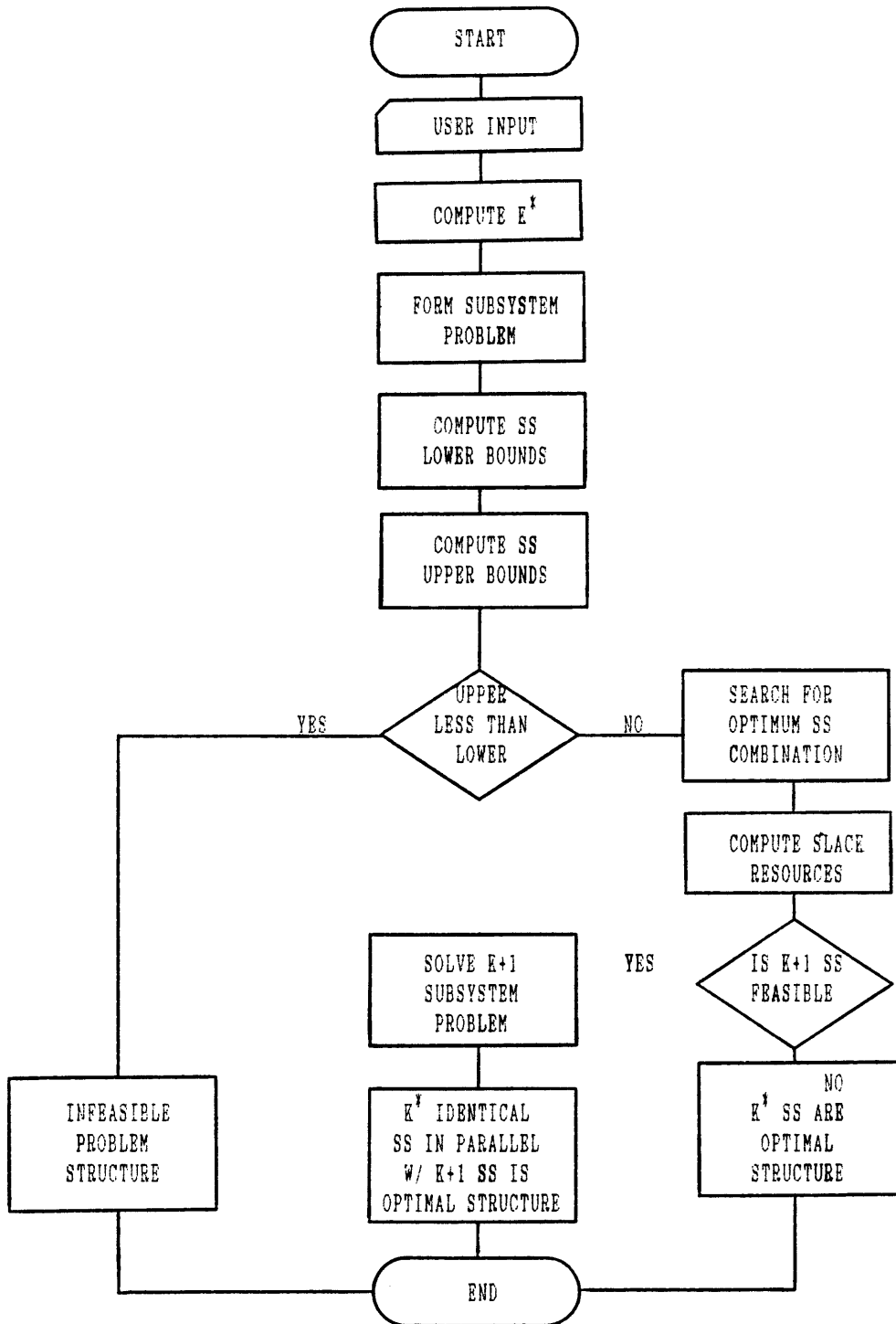
$$l[h(p_1, \dots, p_n)] \leq h[l(p_1), \dots, l(p_n)].$$

For proof we show that

$$\varphi_s \leq \varphi_c. \quad (\text{A.1})$$

Since the inequality is obvious for any state of the set of components such that $\varphi_s = 0$, we shall consider a link of φ_s . The structure γ being parallel, this link includes a link of at least one of the modules $\varphi^{(j)}$, say $\{e_{i_1}^{(j)}, \dots, e_{i_s}^{(j)}\}$. Then, however, the modules $\gamma_{i_1}, \dots, \gamma_{i_s}$ of the structure φ_c function since they have a parallel structure and each of them has at least one component in good state; since they constitute by hypothesis a link of φ , one has $\varphi_c = 1$. In other words, any link of φ_s is a link of φ_c , which proves (A.1).

