Chapters on Reliability

by

Subbarao Venkata Majety

B.Tech, Jawaharlal Nehru Technological University, India 1985

M.A.Sc, University of Windsor, Canada, 1993

Submitted to the Graduate Faculty of the

Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2022

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Subbarao Venkata Majety

It was defended on

September 2, 2022

and approved by

Prakash Mirchandani, PhD, Professor, Katz Graduate School of Business

Lisa Maillart, PhD, Professor, Department of Industrial Engineering

Hoda Bidkhori, PhD, Assistant Professor, Department of Industrial Engineering

Dissertation Director: Jayant Rajgopal, PhD, Professor, Department of Industrial Engineering

Chapters on Reliability

Subbarao Venkata Majety, PhD

University of Pittsburgh, 2022

In this research two problems related to system reliability are addressed: the first is commonly referred to as the reliability allocation problem, and the second problem is the development of a class of optimum test plans for demonstrating system reliability.

The reliability allocation problem addressed in this research is for discrete cost-reliability data sets. Integer programming formulations are presented and solutions are developed based on three approaches: (i) integer programming, (ii) simulated annealing, and (iii) evolutionary algorithms. Except for simple series and simple parallel systems, the integer programs formulated are non-linear. Specifically, Series-Parallel (*SP*) and Parallel-Series (*PS*) systems are discussed in detail in this research. With the integer programming approach, linear relaxations are developed for the systems and an iterative procedure is developed to solve the problems. In this iterative procedure a single infeasible solution is eliminated at each iteration until a feasible optimal solution is achieved. With the simulated annealing approach, a nested algorithm is developed, where an inner process focuses on optimality while an outer one focuses on feasibility. With the evolutionary algorithm, a dynamic penalty approach is used to accelerate the convergence of the algorithm to a good solution.

The second topic addresses test plans for systems and components. In the development of any system, testing is very important to ensure that a good system is accepted while a bad system is rejected. In this research, we address a series system with the possibility of interface failures. For these systems our research evaluates when the less expensive approach of testing only

components is good enough, when it is necessary to test the whole system, and when a combination of component and system tests might be best.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to acknowledge my sincere gratitude to Dr. Jayant Rajgopal, my advisor, and Dr. Mainak Mazumdar for incredible guidance, advice and knowledge imparted to me during my entire stay at the University of Pittsburgh. I would like to acknowledge my sincere gratitude to Dr. Larry Shuman who graciously allowed for the defense of my research. I would like to offer my sincere gratitude to the committee members Dr. Prakash Mirchandani, Dr. Lisa Maillart and Dr. Hoda Bidkhori for graciously agreeing to serve on my thesis dissertation committee. I would like to acknowledge my sincere gratitude to Dr. Egon Balas and Dr. Alice Smith, whose courses have given me immense knowledge, which contributed significantly to this research. I would like to extend my sincere gratitude to Dr. Milind Dawande for his friendship during my years in Pittsburgh.

I would also like to acknowledge my sincere gratitude to my wife Raji, and my daughters Saveda and Anushka, who patiently waited for the day of my graduation. My late brother Dr. Venkata Majeti had dreamed that I would graduate. I offer my sincere prayers to him for his blessings from above.

I would also like to thank all the staff and faculty of the Industrial Engineering department at Pitt, past and present, for their contribution in shaping me. I would like to thank my current institution, Purdue Northwest University, Dean Dietmar Rempfer, and Professsor Chenn Zhou for graciously allowing me to work towards completing my degree.

1.0 Introduction

In this research we address two distinct problems associated with system reliability. Both problems are focused on the design stage. First, it is important to select each component in the system (along with its associated reliability) so as to ensure that the system reliability exceeds some minimum desired level. At the same time, the reliability of each component comes with a cost associated with it. Hence it is important that simultaneously the overall cost of the system is also kept at a minimum. Second, once reliability allocation of the system has been addressed at the design stage, tests are needed to ensure that the system meets reliability requirements prior to it being deployed in the field. These tests may be at the system level where the whole system is assembled and tested, at the component level where only the components are tested and an inference on system reliability is made, or a combination of the two. These tests are often costly and hence it is critical that they be optimally designed. This dissertation addresses these two problems of reliability allocation and testing for reliability demonstration. In the following sections, we elaborate on both problems.

1.1 Reliability Allocation

1.1.1 Problem Background

In reliability allocation problem, one wishes to determine the desired reliability levels for the components that make up a system, given the overall system configuration and possible side

constraints. Most nontrivial systems that consist of many components tend to be expensive, and it is important that these systems sustain their operations for a long time. In this regard, the reliability of each component plays a very significant role in determining the overall reliability of the system. The reliability level of each component in turn, comes with a cost, and the specific choices for each of the components results in a system of a certain reliability with an associated cost. The reliability allocation problem thus assumes great importance where the cost of the system has to be minimized while guaranteeing high reliability for the system, or conversely, where the most reliable system has to be built given a certain budget.

There are two major thrusts to this research on reliability allocation. First, it will consider discrete data sets for cost and reliability of components and address the reliability allocation problem in a system design context. This is distinct from prior research which has focused more on functional relationship between reliability and cost. Second, new integer programming (IP) formulations are developed for the problem. These formulations are nonlinear (except in the case of a simple Series system or a simple Parallel system) primarily due to the complex expressions for reliability. Linear relaxations are developed, and solution procedures are developed based on (a) integer programming, (b) simulated annealing and (c) evolutionary algorithms. Since the problem formulations are integer in nature, an attempt is made to find optimal solutions based on integer programming. However, the problem formulations are nonlinear in nature, which makes it very challenging to develop optimum-seeking methods. Thus, even though for some specific systems, an integer programming approach will work, for more general systems, heuristic approaches are more suitable. The heuristic approaches developed in this research are more broadly applicable to many variety of problems. The computing time for a heuristic solution is also a significant factor in choosing these methods.

A system typically represents an end product designed for use by a customer. Sometimes it may also represent a unit which becomes an integral part of a larger system. A configuration for the system is defined based on the functionality of the components within it. For example, a system is known as a Series System if it fails when any one of its components fails. Similarly, a system is called a Parallel System if it fails only when all of its components fail. A system with a series connection of parallel subsystems is called a Series-Parallel (SP) system. Similarly, a system with a parallel connection of series subsystems is called a Parallel-Series (PS) system. A *K*-out-of-*N* system is one which will function if at least *K* out of a given *N* components function. Configurations for some common systems are shown in Figure 1. In the case of 2-out-of-3 system depicted in Figure 1, the component numbered 1 in both subsystems represent same component. In the case of Series-Parallel system and Parallel-Series system shown in the Figure 1, the components numbered same (for example component numbered 1) in different subsystems represent different components.

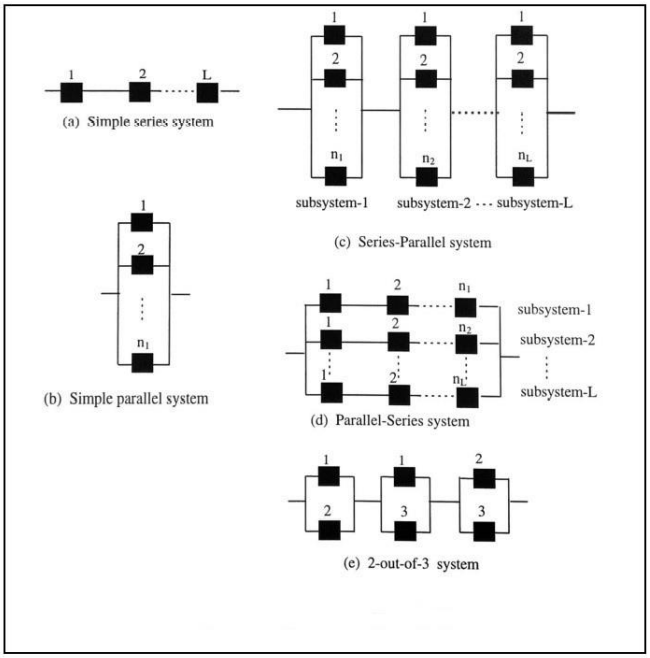

Figure 1: System configurations

For a given system configuration, an allocation problem is typically defined as determining the reliability values of all of its components while optimizing some objective such as the cost or system reliability and satisfying any specified structural or functional constraints (such as weight/volume restrictions, minimum required reliability value for the system, and/or specified limits on the available budget).