Appendix B
 Program Flowchart



Appendix C

Computer Program of the Algorithm

The program developed for this dissertation is designed to run on any computer that has a standard Pascal compiler. For purposes of this dissertation the program was compiled and run on a DEC VAX-8600 using VMS. The program is limited, as written, to no more than ten variables. However, only minor modifications to the array specifications within the program are necessary to provide the user with the capacity to run problems having a larger number of component variables.

To run the program, it must be loaded into the computer and compiled into executable form. The program can then be run by typing "RUN THESIS". The program is designed to be user interactive. Once the program is running, the program will ask the user for all of the necessary input. Appendix D depicts sample computer runs that correspond to the problems in Chapter 5.

The computer program is listed below.

```

program MREL (input, output);
label 99;

type
  GRHS = array [1..4] of real;
  INPT = array [1..10] of real;
  BOUNDS = array [1..10] of integer;
  BOUNDSET = array [1..3] of integer;
  SOLUTIONSET = record
    VBLS: BOUNDS;
    VALS: INPT;
    FEAS: boolean
  end;

var
  STOPLOOP, ADDONE : boolean;
  LBESTSOL, CBESTSOL : SOLUTIONSET;
  COST, WT, VOL, PF : INPT;
  VALUE, LB, UB : BOUNDS;
  SUBGOALRHS, GOALRHS : GRHS;
  i, j, k, l, m, n, NUMBER, KSTAR : integer;
{
=====
CBESTSOL :    A RECORD CONTAINING THE VARIABLE VALUES,
              RELIABILITY, AND FEASIBILITY OF A COMBINATION
LBESTSOL :    A RECORD CONTAINING THE SAME AS CBESTSOL
COST :       ARRAY CONTAINING THE COEFFICIENTS OF THE
              VARIABLES IN THE BUDGET GOAL.
WT :        ARRAY CONTAINING THE COEFFICIENTS OF THE
              VARIABLES IN THE WEIGHT GOAL.
VOL :       ARRAY CONTAINING THE COEFFICIENTS OF THE
              VARIABLES IN THE VOLUME GOAL.
PF :        ARRAY CONTAINING THE PROBABILITY OF FAILURE
              OF EACH VARIABLE.
VALUE :     ARRAY CONTAINING THE VARIABLE VALUES FOR A
              COMBINATION.
LB :        ARRAY CONTAINING THE LOWER BOUNDS ON EACH
              VARIABLE.
UB :        ARRAY CONTAINING THE UPPER BOUNDS ON EACH
              VARIABLE.
GOALRHS :   ARRAY CONTAINING THE RIGHTHAND SIDE VALUE OF
              ALL GOALS.
SUBGOALRHS : ARRAY CONTAINING THE RIGHTHAND SIDE VALUE OF
              ALL SUBGOALS.
KSTAR :     THE NUMBER OF IDENTICAL PARALLEL SUBSYSTEMS
i, j, k, n : COUNTERS.
NUMBER :    INTEGER VARIABLE CONTAINING THE NUMBER OF
              VARIABLES IN THE PROBLEM.

```

```

=====
}
procedure INITIALIZE (var COST, WT, VOL, PF : INPT; var GOALRHS,
                     SUBGOALRHS : GRHS);

{ THIS PROCEDURE INTERACTIVELY OBTAINS THE INITIAL DATA ON
***** EACH OF THE GOALS FROM THE USER. *****}
label 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,19,20,21,22;
var
    i, j : integer;
    ANS : char;

{***** i, j : COUNTERS. *****}
**** ANS : CHARACTER VARIABLE FOR THE (Y/N) REPLY. ****}
begin {INITIALIZE}
1:    writeln ('Please input the probability of failure of each')
      writeln ('component, in order, starting with item #1,');
      writeln ('followed by a <CR>.');
      readln (PF[1]);
      for i := 2 to NUMBER do
      begin
          writeln ('Item #',i:1,'?');
          readln (PF[i])
      end;
      writeln ('The probability of failure for each item is:');
      for i := 1 to NUMBER do
          writeln ('Item #',i:1,' = ',PF[i]:4:3);
2:    writeln ('Are these values correct? Enter (Y/N)');
      readln (ANS);
      if ANS = 'N' then goto 1
      else if ANS <> 'Y' then goto 2;
3:    writeln ('Please input the required level of system');
      writeln ('reliability. ');
      readln (GOALRHS[1]);
      writeln ('The required system reliability is ',
              GOALRHS[1]:5:4);
4:    writeln ('Is this correct? Enter (Y/N)');
      readln (ANS);
      if ANS = 'N' then goto 3
      else if ANS <> 'Y' then goto 4;
5:    writeln ('Input the cost of each component, in order,');
      writeln ('starting with item #1, followed by a <CR>.');
      readln (COST[1]);
      for i := 2 to NUMBER do
      begin
          writeln ('Item #',i:1,'?');
          readln (COST[i])
      end;
end;

```

```

writeln ('The cost coefficients for each item:');
for i := 1 to NUMBER do
    writeln ('Item #',i:1,' = ',COST[i]:6:2);
6:  writeln ('Are all of the values correct? Enter (Y/N)');
    readln (ANS);
    if ANS = 'N' then goto 5
    else if ANS <> 'Y' then goto 6;
7:  writeln ('Input the total budget available:');
    readln (GOALRHS[2]);
    writeln ('The available budget is ', GOALRHS[2]:10:2);
8:  writeln ('Is this value correct? Enter (Y/N)');
    readln (ANS);
    if ANS = 'N' then goto 7
    else if ANS <> 'Y' then goto 8;
9:  writeln ('Input the weight of each component, in order,');
    writeln ('starting with item #1, followed by a <CR>.');
    readln (WT[1]);
    for i := 2 to NUMBER do
        begin
            writeln ('Item #',i:1,'?');
            readln (WT[i])
        end;
    writeln ('The weight of each item:');
    for i := 1 to NUMBER do
        writeln ('Item #',i:1,' = ',WT[i]:6:2);
10: writeln ('Are all of the values correct? Enter (Y/N)');
    readln (ANS);
    if ANS = 'N' then goto 9
    else if ANS <> 'Y' then goto 10;
11: writeln ('Input the total weight available:');
    readln (GOALRHS[3]);
    writeln ('The available weight is ', GOALRHS[3]:10:2);
12: writeln ('Is this value correct? Enter (Y/N)');
    readln (ANS);
    if ANS = 'N' then goto 11
    else if ANS <> 'Y' then goto 12;
19: writeln ('Input the maximum weight for each subsystem:');
    readln (SUBGOALRHS[3]);
    writeln ('The maximum weight is ', SUBGOALRHS[3]:10:2);
20: writeln ('Is this value correct? Enter (Y/N)');
    readln (ANS);
    if ANS = 'N' then goto 19
    else if ANS <> 'Y' then goto 20;
13: writeln ('Input the volume of each component, in order,');
    writeln ('starting with item #1, followed by a <CR>.');
    readln (VOL[1]);
    for i := 2 to NUMBER do
        begin

```

```

        writeln ('Item #',i:1,'?');
        readln (VOL[i])
    end;
    writeln ('The volume of each item:');
    for i := 1 to NUMBER do
        writeln ('Item #',i:1,' = ',VOL[i]:6:2);
14:    writeln ('Are all of the values correct? Enter (Y/N)');
        readln (ANS);
        if ANS = 'N' then goto 13
        else if ANS <> 'Y' then goto 14;
15:    writeln ('Input the total volume available:');
        readln (GOALRHS[4]);
        writeln ('The available volume is ', GOALRHS[4]:10:2);
16:    writeln ('Is this value correct? Enter (Y/N)');
        readln (ANS);
        if ANS = 'N' then goto 15
        else if ANS <> 'Y' then goto 16;
21:    writeln ('Input the maximum subsystem volume:');
        readln (SUBGOALRHS[4]);
        writeln ('The maximum volume is ', SUBGOALRHS[4]:10:2);
22:    writeln ('Is this value correct? Enter (Y/N)');
        readln (ANS);
        if ANS = 'N' then goto 21
        else if ANS <> 'Y' then goto 22;
        writeln ('STANDBY, DETERMINING SYSTEM STRUCTURE');
        writeln ('STANDBY, COMPUTING VARIABLE BOUNDS')
    end; {INITIALIZE}

{=====}
procedure COMPUTEK (var KSTAR : integer; GOALRHS, SUBGOALRHS
                   : GRHS);
{THIS PROCEDURE COMPUTES KSTAR, THE NUMBER OF IDENTICAL SUBSYSTEMS}
var
    KWEIGHT, KVOLUME : integer;
begin
    KWEIGHT := trunc(GOALRHS[3]/SUBGOALRHS[3]);
    KVOLUME := trunc(GOALRHS[4]/SUBGOALRHS[4]);
    if KWEIGHT <= KVOLUME
        then KSTAR := KWEIGHT
        else KSTAR := KVOLUME
end;
{=====}
procedure SUBRELI (var SUBGOALRHS : GRHS; KSTAR : integer;
                  SYSREL : REAL);

{THIS PROCEDURE COMPUTES THE RELIABILITY GOAL FOR THE KSTAR
IDENTICAL SUBSYSTEMS}
begin