1.1.2 Some Applications

System designers are required to consider the reliability of the system as one of the main factors during the design process. In the competition for higher quality, reliability considerations at the early design stages acquire new dimensions of importance, but these must also be traded off against cost. As an example, consider a missile development program. Such programs are usually very expensive, and a very high reliability of the end product is required for strategic reasons. One must thus determine a highly reliable design but one that is also cost effective. As another example, consider the design of a long-distance gas supply line (for example, from Texas to New England). Since gas cannot be transported for long distances via pipe lines based on simple gradients, motor pumps are used at regular intervals to lift and pump gas into a continuing pipeline. These motor pumps are expensive, and their placement within the transportation network needs to be designed appropriately to ensure a highly reliable system, but one that is also a minimum-cost one. As a final example, most large commercial and governmental organizations have several supply chain partners supplying key elements of their products. Since the products are designed and eventually assembled by the organizations while having to stay within some budget, they need to specify reliability requirements to the suppliers in order to receive high quality components at competitive

prices. Hence, it is important for these organizations to do reliability allocation calculations early in the design stage. In short, when quality control measures are incorporated into designing a system (or end product), we require reliability target values to be specified for its components in order to provide an assurance that the system/end product will have some guaranteed minimum reliability. Furthermore, we will almost always have other considerations (most typically cost, but possibly, other technological constraints) that must also be considered during this design process.

Following are a few selected examples from the literature. Abed et al. [2019] address the reliability of a Reduction Oxygen Supply System (ROSS) for a spacecraft. This is a complex system where the allocation of component reliabilities was determined based on the minimization of the total cost of the system. The authors used a genetic algorithm approach to solve the reliability allocation problem. As another example, Hu et al. [2018] address optimal reliability allocation of ±800 kV Ultra HVDC transmission systems. The system architecture comprises several subsystems in series. Some of these series subsystems are connected in parallel to each other, while others are connected in series to the system of parallel-connected series subsystems. The estimation of system reliability is itself a very complex process with many components involved. The authors use different methods to estimate system reliability and then adopt a genetic algorithm to optimize the allocation of reliability to components such that cost is minimized.

In summary, reliability allocation within complex multi-component systems lead to challenging problem formulations with many real-world applications. These have existed for a long time and will continue to exist in the future.

1.1.3 Problem Formulation

A generic formulation of the reliability allocation problem follows one of the following two forms:

Problem A1

Minimize C

subject to

$$R \geq R_S$$

$$g_i(.) \geq b_i, \qquad \forall i \in I$$

Problem A2

Maximize R

subject to

$$C \leq C_S$$

$$g_i(.) \geq b_i, \qquad \forall i \in I$$

Here R and C denote the system reliability and system cost respectively, while R_S and C_S denote the minimum system reliability requirement and maximum budget respectively. Constraints $g_i(.) \geq b_i$ represent other design considerations (e.g., volume/weight) that might be important and are problem specific.

Problem A1 can be viewed as a customer-induced problem since the reliability constraint originates from customer expectations. Thus, it minimizes the producer's cost while meeting customer requirements. On the other hand, Problem A2 can be viewed as a producer's model in that it maximizes the system reliability while meeting the producer's budget constraints. Customer

driven policies are generally more common and we will focus on Problem A1 in this work. The following sections discuss variants or generalizations of reliability allocation problems, viz. redundancy allocation and redundancy design.

1.1.4 Redundancy Allocation And Redundancy Design Problems

The *redundancy* allocation problem is a special case of the reliability allocation problem. For a given system configuration with known component reliability values, the system reliability may be readily computed in most cases. If this computed value is not satisfactory, then it can be increased by the addition of redundant components. Clearly component level redundancy is better than system level redundancy. Thus, for each component, additional redundant units (with the same reliability value) are added. In such a case, the problem becomes one of finding the optimum number of redundant units required for each component so that total cost is minimized while some minimum system reliability level is guaranteed (or reliability is maximized while some budget limit is not exceeded). The redundancy design problem is similar to the redundancy allocation problem, but in this case the reliability value for each component level is also a variable to be determined. This problem may be considered as a further generalization.

1.1.5 Prior Solution Strategies And Their Limitations

It is a known fact that any system can be represented either as a Series-Parallel (SP) system based on minimal cut sets, or as a Parallel-Series (PS) system based on minimal path sets (Leemis [1995]). Thus, it is not surprising to see that SP and PS systems have attracted the most attention

from researchers working on reliability allocation. Most of the previous work in this area can be divided into two broad categories. One emphasizes multiple choices for components, where each choice comes with known reliability and cost (and sometimes weight, volume, or other features). The problems, in general, are addressed via integer programming or dynamic programming. However, from a design point of view the specific problem addressed in most of these formulations has been the simpler case of redundancy allocation rather than reliability allocation. As is common in redundancy problems, most of these formulations assume that the reliability of every component in a subsystem is the same. This restriction is not useful from a design perspective. Nor does it give rise to any significantly easier solution procedures. The resulting integer programming formulations are usually nonlinear in nature (primarily due to the reliability expression) and the commonly recommended solution procedures do not guarantee optimal solutions.

The second category constitutes nonlinear (often, integer nonlinear) formulations where some functional form is assumed for the relationship between a component's reliability and its cost. It is generally true that a component's cost is an increasing function of its reliability; most researchers to date adopt exponentially increasing, closed-form functions to relate cost and reliability. However, in practice such functions are often unknown or difficult to construct. While such an assumption tends to make the optimization procedures easier, there is no compelling reason put forth as to why such a relationship is appropriate. On the contrary, Figure 2 shows, the cost of a component can be a nonconvex and sometimes discontinuous function of its reliability.

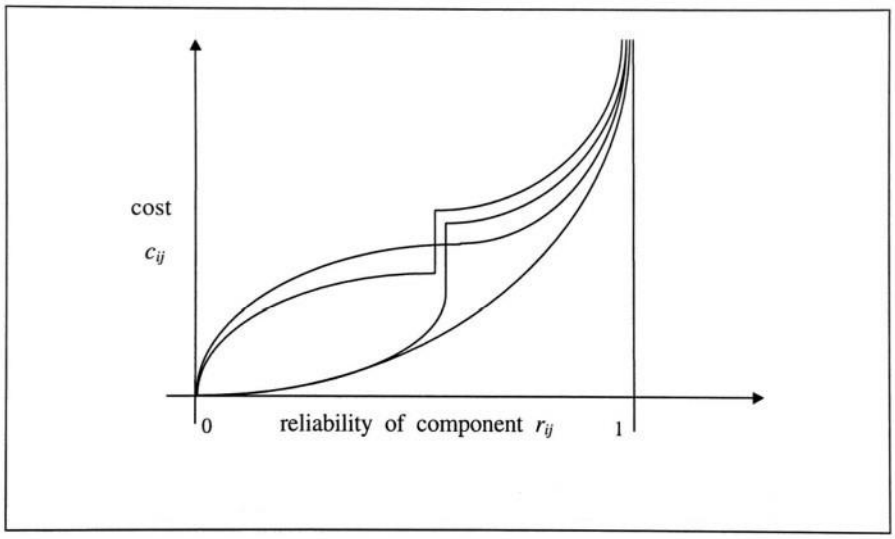

Figure 2: Cost vs. Reliability relationships for components

From a manufacturer's perspective, it is much more reasonable to provide a cost for some specified reliability of a component, as opposed to providing a precise (and typically, continuous) cost-reliability relationship. For instance, it might be possible to manufacture the component (which might be a small subsystem by itself) using different grades of materials, different part qualities, different configurations, different designs, or different levels of built-in redundancies. For each of these options, it would be much easier for the manufacturer to specify a reliability level and quote a price that depends on this level, rather than having to provide a precise mathematical expression for the cost of the component as a function of its reliability. Indeed, such a function may well be impossible to determine, especially since the different levels of reliability that are realizable for the component may be discrete or finite in number. Given this difficulty in deriving precise mathematical expressions for the cost-reliability curves, an alternative and more practical/realistic option is to consider discrete cost-reliability data sets for components. In this research we focus on such data and develop appropriate problem formulations.

The solution procedures to be considered as part of this research are (a) an optimum seeking approach, namely integer programming (IP), and (b) heuristics, namely (i) nested simulated

annealing (NSA) and (ii) evolutionary algorithms (EA). As part of the IP approach, problem specific valid inequalities are generated iteratively to arrive at the optimal solution. Such an approach guarantees an optimal solution but is usually very time consuming. Heuristics on the other hand yield "good-enough" solutions that are feasible but not necessarily being optimal. However, they do so much more quickly and with much lesser computational effort than with IP.

1.2 Test Plans For Systems And Components

1.2.1 Introduction

Every system must go through some testing process that assesses its competence before it is eventually deployed. The typical goal of the testing program is to demonstrate that the system will perform at an acceptable level of reliability for the mission for which it was designed. Reliability is typically expressed as a mean time to failure or as the probability of no failures over some specified mission time such as a warranty period or a deployment interval. System testing can be conducted in many ways, for example, (i) only components are tested and inferences on system reliability are made from that information and how the components are configured to form the system, (ii) the complete system is assembled and tested and inferences are then made on its reliability, (iii) some component testing as well as some system testing is done separately, and inferences on system reliability are then made from the combined results of these tests. It is worth noting that system-level tests are typically much more expensive than component tests because the entire system must be assembled before it is tested. They also tend to be more technologically complex and more difficult to instrument.

In addition to cost and complexity. component tests have other advantages such as (i) they take a shorter time to schedule, (ii) they can be done at individual locations by different teams, and (iii) a system need not be assembled until there is a minimum guarantee that it will perform well, based on component testing. The interested reader is referred to Rajgopal and Mazumdar [2001] for more details.

It is also worth mentioning that in addition to the advantages given above, system-based component tests are also useful in other situations. One example is systems that mix old and new components. Sometimes a system is an improvement over a previous version of the same system. Such systems are designed by combining new components and/or new subsystems together with others that have been successfully used before in earlier designs. In such situations, component testing to demonstrate a system reliability requirement is particularly effective. A component or a subsystem that has been used previously requires little or no additional testing and only the new components require additional testing. The same is true for systems which are evolving continuously. Also, for complex systems that have several subsystems assembled in different parts of the world, it makes sense to test individual subsystems to make an inference on overall system reliability. Once there is a guarantee of overall system reliability, the whole system can then be assembled.

Much work has been done on expressing the reliability of a system as a function of the reliabilities of its components, and on using knowledge on the component reliabilities individually to draw inferences on system reliability. There is abundant literature (e.g., Mann et al. [1974]) on interval estimates of system reliability based on component test data. However, the same cannot be said about the proper design of statistically sound, mathematically tractable and economically desirable component tests designed specifically for drawing inferences on system reliability.

Research that has directly addressed this aspect of test design will be discussed in the next chapter on literature review.

One must note that component testing does not necessarily preclude system testing altogether. There will be instances where system testing is necessary and warranted. Examples of such situations would be (i) if component failures are not independent, (ii) if interfaces between components that make up a system are unreliable, or (iii) even if a system designer is simply uncomfortable with not doing any system testing. However, even in such situations it might be possible to reduce the amount of (more expensive) system-level testing by combining it with the results of some (less expensive) component testing.

1.2.2 Problem Formulation

A general formulation for the component-test design problem is now provided (Rajgopal and Mazumdar [2001]). Suppose a parameter set $\boldsymbol{\theta_j}$ is associated with each component j of the system. This parameter set determines the reliability of the component. For example, suppose the component life time is exponentially distributed. Then the single parameter λ_j representing the failure rate for that component measures its reliability. In such a case $\boldsymbol{\theta_j} \equiv \lambda_j$. The system reliability R_S may then be expressed as a function of $\boldsymbol{\theta_1}$, $\boldsymbol{\theta_2}$, etc. To be precise let $R_S = f(\boldsymbol{\theta})$ where $\boldsymbol{\theta} = [\boldsymbol{\theta_1}, \boldsymbol{\theta_2}, \ldots, \boldsymbol{\theta_n}]$ is a vector of parameters, and the exact form of the function f depends on the system configuration (series, parallel, serial connection of parallel systems, etc.). Let C_j be the cost of testing component j; again, C_j will depend on the format of the test plan. Let us consider two sets:

$$S_1 = \{\boldsymbol{\theta} | R_S = f(\boldsymbol{\theta}) \geq R_1\}$$

1.1

$$S_0 = \{\boldsymbol{\theta} | R_S = f(\boldsymbol{\theta}) \leq R_0\}$$

1.2

Here R_0 is some specified value below which a system would definitely be considered unreliable, while R_1 (>R_0) is some minimal level of desired system reliability (the interval (R_0, R_1) is sometimes referred to as a zone of indifference). It may be noted that S_1 is the set of all combinations of values for the component reliability parameters that lead to a system with a definitely acceptable reliability, while S_0 is the set of all combinations of values that lead to a system with a definitely unacceptable reliability. Then the problem of minimizing total test costs subject to constraints on Type 1 and Type 2 error probabilities is stated as follows.

Problem P: $Minimize\ Z = \sum_j C_j$

s.t. $Minimum_{\boldsymbol{\theta} \in S_1}\{Prob(\text{Accept the system})\} \geq 1 - \alpha$

$Maximum_{\boldsymbol{\theta} \in S_0}\{Prob(\text{Accept the system})\} \leq \beta$

$0 < \alpha, \beta < 1$

The values of $R_0, R_1, \alpha\ and\ \beta$ are typically specified for specific applications. The constants α and β are suitably low, pre-assigned values representing bounds on Type 1 and Type 2 error probabilities, respectively. These probability requirements are similar to the ones encountered in many conventional testing plans; for example, those listed in the Department of the Navy document MIL-HDBK-781D (1987).

For a given system configuration, Problem P above is a two-stage optimization problem. In the "inner" stage, assume that we are given a vector of test times ***t***. The probability of accepting

the system will, in general, be some function of (i) the vector $\boldsymbol{t}$, and (ii) the reliability parameters of the components of the system (as represented by the vector $\boldsymbol{\theta}$). Now, there may be many *different* vectors $\boldsymbol{\theta}$ in the set S_1 (each representing some combination of component reliabilities) that lead to an acceptable system. Hence, for a given $\boldsymbol{t}$, the probability of system acceptance should be at least (1-α) for <u>all</u> $\boldsymbol{\theta} \in S_1$. Equivalently, the *minimum* probability of acceptance across all $\boldsymbol{\theta} \in S_1$ should be at least (1-α). This is the first constraint. Similarly, the second constraint imposes a restriction on the *maximum* probability of accepting an unacceptable system. Thus, given $\boldsymbol{t}$, the LHS of these two constraints lead to two optimization sub-problems (in $\boldsymbol{\theta}$) over the sets S_1 and S_0 respectively. If the optimum values of these two subproblems are respectively $\geq$ (1-α) and $\leq \beta$ then the corresponding vector of test times $\boldsymbol{t}$ is feasible. The "outer" stage optimization problem aims to find among all such feasible $\boldsymbol{t}$, the vector that also minimizes the objective function.

1.2.3 Difficulties Associated With The Optimization Problem

Problem P represents the optimization problem in its most general form. The parameters, the reliability expression, the costs for testing, the sets of parameters leading to definite acceptance or rejection, and the mathematical representation of the two constraints in the optimization problem are all determined by not just the system configuration and the failure time distributions but also by the test format adopted. These formats might be quite different. For instance, a life test format could be where each component type is tested with replacement for some fixed length of time and the number of failures observed. Another format could be that a fixed number of components may be tested to failure and the times to failure observed. A third format could be a binomial type test where a fixed number of components of each type are subject to a pass-or-fail type test, and the

number of successful trials observed. In each case, the values observed at the component level must be combined into some test statistic, and then translated into a decision rule for system acceptance or rejection. The choice of the best test statistic and the corresponding decision rule are both open research issues. Realistically, the rules must be (i) analytically tractable, and (ii) plausible for the specific system being considered. The test costs will also depend on the test format. The cost structure could be defined in terms of cost per unit time on test, or in terms of cost per unit tested, depending on the format of the test.