```

```

        SUBGOALRHS[1] := 1.0 - exp((1/KSTAR) * ln(1.0 - SYSREL))
end;
{=====}
procedure SUBCOST (var SUBGOALRHS : GRHS; KSTAR : integer;
SYSCOST : REAL);

{THIS PROCEDURE COMPUTES THE COST GOAL FOR EACH OF THE KSTAR
IDENTICAL SUBSYSTEMS}
begin
    SUBGOALRHS[2] := SYSCOST/KSTAR
end;
{=====}
procedure LOWER (var LB : BOUNDS; REL : real; PF : INPT);

{THIS PROCEDURE COMPUTES THE LOWER BOUNDS ON EACH VARIABLE}
var
    i : integer;
begin
    {LOWER}
    for i := 1 to NUMBER do
        begin
            LB[i] := trunc(ln(1.0 - REL) / ln(PF[i])) + 1
        end;
        writeln ('The lower bounds on each variable:');
        for i := 1 to NUMBER do
            begin
                writeln ('X',i:1,' = ',LB[i]:1)
            end
        end;
    {LOWER}
end;
{=====}
procedure COMPUTESET (var X : BOUNDS; COEFF : INPT;
RHS : real; LB : BOUNDS);

{THIS PROCEDURE COMPUTES THE UPPER BOUND ON EACH VARIABLE
***** WITH RESPECT TO EACH GOAL. *****}
var
    i, j : integer;
    TEMP : real;
begin
    {COMPUTESET}
    for i := 1 to NUMBER do
        begin
            TEMP := 0.0;
            for j := 1 to NUMBER do
                begin
                    if j <> i then
                        TEMP := TEMP + (COEFF[j]*LB[j])
                    end;
                X[i] := trunc((RHS - TEMP) / COEFF[i])
            end
        end
    end
end

```

```

end; {COMPUSET}
{=====}
procedure UPPER (var UB : BOUNDS; COST, WT, VOL : INPT;
                 GOALRHS : GRHS; LB : BOUNDS);

{** THIS PROCEDURE DETERMINES THE UPPER BOUNDS ON EACH **
***** VARIABLE. *****}

var
    i, j, TEMP : integer;
    XC, XW, XV : BOUNDS;

{** XC, XW, XV : ARRAYS CONTAINING UPPER BOUNDS ON EACH **
***** VARIABLE WITH RESPECT TO COST, WEIGHT AND *****
***** VOLUME GOALS. *****}

begin
    COMPUTESET (XC, COST, SUBGOALRHS[2], LB);
    COMPUTESET (XW, WT, SUBGOALRHS[3], LB);
    COMPUTESET (XV, VOL, SUBGOALRHS[4], LB);
    TEMP := MAXINT;
    writeln;
    writeln ('The upper bounds on each variable:');
    for i := 1 to NUMBER do
    begin
        if XC[i] < TEMP then
            TEMP := XC[i];
        if XW[i] < TEMP then
            TEMP := XW[i];
        if XV[i] < TEMP then
            TEMP := XV[i];
        UB[i] := TEMP;
        writeln ('X',i:1,' = ',UB[i]:1);
        TEMP := MAXINT
    end;
    for i := 1 to NUMBER do
    begin
        if UB[i] < LB[i] then

{*** IF THE UPPER BOUND ON A VARIABLE IS LESS THAN THE ***
* LOWER BOUND THEN THE SYSTEM IS STRUCTURALLY INFEASIBLE *}

        begin
            writeln ('System of goals has an infeasible');
            writeln ('structure. One items lower bound');
            writeln ('is greater than its upper bound.')
        end
    end
end

```

```
end; {UPPER}
```

```
{=====}
```

```
procedure COMPUTEVALUES (VALUE : BOUNDS; COST, WT, VOL, PF : INPT;
    var STOPLOOP : boolean; SUBGOALRHS : GRHS;
    var CBESTSOL : SOLUTIONSET);
```

```
{THIS PROCEDURE COMPUTES THE VALUES, WITH RESPECT TO EACH
** GOAL, AND STORES THE CURRENT BEST SOLUTION **}
```

```
var
```

```
    TEMPR, TEMPC, TEMPW, TEMPV : real;
    FEAS : boolean;
    TBESTSOL : SOLUTIONSET;
```

```
{=====}
```

```
TEMPV : " " " " " " VOLUME "
FEAS : BOOLEAN VARIABLE (true/false) BASED ON FEASIBILITY
      OF A COMBINATION.
```

```
=====}
```

```
begin
```

```
    FEAS := true;
    TEMPR := 1.0;
    TEMPC := 0.0;
    TEMPW := 0.0;
    TEMPV := 0.0;
    for i := 1 to NUMBER do
```

```
{***** COMPUTE THE RELIABILITY OF THIS COMBINATION. *****)
```

```
        TEMPR := TEMPR * (1.0 - (exp(VALUE[i] * ln(PF[i])))
    TBESTSOL.VALS[1] := TEMPR;
    for i := 1 to NUMBER do
```

```
{***** COMPUTE THE COST OF THIS COMBINATION. *****)
```

```
        TEMPC := TEMPC + (COST[i] * VALUE[i]);
    TBESTSOL.VALS[2] := TEMPC;
    for i := 1 to NUMBER do
```

```
{***** COMPUTE THE WEIGHT OF THIS COMBINATION. *****)
```

```
        TEMPW := TEMPW + (WT[i] * VALUE[i]);
    TBESTSOL.VALS[3] := TEMPW;
    for i := 1 to NUMBER do
```

```
{***** COMPUTE THE VOLUME OF THIS COMBINATION. *****)
```

```

        TEMPV := TEMPV + (VOL[i] * VALUE[i]);
    TBESTSOL.VALS[4] := TEMPV;
    for i := 1 to NUMBER do

{***** RECORD THE VARIABLE VALUES OF THIS COMBINATION. *****)

        TBESTSOL.VBLS[i] := VALUE[i];
    if (TBESTSOL.VALS[1] < SUBGOALRHS[1]) or
        (TBESTSOL.VALS[2] > SUBGOALRHS[2]) or
        (TBESTSOL.VALS[3] > SUBGOALRHS[3]) or
        (TBESTSOL.VALS[4] > SUBGOALRHS[4]) then
        TBESTSOL.FEAS := false
    else TBESTSOL.FEAS := true;

{***** CHECK IF THE COMBINATION SATISFIES ALL GOALS. *****)

    if TBESTSOL.FEAS then

{** IF ALL GOALS ARE SATISFIED THEN CHECK IF THIS IS BEST. **}
        if TBESTSOL.VALS[1] > CBESTSOL.VALS[1] then
            CBESTSOL := TBESTSOL
        else
            begin
                STOPLOOP := true;
                if not CBESTSOL.FEAS then
                    if (TBESTSOL.VALS[1] <= SUBGOALRHS[1]) and
                        (TBESTSOL.VALS[2] <= SUBGOALRHS[2]) and
                        (TBESTSOL.VALS[3] <= SUBGOALRHS[3]) and
                        (TBESTSOL.VALS[4] <= SUBGOALRHS[4]) then
                            CBESTSOL := TBESTSOL
                    end
            end;
    end; {COMPUTEVALUES}

{=====}

procedure CHECKCOMBS (LB, UB : BOUNDS; LOOPS : integer;
    var VALUE : BOUNDS; COST, WT, VOL, PF : INPT;
    SUBGOALRHS : GRHS; var STOPLOOP : boolean;
    var CBESTSOL:SOLUTIONSET);

{***** THIS PROCEDURE RECURSIVELY CHECKS ALL POSSIBLE *****)
{** COMBINATIONS TO DETERMINE THOSE THAT ARE FEASIBLE. ** }

var
    i, j : integer;

begin {CHECKCOMBS}

```

```

STOPLOOP := false;
for i := UB[LOOPS] downto LB[LOOPS] do
begin
    VALUE[LOOPS] := i;
    if LOOPS > 2 then

**** IF ALL VARIABLE VALUES FOR THIS COMBINATION HAVE ****
**** NOT BEEN DETERMINED THEN RECURSE ONE MORE LEVEL. ****

        CHECKCOMBS (LB, UB, LOOPS - 1, VALUE,
                    COST, WT, VOL, PF, SUBGOALRHS, STOPLOOP,
                    CBESTSOL);

    for j := UB[1] downto LB[1] do
    begin
        VALUE[1] := j;
        if not STOPLOOP then
        begin
            COMPUTEVALUES (VALUE, COST, WT,
                          VOL, PF, STOPLOOP, SUBGOALRHS, CBESTSOL)
        end
    end;
    STOPLOOP := false
end
end; {CHECKCOMBS}

{=====}
procedure CHECKSLACKS (CBESTSOL : SOLUTIONSET; var SUBGOALRHS:
GRHS; GOALRS: GRHS; COST, WT, VOL : INPT; var ADDON: boolean);

{THIS PROCEDURE CHECKS IF THIS ARE ENOUGH REMAINING RESOURCES
TO SUPPORT THE K+1 SUBSYSTEM}
var
    i,j : integer;
    TCST, TWT, TVLM, TREL, SCST, SWT, SVLM : real;

begin
    TCST := 0.0;
    TWT := 0.0;
    TVLM := 0.0;
    SCST := 0.0;
    SWT := 0.0;
    SVLM := 0.0;
    TREL := 1.0;
    for i := 1 to NUMBER do
    begin
        TCST := TCST + (COST[i] * CBESTSOL.VBLS[i] * KSTAR);
        TWT := TWT + (WT[i] * CBESTSOL.VBLS[i] * KSTAR);
        TVLM := TVLM + (VOL[i] * CBESTSOL.VBLS[i] * KSTAR);
    end
end