The preceding discussion points to the complexity of the problem. There are two distinct considerations here: statistical and optimization. Statistical considerations pose numerous challenging issues such as identifying the underlying failure time distributions, establishing the relationship between component and system reliabilities, selecting an appropriate test statistic and developing an appropriate acceptance rule. The optimization considerations include how we formulate the optimization problem in terms of the component failure time distribution and test parameters, developing algorithms to solve these optimization problems, deriving suitable approximations when the problem is intractable, and incorporating problem specific information into the solution methodology.

Prior knowledge is another important factor. One could sometimes have reliable estimates for the component failure rates based on prior experience, while at other times one may have no such knowledge at all. When prior estimates are available, the optimization procedure could possibly exploit such prior knowledge to develop a plan that will be less costly. Another common situation is that the interfaces that enable components to be assembled into a system might not be perfect. In the absence of perfect interfaces, one might be required to perform at least some system level testing and there is a need to investigate conditions where such system level testing is

necessary. In this research we will address optimum test plans for a series system where interfaces are not perfect.

1.3 Research Objectives

The principal objectives of this research are:

(i) For the reliability allocation problem:

- Develop new integer programming formulations for ***SP*** and ***PS*** systems.
- Develop linear relaxations for the problem formulations addressed.
- Evaluate the strength of the linear relaxations.
- Develop problem-specific, iterative solution procedures by identifying valid inequalities.
- Develop heuristic solution procedures that can be adopted for most generic problems.
- Summarize the findings and evaluate the degree of usefulness of the solution procedures developed in this research.

(ii) For optimum test plans:

- Develop problem formulations for a series system with imperfect interfaces.
- Consider interface reliabilities when deriving solutions
- Summarize the findings and evaluate the usefulness of the solutions obtained.

1.4 Organization Of This Document

In this research we address (a) the reliability allocation problem and (b) the reliability demonstration problem The reminder of the document is organized as follows.

Chapter 2 comprises two parts. In the first part the research literature pertinent to the reliability allocation problem and solution procedures is discussed. The second part contains research literature pertinent to optimum test plans for reliability demonstration.

Chapter 3 discusses the reliability allocation problem formulations and solution procedures. The integer programming approach and valid inequalities for the problem are also described in this chapter.

Chapter 4 discusses solutions to the reliability allocation problem based on two heuristic approaches viz., (i) simulated annealing and (ii) evolutionary algorithm. A novel nested simulated annealing algorithm is developed for this problem. Such nested simulated annealing algorithms can be generalized and may be used for a variety of other problems. A solution to the reliability allocation problem based on an evolutionary algorithm considers a special penalty function that accelerates the solution procedure and yields quick solutions.

Chapter 5 discusses the optimum test plans for systems and components. In this, a formulation of the problem is presented that considers Type 1 error as well as Type 2 error simultaneously. Interface reliabilities are also taken into account and solutions are obtained for various conditions encountered.

Chapter 6 summarizes the research with a final analysis of all solution approaches adopted in this research.

2.0 Literature Review

In this chapter, a literature review on the two problems addressed is presented. The literature review focuses mostly on the problems addressed in this research, various formulations used to address these problems in earlier research along with any limitations, and solution procedures adopted. We also review the relevant literature associated with the specific heuristic methodologies used in this research to solve the problems.

2.1 Reliability Allocation

2.1.1 Introduction

Reliability allocation is a very old problem but still generates a lot of interest within the research community. The problem and the formulation evolved over many years and still continues to evolve. For example, Elegbede et al. [2003] address the reliability allocation problem in their work. Malaiya [2008] has also studied the reliability allocation problem to apply it to a software reliability application. His work is clearly inspired by the formulations used in this research. Yalaoui et al. [2005] have used the Tillman functions for cost-reliability relationships and derived solution procedures for a system that is a parallel connection of many series systems. An overview of optimal reliability allocation may be found in the paper by Kuo and Wan [2007]. They stated that "*Optimal reliability design has attracted many researchers who have produced hundreds of publications since 1960. Due to the increasing complexity of practical engineering systems and*

the critical importance of reliability in these complex systems, this still seems to be a very fruitful area for future research." Cruz [2016] has noted in his report that applying reliability allocation techniques without understanding their limitations and assumptions can produce unrealistic results. More recently, Si et al. [2019] propose a generalized Birnbaum importance measure (GBIM) to quantify the contribution of individual components to system reliability improvement by considering reliability range, manufacturing complexity, and technology feasibility. GBIM possesses several unique features in terms of guiding system reliability optimization. The authors develop a GBIM-based genetic algorithm to solve a type of optimal reliability allocation problem. All of this work clearly indicates that the long-standing reliability allocation problem remains an important one that continues to attract the attention of researchers in the field.

2.1.2 The Reliability Allocation Problem

As stated in Chapter I, the generic formulation of the reliability allocation problem follows in one of the following two forms:

Problem A1

Minimize C

subject to

$$R \geq R_S$$

$$g_i(.) \geq b_i, \qquad \forall i \in I$$

Problem A2

Maximize R

subject to

$$C \leq C_S$$

$$g_i(.) \geq b_i, \qquad \forall i \in I$$

Here R and C denote system reliability and system cost respectively, while R_S and C_S denote the minimum system reliability requirement and maximum budget respectively. Constraints $g_i(.) \geq b_i$ represent other design specifications (such as volume/weight) which are problem specific. In the following subsections we discuss several variants of the above models that have been addressed by prior work and contrast the proposed research with respect to these variants.

2.1.3 Redundancy Allocation

Redundancy allocation is a classical problem that was first comprehensively discussed by Tillman et al. [1980] and more recently by Kuo and Wan [2007]. The primary goal of this model is to achieve reliability goals by using redundancy at the component level. The initial model was presented by Tillman and Littschwager [1967] and was subsequently adopted by Tillman et al. [1968], Sharma and Venkateswaran [1971] and many others (refer to the second chapter of Tillman et al. [1980]). In this model they assumed that the reliability of each component is known and the decision to be made is the number of redundant units to be placed for each component. Their formulation for this model expressed the reliability expression for a series system with L different unit types (or stages) connected in series as $R = \prod_{j=1}^{L}(1 - (1 - R_j)^{x_j})$, where x_j represents the number of redundant units to be placed at stage j. In this representation the reliability of each unit in a stage j is assumed to be the same and is equal to a known quantity R_j. In addition, the cost function with respect to the number of redundant units is presented as $C = \sum_{j=1}^{L} c_j(x_j + e^{\left(\frac{x_j}{4}\right)})$.

The first drawback of this model is that there is no compelling reason presented for the assumption that all redundant units in a stage should have the same reliability. A perfect example to contradict this assumption would be where a primary power station with high capacity is used for power supply, while a secondary power station with relatively low capacity (and perhaps lower reliability) is used as a redundant unit to boost the reliability of the system. Thus, in a design context, where we wish to determine the reliability level for each component, this model is not applicable. More importantly, no explanation is given for the cost expression for the system that is presented. From the expression it appears that it is used as some sort of heuristic approximation. The first term in the cost, $c_j x_j$, represents the cost directly proportional to the number of redundant units. However, the second term, $c_j e^{\left(\frac{x_j}{4}\right)}$, which is interpreted as the cost of interconnecting parallel elements, represents an exponential increase in the cost due to more redundant units. This gives the impression that it is used to discourage solutions where a large number of redundant units would be adopted in a single stage. Such a cost expression may be applicable for certain cases and may not be applicable to others. However, with such a cost structure the generality of the model is lost.

2.1.4 Redundancy Allocation In A Design Context

In order to make the earlier model work from a design point of view, Tillman et al. [1968] proposed a second model. In this model, the decision to be made is still with respect to the number of redundant units in each stage in a series system, but the reliability level of each component also has to be determined. The reliability expression for a series stage is the same as earlier, i.e., $R = \prod_{j=1}^{L}(1-(1-R_j)^{x_j})$ but in this expression both R_j and x_j are both unknown. However, the

reliabilities for all units in the same stage are still assumed equal. In this model the cost expression used was the same as earlier, i.e., $C = \sum_{j=1}^{L} c_j(x_j + e^{\left(\frac{x_j}{4}\right)})$, where c_j is cost per component at the j^{th} stage which is a function of R_j. This cost $c_j(R_j)$ is expressed as a decreasing function of the component failure rate, i.e., $c_j(R_j) = \alpha_j \left\{\frac{-t}{\ln R_j}\right\}^{\beta_j}$ where α_j and β_j are constants representing characteristics of components at the j^{th} stage, while t represents the operating time during which the component at stage j does not fail.

The main drawbacks of this model are similar to those mentioned for the earlier model, since the models are essentially identical except that the R_j are assumed to be unknown in the second model. In addition, the authors fail to provide any specific reason or justification for the complex cost versus reliability relationship assumed in this model. As discussed in Chapter I, it is an accepted notion that cost is an increasing function of reliability; however, there is no compelling reason for it to be a convex function. More importantly, from a practical point of view such closed form functions are not easy to obtain. It may be possible that a particular functional form can be constructed for a particular application, but the use of the functional form would then be limited only to that application (or perhaps very similar ones).

2.1.5 Reliability Allocation In The Context Of Assembly

This problem is different from the one addressed by the earlier models. Here the goal is to place components into known positions within a system configuration. The reliability values for all components are known; however, the system reliability is dependent on the positioning of components in the system. Hence the model has an objective of maximizing the system reliability while placing components in their respective positions. In certain systems all components are

functionally similar and thus any component can be placed in any position of the system configuration. This problem for various systems is addressed by Kontoleon [1979], Malon [1984], El-Neweihi et al. [1986, 1987, 1988], Papastavridis and Sfakianakis [1991], Zuo and Shen [1992] among others. El-Neweihi et al. used the theory of majorization and Schur convex functions to obtain optimal allocations for several systems including Series-Parallel and Parallel-Series systems. Other researchers approach a similar problem for other systems using heuristic approaches based on concepts of reliability importance, where the importance of a component is defined as a partial derivative of the system reliability with respect to the component's reliability.

The problem addressed with this model is applicable in situations where all components are functionally similar and it is assumed that their placement within the system configuration has no bearing on cost. A more general version of this problem would be one where there is a cost associated with the assignment of a particular component to a particular position. If cost is considered in this model, the problem would become a more traditional optimization problem and the concepts of reliability importance and majorization theory offer little assistance in solving the problem. The following example will justify the need to consider cost.

Consider the example of the design of a long-distance gas supply line (such as from Texas to New England) mentioned in Chapter I. In this example the motor pumps are functionally similar and thus each pump can be placed at any of the fixed locations. However, they might come from different manufacturers and might be associated with different reliability and cost values. All motor pumps would require a proper foundation and a platform to place them, which in turn might be dependent on the location. Hence the cost of fixing a motor pump would directly depend on the location. Hence it is important that cost considerations be taken into account in these models.

2.1.6 Motivation For Our Research

The inadequacies of earlier models stem from two factors: (1) the problems associated with complex system reliability expressions in terms of the decision variables and (2) the inability to handle the wide range of problems that are concerned with reliability allocation as a part of the design. In this research we attempt to address these issues. To be more specific, with respect to earlier research the solution procedures suggested are mostly heuristics since the formulations are very complex and none of the available integer programming procedures suit these formulations. In fact, Kececioglu [1991] describes existing allocation methods with respect to redundancy as "poor approximations at best". For example, Sharma and Venkateswaran [1971] suggested a heuristic solution to the redundancy allocation problem based on incrementing redundant components in the most unreliable stage at each successive iteration until the reliability requirement is met or a constraint is violated. Agarwal et al. [1975] proposed a variant of the same approach where they propose a different solution procedure for selection of a stage where a redundant component is added. Several other heuristics are listed in detail by Tillman et al. [1980].

Dynamic Programming was used for solving the redundancy allocation problem by Katelle [1967], Woodhouse [1972], and others (refer to Tillman et al. [1980] for a summary of dynamic programming in redundancy allocation). A common criticism leveled against dynamic programming is that when multiple constraints are present the dimensionality of the problem increases. Hence most of the dynamic programming applications to reliability allocation are limited to single constraint problems and unfortunately, most system design problems have many functional constraints in addition to reliability requirements. Scores of other approaches from the literature are recorded and summarized by Tillman et al. [1980] for redundancy problems. It is surprising to see that almost all of them are heuristic solutions and are applied to only simple

systems such as a series system. For the redundancy design problem, the formulation is even more complex given the fact that both R_j and x_j are variables (with x_j also being required to be integer) and optimization algorithms become very difficult to develop. This is reflected by the fact that numerous heuristic solution procedures have been proposed (e.g., Agarwal et al. [1975], Nakagawa and Nakashima [1977], Tillman et al. [1977], Cateanu et al. [1986], Coit and Smith [1996], Jacobson and Arora [1996]). No attempt has been made to look at alternative formulations where better solution procedures can be developed so that the model and solution procedures become widely applicable for more general systems. While the development of heuristic solution procedures is a logical approach given the complexity of the problem, the formulations themselves need some attention. Since these formulations are difficult to solve optimally even for a simple series system (where redundancy is sought for stages that are in series), extending them to more complex systems offer little hope. In this regard, the approach taken by Bulfin and Liu [1985] offers interesting insights and in some sense inspires the approach proposed in this research. Instead of considering the number of redundant components as a variable (i.e., x_j), they discretize this into binary variables and formulate a 0-1 integer programming problem. This research uses an analogous approach by assuming discrete cost vs. reliability relationships, which is often a more reasonable approach than assuming a closed form functional relationship.

Consider system design problems such as transportation problems, supply-chain problems or network design problems. Most of these problems are addressed either by network-based solutions or integer programming-based solution procedures. However, when reliability considerations are considered for these problems, they become much harder to approach. The earlier reliability models offer little assistance with regard to such design problems where the reliability requirements are desired. There is no effort by any researchers to the best of our

knowledge to consider reliability requirements for these problems thus far. This may be partly attributed to the fact that the computation of reliability for a system given the reliability for each component is itself a very hard problem and no known polynomial algorithms are available for general systems. However, many systems of practical significance (viz. series-parallel, parallel-series and systems known as series-parallel-reducible-networks) have polynomial algorithms to derive reliability of the system from given component reliability values. The solution procedures developed in this research can be readily applied to any design problem for such systems that require reliability considerations.

The model proposed in this research is based on discrete data sets for reliability and cost. It is a generally accepted notion that a component's cost is an increasing function of its reliability. Discrete data has been used in the past although the term "discrete data" was not in use at that time. Fyffe et al. [1968] used a slightly different formulation for redundancy allocation where they have alternative choices for a component in a stage. Once a choice is made, the problem is then to determine the number of redundant components of that choice. The problem with this model is that every component in a stage would have same reliability/weight/cost; in other words, duplicate copies of the same component are added as redundant. If they had allowed the freedom for all redundant components in a stage to have their own characteristics (such as reliability/weight/cost etc.), the model would have much more general appeal. For example, in the power plant example from the previous chapter, a redundant plant would be of less capacity with relatively less reliability.

In this research, our model exploits this fact and allows for each component to have its own characteristics so that the most suitable alternative is chosen via the optimization procedure. For example, in the electronic industry, components are manufactured in advance and are available

from a variety of sources. Thus, one can choose a particular component from a finite number of sources with known reliability and cost values. The model based on discrete data can be used for all of the redundancy problems mentioned above and in addition, the solution procedures recommended in this research easily accommodate additional constraints such as functional requirements based on the nature of the system under consideration. When only discrete data is available, this model can be used directly. However, it can still be used when a functional relationship is available for cost and reliability, because we can always draw discrete datasets from the relationship.

We next provide some background on the methodologies we develop here for solving the reliability allocation problem.

2.1.7 Integer Programming

In this research work, we formulate the reliability allocation problem as a 0-1 integer Programming problem. For complex systems, the problem formulation is nonlinear in nature. For specific systems we tried to identify linear relaxations of the formulations. Once a linear relaxation is identified, an iterative procedure is recommended to eliminate infeasible solutions resulting from solutions to the relaxation. Some of these results are given by Majety et al. [1999]. Even 0-1 linear integer programming problems for simple Series Systems (or Simple Parallel systems) are NP-hard Knapsack problems. The iterative procedure, developed in this research by solving ILP relaxations at each iteration, is very time consuming and computationally very taxing. However, the solution procedure is presented in the hope that better solutions will follow in future. More details are given in Chapter 3.