```

```

        SCST := SCST + COST[i];
        SWT := SWT + WT[i];
        SVLM := SVLM + VOL[i]
    end;
    if ((GOALRHS[2] - TCST) < SCST) or
        ((GOALRHS[3] - TWT) < SWT) or
        ((GOALRHS[4] - TVLM) < SVLM) then
        ADDONE := false;
    if ADDONE then
    begin
        for i := 1 to NUMBER do
            TREL := TREL * (1.0 - PF[i]);
            SUBGOALRHS[1] := TREL;
            SUBGOALRHS[2] := GOALRHS[2] - TCST;
            SUBGOALRHS[3] := GOALRHS[3] - TWT;
            SUBGOALRHS[4] := GOALRHS[4] - TVLM
        end
    end;
    {=====}
    procedure FINDLAST (SUBGOALRHS : GRHS; var LB, UB,
        VALUE : BOUNDS; COST, WT, VOL, PF : INPT;
        var CBESTSOL : SOLUTIONSET);

    {THIS PROCEDURE COMPUTES THE SYSTEM SPECIFICATIONS FOR THE
    FINAL SUBSYSTEM}
    var
        i, j : integer;
    begin
        for i:= 1 to NUMBER do
            LB[i] := 1;
            UPPER (UB, COST, WT, VOL, SUBGOALRHS, LB);
            CHECKCOMBS (LB, UB, NUMBER, VALUE, COST, WT, VOL, PF,
                SUBGOALRHS, STOPLOOP, CBESTSOL)
        end;
    {=====}
    procedure PRINTOUT (LBESTSOL, CBESTSOL : SOLUTIONSET;
        KSTAR : integer; REQREL : real; ADDONE : boolean);

    {THIS PROCEDURE COMPUTES TOTAL SYSTEM RELIABILITY AND PRINTS
    FINAL SYSTEM STRUCTURE AND SPECIFICATIONS}

    var
        i, j : integer;
        TEMP1, TEMP2, TEMP3 : real;
    begin
        if ADDONE then
            TEMP2 := CBESTSOL.VALS[1]
        else

```

```

        TEMP2 := 0.0;
        TEMP1 := LBESTSOL.VALS[1];
        TEMP3 := 1.0 - (exp(KSTAR * ln(1.0 - TEMP1)) *
                        (1.0 - TEMP2));
        if TEMP3 >= REQREL then
        begin
            writeln ('THE OPTIMAL SYSTEM MEETS THE');
            writeln ('REQUIRED LEVEL OF RELIABILITY');
        end
        else
        begin
            writeln ('THE SYSTEM FAILS TO MEET THE');
            writeln ('REQUIRED LEVEL OF RELIABILITY');
        end;
        writeln;
        writeln ('THE ',KSTAR:1,' IDENTICAL SUBSYSTEMS');
        writeln ('EACH HAVE THE FOLLOWING SPECIFICATIONS:');
        writeln;
        for i := 1 to NUMBER do
            write ('X[' ,i:1,' ] = ',LBESTSOL.VBLS[i]:1,' ');
        writeln;
        if ADDONE then
        begin
            writeln;
            writeln ('THE FINAL SUBSYSTEM HAS THE ');
            writeln ('FOLLOWING SPECIFICATIONS:');
            writeln;
            for i := 1 to NUMBER do
                write ('X[' ,i:1,' ] = ',CBESTSOL.VBLS[i]:1,' ');
            writeln
        end;
        writeln;
        writeln ('TOTAL SYSTEM RELIABILITY IS: ',TEMP3:5:4)
    end;

    {=====}

begin {MAIN PROGRAM}
    writeln ('This algorithm will provide the user with');
    writeln ('information on all integer combinations that');
    writeln ('will satisfy a series of system reliability');
    writeln ('goals. If there are no combinations that will');
    writeln ('satisfy all of the goals then this algorithm');
    writeln ('will provide the user with the best');
    writeln ('alternative based on the goal priority');
    writeln ('established by the user');
    writeln;
    writeln ('Please input the number of variables,');