From a practical perspective, these hard problems can be addressed using various heuristic approaches such as Simulated Annealing, Genetic Algorithms, Evolutionary Algorithms or Tabu Search etc. We will explore two of these approaches viz., Simulated Annealing (SA) and Evolutionary Algorithms (EA), in this research.

2.1.8 Simulated Annealing

Many heuristic procedures such as SA (Brusco and Jacobs [1993], Kirkpatrick et al. [1983]), Genetic Algorithms (Bean and Hadj-Alouane [1992], and Tabu Search (Glover [1989a, 1989b]) have been proposed for difficult combinatorial optimization problems. SA is a heuristic algorithm for obtaining good, although not necessarily optimal solutions, to optimization problems. Brusco and Jacobs [1993] list many combinatorial problems for which SA has been successfully applied. Those include school time-tabling, multilevel lot sizing and cyclic staff- scheduling. Kanagaraj and Jawahar [2009] are inspired by our research on nested simulated annealing (Majety and Smith [1996]) and develop a simulated annealing algorithm for their optimal supplier selection problem using the reliability-based total cost of ownership model, which is also a nonlinear integer programming formulation.

In a traditional SA approach, one starts with a feasible solution and identifies a feasible neighboring solution. If this new solution improves the objective function, it is immediately accepted, and a move is made to that solution. If not, the new solution is accepted probabilistically based on some annealing schedule. Eglese [1990] mentions that the efficiency of an SA algorithm depends on the definition of the feasible neighborhood of the current solution. However, instead of defining a feasible neighborhood, if the problem has a difficult-to-satisfy constraint set, the addition of an exterior penalty function to the objective function may be used. In this research, we

used a nested SA algorithm instead of a penalty function. In principle it is quite similar to the penalty function approach, but the difference is that the penalty is applied to the acceptance probability for an infeasible solution rather than to objective function, as is done normally. Our initial experiments with the nested SA always resulted in feasible final solutions (not always achieved with penalty functions) and thus we adopted the nested SA instead of a regular penalty function approach. More details are presented in Chapter 4.

2.1.9 Evolutionary Strategy Approach

Evolution strategy [ES] refers to an algorithm that tries to simulate the evolution process. A detailed description of the method may be found in Back et al. [1991]. Briefly, in the simulation of an evolution process one tries to find links between the characteristics of an offspring and its parents. The usual procedure adopted is as follows: the problem is encoded in terms of its variables (viz. characteristics). An objective is defined as a fitness function value based on these variables. For the problem, an initial set of solutions is generated, and they are treated as parents. A new set of offspring is then generated from these parents. Now, from the set of offspring solutions and parent solutions, a fixed number of fittest solutions are accepted and set as the new set of parents for the next generation. This process of creation and selection continues until a stopping criterion is met. A more detailed description of this method and how it is adopted for the reliability allocation problems of this research are given in Chapter 5.

2.2 Optimum Test Plans

Gal [1974] was the first to address the area of system based component testing. His initial work considered an arbitrary coherent system composed of *n* different component types with independent failures, and his plan called for each component type *j* to be tested for t_j time units, with the system being labeled acceptable if no failures were observed during the prescribed testing period for each component type. He formulated the problem as

$$Minimize\ C(t) = \sum_{j=1}^{n} c_j t_j$$

$$\text{s.t. } \Pr\{\text{accept system when } R_S \leq R_0\} \leq \beta \qquad \textbf{2.1}$$

where

c_j is the cost of testing component *j* per unit time,

R_S is the system reliability for a unit time period,

R_0 is a value such that any system with reliability lower than this value would be deemed as definitely unacceptable, and

β is a suitably low pre-assigned probability.

Under the assumption of exponentially distributed component lifetimes, he derived a general procedure for obtaining the optimum test times. He also provided specific examples for a few common system configurations. In Gal's work, the acceptance rule was very demanding and it could result in a good system being unnecessarily rejected. This issue was addressed by Mazumdar [1977], who considered a formulation identical to that considered by Gal, but with another probability constraint in keeping with the standard statistical practice for determining the sample size. This constraint is:

$$\Pr\{\text{accept system when } R_S \geq R_1\} \geq 1 - \alpha \qquad \textbf{2.2}$$

where

R_1 ($>R_0$) is a value such that any system with reliability greater than this value would be deemed as definitely acceptable, and

α is some suitably low pre-assigned probability.

Mazumdar [1977] assumed that component testing took place with replacement. Let X_j denote the number of failures of component *j* that occur when it is tested for t_j time units. He then proposed the following alternative to Gal's acceptance criterion, which he referred to as the *sum rule*: "accept the system if $\sum_j X_j \leq m$," where m is an integer-valued decision variable. He then showed that with this rule the optimum component test times are the same for each component irrespective of the testing costs. In his original work, Mazumdar [1977] did not explicitly prove that a feasible *m* was guaranteed to exist. It was subsequently shown by Rajgopal et al. [1994] that when this criterion is used, there exists an m^* such that both (2.1) and (2.2) will be satisfied for all values of $m \geq m^*$, as long as $\alpha + \beta < 1$.

In the following subsection we review research on this topic of system-based component testing that specifically addresses series systems, since that is the focus of the work in this dissertation. Subsequently, in Section 1.1.2 we also briefly overview other research related to this topic.

2.2.1 Testing Of Series Systems

In a series system, every component must work for the system to work. Yan and Mazumdar [1986] were the first to address series systems and studied three different test procedures based upon (1) the *total* number of failures across all component types, (2) the number of failures for *each* component type and (3) the maximum likelihood estimator of system reliability. They showed that the first and third procedure led to identical results. They also concluded that for similar levels of protection from Type I and Type II errors, these procedures were generally superior to the second case from the point of view of costs. More interestingly, they showed that in an optimum policy, all component types need to be tested for the *same* length of time regardless of the test costs. This is intuitive because a series system is only as good as its weakest component and in the absence of any prior knowledge, we would need to treat each component equally.

Easterling et al. [1991] address a series system where the individual component failures follow a binomial distribution. More importantly, this work presented a detailed discussion of the need for *system-based* component test plans by demonstrating the drawbacks associated with the ones customarily used in practice, where it is common to "allocate" the required reliability for a system among its individual components and then independently require each component to show this level of reliability with some specified level of confidence. Using system O.C. curves, the authors show how this could lead to probabilities of Type 1 and Type 2 error that could be vastly different from the advertised values. They also presented a justification for the use of the so-called *sum rule* with binomial failure data, that Mazumdar [1977] had introduced earlier. Several years later, Mazumdar and Rajgopal [2000] set up the mathematical program for computing the maximum likelihood estimator for system reliability, and derived the Karush-Kuhn-Tucker

conditions for the optimum. These conditions yield expressions that involve the sum of the number of failures observed. The results also indicated that for a series system with binomial failure data, the sample sizes for each component type should be the same (just as with exponentially distributed failure times).

The result of equal test times can be intuitively unappealing because designers often have some idea about how reliable a component is, e.g., they might know based on prior experience that component X is likely to be more reliable than component Y, or that component Z is known to have some minimal reliability level. The next major extension for series systems addressed this scenario, where it was assumed that *a priori* information of some kind is available on the failure rates of the individual components (Altinel [1992], Altinel [1994], Rajgopal and Mazumdar [1995]). Specifically, it was assumed that each component failure rate λ_j had a known upper bound u_j. Interestingly, with *a priori* knowledge the optimal test plan need not require all components to be tested equally, but rather, for times that depend on the magnitude of the upper bound u_j as well as the unit test cost c_j. Thus, a component j with a smaller value of u_j and a higher value of c_j will require less testing than one with a larger u_j a smaller c_j. A detailed discussion of an algorithm along with computational results may be found in Rajgopal and Mazumdar [1995]. Soon after, Raghavachari [1998] proposed a simpler procedure to solve the same problem. This method was based on results from linear programming duality; and the same numerical results were obtained but with far less computational effort.

Rajgopal and Mazumdar [1995] also provided the formulation and a solution procedure for the case where each component's failure time follows a gamma distribution with a common, known shape parameter and unknown scale parameter. Once again, they considered both bounded and unbounded failure rates. For the latter case, they developed exact procedures which as

expected, yielded equal test times for all components. The former case is much more complex and the authors used a Normal approximation to formulate a nonlinear optimization problem along with a cutting plane strategy where a sequence of nonlinear programs were solved at each iteration. The same authors (Rajgopal and Mazumdar [1997]) also studied the series system with component failure times that follow a Weibull distribution. Once again, they used a Normal approximation to develop a solution procedure for the resulting formulation. The results for both the gamma and Weibull distributions were consistent with those for the exponential distribution.

Rajgopal and Mazumdar [1997] were also the first to introduce the notion of imperfect interfaces between the components of the system. All of the prior research had assumed that interfaces were perfect and when the system failed it was only because a component had failed. In practice, systems often fail at an interface (e.g., a weld or a connecting wire); in fact, this is one reason why many practitioners tend to use system test as opposed to just relying on component tests. The research presented in Chapter 5 (Rajgopal et al. [1999]) addresses this issue by combining component and system testing.

In other related work on series systems, Sankar and Vellaisamy [2000] develop a two-stage test plan for a series system, similar to double sampling plans that are commonly used in acceptance sampling. The objective is to try and reduce testing effort while still providing protection from Type I and Type II errors. The approach has two parameters m_1 and m_2. In the first stage, each component j is tested for t_{1j} units of time and the number of failures X_{1j} observed. If $\sum_j X_{1j} \leq m_1$ the system is immediately accepted, and if $\sum_j X_{1j} > m_2$ the system is immediately rejected. If $m_1 \leq \sum_j X_{1j} \leq m_2$ further testing is conducted. In this second stage, each component j is tested further for t_{2j} units of time and; suppose the number of failures at this stage is X_{2j}. The system is finally accepted if $\sum_j X_{1j} + X_{2j} < m_2$ and rejected otherwise. Note that the total test

cost is not deterministic anymore; the authors therefore minimize an expression for the maximum *expected* cost subject to the usual constraints. They also address both cases with and without prior information on failure rates. For the latter case, they adapt the approach proposed by Raghavachari [1998], and for the former case, they use a meta-heuristic genetic algorithm to solve the optimization problem. The authors show that the maximum average cost is lower than the minimum cost for the single stage plan (Rajgopal and Mazumdar [1995]) by a little over 10%.

2.2.2 Testing Of Parallel Systems

We now briefly discuss work in this area beyond series systems. Parallel systems (where at least one component should work for the system to work) are generally more complicated than series systems because of how system reliability must be expressed as a function of component reliabilities. Thus, work in this area has exclusively addressed parallel systems with exponentially distributed failure times. Yan and Mazumdar (1987a) were the first to examine such systems where they compared several plans using Type I censoring. They examined a system of n components in parallel where the j^{th} component had an exponentially distributed time to failure with parameter λ_j and failures are independent. They also made the assumption that each λ_j was "much smaller" than 1, i.e., that the system was "highly reliable." This allowed them to approximate the system reliability expression as

$$R_S = 1 - \prod_j \left[1 - exp(-\lambda_j)\right] \approx 1 - \prod_j \lambda_j \qquad \textbf{2.3}$$

This simplification allows the optimization problem to become tractable. Yan and Mazumdar [1987b] also looked at a parallel system with Type II censoring. Once again it was

assumed that component failure times are independent and exponentially distributed, and that unit test costs for each component type are different. Rajgopal and Mazumdar [1988] present a slightly different acceptance criterion for the same design problem, based on the sum of the logarithms of the test times. This vastly simplifies the process of computing the critical value for the test statistic. The authors also provided several approximate procedures for this computation. With respect to other system configurations Mazumdar [1980] studied a serial connection of parallel subsystems. Units in subsystem j are assumed to be identical with independent, exponentially distributed failure times with parameter λ_j. The usual sum-rule is used for deciding on whether the system should be accepted or not. The results indicate that the optimal test times are independent of the unit test costs.

Altinel and Ozekici [1997] consider the situation where component failure rates might not stay fixed over time. For example, it might be that the failure rate is small during the early stages of deployment but the component degrades with use. The authors formulate the optimization problem as a semi-infinite linear program and use essentially the same cutting-plane algorithm presented by Altinel [1994] to solve the problem. They also provide an example of a serial connection of two subsystems working in three different fixed environments. Altinel and Ozekici [1998]) extend these ideas and consider a model where component failures are stochastically dependent.

An interesting extension of system-based component testing is to the area of software reliability. Cheung [1980] developed a Markovian model for the transfer of control between various modules in a software system. This allows the system reliability to be expressed as a polynomial function of the component reliabilities. Poore et al. [1993] suggested following the standard practice of allocating the system reliability goal among its modules, and as discussed

earlier, this could be erroneous. Rajgopal et al. [2001] suggest a system-based component test procedure. Only bounds on Type 2 error probability are considered and an optimization problem is formulated where the objective is to minimize the total number of test instances across all modules subject to this constraint. The sum rule is used again and the authors present solutions to several problems in the literature. Finally, Jin and Coit [1999] address the use of component test plans to minimize the variance of the system reliability estimate for a series-parallel system with binomial failure data.

2.3 Review Of Literature That Followed The Research Presented In This Document

In this section we present the literature that followed the publication of research work presented in this document. For the reliability allocation problem, the main thrust of the research presented in this document is to consider discrete cost-reliability data sets. This is validated and such discrete data is adopted by several researchers including Jin et al [2003], Nimah [2015], Aneja et al [2004], Moreb [2007], Negi et al [2021], Yeh et al [2010], Pant et al [2017].

Gupta and Agarwal [2006] have used a genetic search algorithm in their redundancy optimization problem of multi-state series-parallel power system. They adopted a dynamic penalty function approach on some difficult to satisfy constraints based on the distance (some degree of violation of the constraints). This penalty approach is clearly inspired by the dynamic penalty function used in this research (Chapter 4).

Heydari et al [2014] have addressed optimal allocation of testing resources in reliability growth in the design of a system. They are clearly inspired by and used the simulated annealing strategy presented in this research (Chapter 4).

The literature that followed our research is divided into following categories: (i) Similar formulations but other solution approaches (ii) Different formulations with similar goals and, (iii) Different formulations with different goals.

2.3.1 Similar Formulations With Other Solution Approaches

Several researchers have formulated the problem similar to the combinatorial optimization problem formulations presented in this research. However, the solution procedures adopted vary. Aneja et al [2004] have adopted dynamic programming approach to address the same problem that is presented in this research. They extended their solutions to *k*-out-of-*N*:G and *k*-out-of-*N*:G-reducible systems with some success. NG and Sancho [2001] have developed a hybrid dynamic programming-depth first search algorithm with an application to redundancy allocation.

Yeh [2007] used an interactive augmented max-min MCS-RSM method to address a multi-objective network reliability problem. Here MCS stands for monte carlo simulation which is used to estimate the network reliability. RSM stands for the response surface methodology. RSM is used to derive an approximate function for the reliability of the network there by reducing the errors in the MCS using regression techniques. They addressed maximization of reliability along with minimization of cost.

There are many heuristic algorithms developed in recent years to address difficult to solve combinatorial optimization problems such as (1) Grey wolf optimization, (2) Particle Swarm optimization,

2.3.1.1 Grey Wolf Optimization

Grey wolf optimization simulates the chasing, hunting by grey wolves. In grey wolf optimization, some initial solution vectors are generated. The best solution is termed as alpha. Second best solution is termed as beta and the third best solution is termed as delta. The remaining solutions vectors are termed omega. The omega vectors in each iteration are updated based on their distance from alpha, beta, delta vectors. Thus in each iteration new vectors are generated by altering current omega vectors based on the distance from leader vectors alpha, beta and delta. After generating new vectors, all the vectors are evaluated to select new alpha, beta and delta vectors. Thus the selection process continues for a set number of iterations.

Kumar et al [2017] adopted Grey Wolf optimizer algorithm to solve a system reliability optimization problem. They report that this algorithm has given very satisfactory results.