```

```
writeln ('followed by a <CR>');
readln (NUMBER);
STOPLOOP := false;
ADDONE := true;
INITIALIZE (COST, WT, VOL, PF, GOALRHS, SUBGOALRHS);
COMPUTEK (KSTAR, GOALRHS, SUBGOALRHS);
SUBRELI (SUBGOALRHS, KSTAR, GOALRHS[1]);
SUBCOST (SUBGOALRHS, KSTAR, GOALRHS[2]);
LOWER (LB, SUBGOALRHS[1], PF);
UPPER (UB, COST, WT, VOL, SUBGOALRHS, LB);
for i := 1 to NUMBER do
    if UB[i] < LB[i] then
        goto 99;
for i := 1 to NUMBER do
    CBESTSOL.VBLS[i] := 0;
for i := 1 to 4 do
    CBESTSOL.VALS[i] := 0;
CBESTSOL.FEAS := false;
CHECKCOMBS (LB, UB, NUMBER, VALUE, COST, WT, VOL, PF,
            SUBGOALRHS, STOPLOOP, CBESTSOL);
CHECKSLACKS (CBESTSOL, SUBGOALRHS, GOALRHS, COST, WT,
            VOL, ADDONE);

STOPLOOP := false;
LBESTSOL := CBESTSOL;
for i := 1 to NUMBER do
    CBESTSOL.VBLS[i] := 0;
for i := 1 to 4 do
    CBESTSOL.VALS[i] := 0;
if ADDONE then
begin
    FINDLAST (SUBGOALRHS, LB, UB, VALUE,
            COST, WT, VOL, PF, CBESTSOL)
end;
PRINTOUT (LBESTSOL, CBESTSOL, KSTAR, GOALRHS[1], ADDONE);
99: writeln
end. {MAIN PROGRAM}
```

APPENDIX D

Sample Program Output

This appendix provides a verbatim transcript of the computer program interaction with the user in solving the three example problems shown in Chapter 5.

TRANSCRIPT FOLLOWS:

This algorithm will provide the user with information on all integer combinations that will satisfy a series of system reliability goals. If there are no combinations that will satisfy all of the goals then this algorithm will provide the user with the best alternative based on the goal priority established by the user

Please input the number of variables,
followed by a <CR>

3

Please input the probability of failure of each component, in order, starting with item #1, followed by a <CR>.

.2

Item #2?

.3

Item #3?

.25

The probability of failure for each item is:

Item #1 = 0.200

Item #2 = 0.300

Item #3 = 0.250

Are these values correct? Enter (Y/N)

Y

Please input the required level of system reliability.

.95

The required system reliability is 0.9500

Is this correct? Enter (Y/N)

Y

Input the cost of each component, in order, starting with item #1, followed by a <CR>.

1.2

Item #2?

2.3

Item #3?

3.4

The cost coefficients for each item:

Item #1 = 1.20

Item #2 = 2.30

Item #3 = 3.40

Are all of the values correct? Enter (Y/N)

Y

Input the total budget available:

45

The available budget is 45.00

Is this value correct? Enter (Y/N)

Y

Input the weight of each component, in order, starting with item #1, followed by a <CR>.

5

Item #2?

4

Item #3?

8

The weight of each item:

Item #1 = 5.00

Item #2 = 4.00

Item #3 = 8.00

Are all of the values correct? Enter (Y/N)

Y

Input the total weight available:

115

The available weight is 115.00

Is this value correct? Enter (Y/N)

Y

Input the maximum weight for each subsystem:

19

The maximum weight is 19.00

Is this value correct? Enter (Y/N)

Y

Input the volume of each component, in order, starting with item #1, followed by a <CR>.

10

Item #2?

15

Item #3?

34

The volume of each item:

Item #1 = 10.00

Item #2 = 15.00
Item #3 = 34.00
Are all of the values correct? Enter (Y/N)

Y

Input the total volume available:
475

The available volume is 475.00
Is this value correct? Enter (Y/N)

Y

Input the maximum subsystem volume:
65

The maximum volume is 65.00
Is this value correct? Enter (Y/N)

Y

STANDBY, DETERMINING SYSTEM STRUCTURE

STANDBY, COMPUTING VARIABLE BOUNDS

The lower bounds on each variable:

X1 = 1

X2 = 1

X3 = 1

The upper bounds on each variable:

X1 = 1

X2 = 1

X3 = 1

THE OPTIMAL SYSTEM MEETS THE
REQUIRED LEVEL OF RELIABILITY

THE 6 IDENTICAL SUBSYSTEMS
EACH HAVE THE FOLLOWING SPECIFICATIONS:

X[1] = 1 X[2] = 1 X[3] = 1

TOTAL SYSTEM RELIABILITY IS: 0.9619

This algorithm will provide the user with information on all integer combinations that will satisfy a series of system reliability goals. If there are no combinations that will satisfy all of the goals then this algorithm will provide the user with the best alternative based on the goal priority established by the user

Please input the number of variables,
followed by a <CR>

3

Please input the probability of failure of each component, in order, starting with item #1, followed by a <CR>.

.2

Item #2?

.3

Item #3?

.25

The probability of failure for each item is:

Item #1 = 0.200

Item #2 = 0.300

Item #3 = 0.250

Are these values correct? Enter (Y/N)

Y

Please input the required level of system reliability.

.95

The required system reliability is 0.9500

Is this correct? Enter (Y/N)

Y

Input the cost of each component, in order, starting with item #1, followed by a <CR>.

1.2

Item #2?

2.3

Item #3?

3.4

The cost coefficients for each item:

Item #1 = 1.20

Item #2 = 2.30

Item #3 = 3.40

Are all of the values correct? Enter (Y/N)

Y

Input the total budget available:

45

The available budget is 45.00

Is this value correct? Enter (Y/N)

Y

Input the weight of each component, in order, starting with item #1, followed by a <CR>.

5

Item #2?

4

Item #3?

8

The weight of each item:

Item #1 = 5.00

Item #2 = 4.00

Item #3 = 8.00
Are all of the values correct? Enter (Y/N)
Y
Input the total weight available:
115
The available weight is 115.00
Is this value correct? Enter (Y/N)
Y
Input the maximum weight for each subsystem:
105
The maximum weight is 105.00
Is this value correct? Enter (Y/N)
Y
Input the volume of each component, in order,
starting with item #1, followed by a <CR>.
10
Item #2?
15
Item #3?
34
The volume of each item:
Item #1 = 10.00
Item #2 = 15.00
Item #3 = 34.00
Are all of the values correct? Enter (Y/N)
Y
Input the total volume available:
475
The available volume is 475.00
Is this value correct? Enter (Y/N)
Y
Input the maximum subsystem volume:
400
The maximum volume is 400.00
Is this value correct? Enter (Y/N)
Y
STANDBY, DETERMINING SYSTEM STRUCTURE
STANDBY, COMPUTING VARIABLE BOUNDS
The lower bounds on each variable:
X1 = 2
X2 = 3
X3 = 3

The upper bounds on each variable:
X1 = 13
X2 = 14
X3 = 9
THE OPTIMAL SYSTEM MEETS THE

REQUIRED LEVEL OF RELIABILITY

THE 1 IDENTICAL SUBSYSTEMS
EACH HAVE THE FOLLOWING SPECIFICATIONS:

X[1] = 5 X[2] = 8 X[3] = 6

TOTAL SYSTEM RELIABILITY IS: 0.9994

This algorithm will provide the user with information on all integer combinations that will satisfy a series of system reliability goals. If there are no combinations that will satisfy all of the goals then this algorithm will provide the user with the best alternative based on the goal priority established by the user

Please input the number of variables,
followed by a <CR>

3

Please input the probability of failure of each component, in order, starting with item #1,
followed by a <CR>.

.2

Item #2?

.3

Item #3?

.25

The probability of failure for each item is:

Item #1 = 0.200

Item #2 = 0.300

Item #3 = 0.250

Are these values correct? Enter (Y/N)

Y

Please input the required level of system
reliability.

.95

The required system reliability is 0.9500

Is this correct? Enter (Y/N)

Y

Input the cost of each component, in order,
starting with item #1, followed by a <CR>.

1.2

Item #2?

2.3

Item #3?

3.4

The cost coefficients for each item:

Item #1 = 1.20

Item #2 = 2.30

Item #3 = 3.40

Are all of the values correct? Enter (Y/N)

Y

Input the total budget available:

45

The available budget is 45.00

Is this value correct? Enter (Y/N)

Y

Input the weight of each component, in order, starting with item #1, followed by a <CR>.

5

Item #2?

4

Item #3?

8

The weight of each item:

Item #1 = 5.00

Item #2 = 4.00

Item #3 = 8.00

Are all of the values correct? Enter (Y/N)

Y

Input the total weight available:

115

The available weight is 115.00

Is this value correct? Enter (Y/N)

Y

Input the maximum weight for each subsystem:

40

The maximum weight is 40.00

Is this value correct? Enter (Y/N)

Y

Input the volume of each component, in order, starting with item #1, followed by a <CR>.

10

Item #2?

15

Item #3?

34

The volume of each item:

Item #1 = 10.00

Item #2 = 15.00

Item #3 = 34.00

Are all of the values correct? Enter (Y/N)

Y

Input the total volume available:

475

The available volume is 475.00

Is this value correct? Enter (Y/N)

Y

Input the maximum subsystem volume:

150

The maximum volume is 150.00

Is this value correct? Enter (Y/N)

Y

STANDBY, DETERMINING SYSTEM STRUCTURE

STANDBY, COMPUTING VARIABLE BOUNDS

The lower bounds on each variable:

X1 = 1

X2 = 2

X3 = 2

The upper bounds on each variable:

X1 = 3

X2 = 4

X3 = 3

The upper bounds on each variable:

X1 = 5

X2 = 3

X3 = 2

THE OPTIMAL SYSTEM MEETS THE
REQUIRED LEVEL OF RELIABILITY

THE 2 IDENTICAL SUBSYSTEMS
EACH HAVE THE FOLLOWING SPECIFICATIONS:

X[1] = 2 X[2] = 3 X[3] = 2

THE FINAL SUBSYSTEM HAS THE
FOLLOWING SPECIFICATIONS:

X[1] = 2 X[2] = 3 X[3] = 1

TOTAL SYSTEM RELIABILITY IS: 0.9954