2.3.1.2 Particle Swarm Optimization

Particle Swarm Optimization simulates food searching by a flock of birds. In Particle Swarm Optimization, each particle has its own location and velocity, which determine the flying direction and distance, respectively. In this approach, an initial set of solutions, called particles, are generated along with some randomly generated velocities for each. Each particle is associated with a position vector and a velocity vector. The position vector and velocity vector of each particle (solution) is then changed using some acceleration coefficients in each iteration. These acceleration coefficients are then updated in each iteration as well. By carefully altering accelerations and velocities in each iteration, search process is guided towards global optimum as the iterations progress.

Pant et al [2017] have adopted a Particle Swarm Approach to reliability optimization problem. They report that the results are very encouraging. Yeh et al [2010} have used a particle

swarm approach combined with Monte Carlo simulation to solve reliability optimization problem. Monte Carlo simulation helps in estimating the reliability of the system while PSO guides the solution towards optimum. They are able to apply this approach to more complex network reliability optimization problems. Negi et al [2021] have used a hybrid PSO and GWO-algorithm to complex system reliability optimization problem. They concluded that such hybrid meta heuristic performed better than previous heuristic solutions.

There are several other Nature-inspired algorithms that are being used by several researchers. Some of the other algorithms are (a) Ant colony optimization (b) Flower pollination algorithm (c) Cuckoo search algorithm. Interested readers may easily search and read literature on these topics. All these algorithms along with PSO and GWO are all population-based algorithms. In these algorithms, an initial population is generated and these populations are modified to generate new populations from iteration to iteration so that best solution is continuously improved through iterations. The evolutionary algorithm presented in this research is also one such population-based algorithm which performed satisfactorily. The most common criticism for such heuristic algorithms is that many parameters are required to guide the search through iterations. Setting of such parameters is problem specific and there are very few guidelines to set these parameters.

2.3.1.3 Neural Network Approach

Some researchers have adopted artificial neural networks to maximize reliability on some network problems. Yeh et al [2007] combine artificial neural network with Monte Carlo simulation to predict reliability of a network. They selected a multilayer feed forward network with non-polynomial activation functions as it approximated any continuous function in any degree of accuracy. They used back propagation to train this feed forward neural network. By using the neural network, they claimed good results.

Kaushik et al [2013] (in two articles of the same year) have demonstrated by using an adaptive gradient descent neural network approach with high learning rate and variable convergence rate. They used a back propagation training algorithm for their neural network. The input provided for the neural network is (1) the minimal cut sets of network problem for which reliability is desired, (2) link reliabilities for fixed and variable links of the network and, (3) the network reliability lower and upper bound obtained from minimal cut set added with neural network lower and upper bound. They have compared their results with simulation-based analytical methods and showed that their results are an improvement.

2.3.2 Different Formulations With Similar Goals

Moreb [2007] addressed repairable systems reliability allocation problem. They used optimum selection based on mean time to failure and mean time to repair. Wang and Li [2015] have addressed redundancy allocation for reliability design of engineering systems with failure interactions. They proposed an analytical model that describes the failure rates with failure interactions. This is followed by a modified analytical hierarchy process to solve redundancy allocation problem with failure interactions.

Kapur et al [2003] addresses the problem of optimal selection of components for a modular software system. Such a software is supposedly built by (i) assembling a set of off-the-shelf components that are available commercially and/or, (ii) In-built independent programs. It is assumed that more than one alternative is available for each module, they all come with associated cost. Redundant components are added to enhance fault tolerance. The objective is to maximize the reliability while subjected to budgetary constraints. In a separate model, they account for compatibility of alternatives available for different modules.

2.3.3 Different Formulations With Different Goals

Several researchers have developed different goals with variety of formulations while still addressing some kind of allocation problems. For example, Anzanello [2009] considered reliability evaluation of systems composed of non-identical multi-state components. When components have different magnitudes of failure probabilities, then behavior of reliability of system is difficult to understand. Anzanello address this problem for series and parallel systems that have non-identical three-state components. They attempted to allocate non-identical components with budgetary constraints. Their model comprised of nonlinear programming which is aimed at optimizing allocation of component under restrictions and solved for series system.

Goel et al [2004] have considered the problem of retrofit design of a multiproduct batch plant with a consideration of inherent reliability and maintainability of existing as well as new equipment. In their optimization problem, they sought optimal size, optimal operating mode and optimal allocation of inherent availability for new equipment during the retrofit stage. The production capacity is defined by the three decision parameters of (i) batch size, (ii) limiting cycle time and, (iii) overall plant availability. The availability is an indication of overall reliability of the plant here. Their formulation of the problem is a mixed integer nonlinear program. They also used discrete data sets in their formulations.

Keebom et al [2010] have developed decision support models for valuing improvements in component reliability and maintenance. Providing minimum life cycle cost and maximizing system availability is the main goal of their research. They considered (i) cost to repair (ii) cost to hold in inventory in the overall cost function. They used discrete data to formulate their problem.

Salmasnia et al [2017] have also used total cost of the system life cycle in their approach to address the system design. They have tried to design a multi-state degraded system while

optimizing maintenance, repair costs and mean system availability. Their estimation of maintenance costs of system is based on Poisson distribution for failures. They used categories such as minor maintenance and major maintenance. A minor maintenance restores system to previous better state where as a major maintenance restores system to the 'as good as new' state. They suggested an integrated optimization scheme and an aggregation method such that both objectives, (i) total cost (ii) mean availability of system, fall into the decision maker's acceptable region.

Dinesh Kumar et al [2007] have emphasized on considering total cost of ownership at the time of design itself as opposed to unit cost alone which is adopted by many system designers. According to them the total cost must include upstream unit cost as well as downstream operations, maintenance and support costs. They proposed a non-linear mathematical model that allocates system-level reliability and minimizes the total cost for a series-parallel system.

Kanagaraj and Jawahar [2009] have used a simulated annealing algorithm for optimal supplier selection using the reliability-based total cost of ownership model. They also emphasize the consideration of total cost of ownership which includes procurement, maintenance and downtime costs. They also included weight limitations as constraints in their optimization problem which minimizes the total cost while maintaining the required reliability level of the system. The formulations are nonlinear integer programs and their solution approach is simulated annealing algorithm which is clearly inspired by the nested simulated annealing algorithm given in this document (Chapter 4).

The above researchers are pointing in the right direction of considering the overall cost of the system during its life cycle. However, there is no consistency and firm theoretical approach in

estimating the costs associated. Each derive cost expressions based on different factors. There must be a consistent approach that should guide the estimation of overall cost.

3.0 Integer Programming Approach

Note: Portions of the work in this chapter were published in *Operations Research* (Majety, Dawande and Rajgopal [1999]).

3.1 Introduction And Notation

We formulate the reliability allocation problem as a 0-1 mixed integer programming problem. In the formulation, each binary variable either represents (a) a specific cost-reliability datum (when such data are purely discrete), or (b) a prespecified point on the cost-reliability curve of a component (when such a curve is available). We emphasize that in the latter instance, no assumption is made regarding the specific function describing the curve.

As will be seen in the following sections, except for simple series and simple parallel systems, the system reliability is a nonlinear function of these binary variables. Careful, and often painful, expansions of the expression for system reliability contain terms that are products of binary variables. While standard procedures could be adopted to linearize these terms, such product terms increase exponentially as the complexity of the system increases. Thus, this approach becomes impractical from a computational standpoint for anything other than small systems. In the following sections, we study the reliability expression for each system independently and identify linear relaxations of otherwise nonlinear constraints. Three common system configurations are analyzed: (1) Series-Parallel Systems, (2) Parallel-Series Systems, and

(3) *K-out-of-N* Systems. For each of these, the structure of the reliability expression is independently examined and an algorithm for exact solution is presented.

To generalize the formulations, we will consider any system as being comprised of subsystems that are assembled and configured with each other in a suitable fashion. Each subsystem in turn, is made up of components that are assembled and configured with each other in a suitable fashion. For each such component we have a discrete set of options from which we must make a selection. The following notation will be used:

i	Index for the subsystems comprising the system; $i \in \{1,2,\dots,L\}$
L	No. of subsystems
n_i	No. of components in subsystem i
j	Index for component in subsystem i; $j \in \{1,2,\dots,n_i\}$
K_{ij}	No. of discrete cost-reliability data elements for component j in subsystem i
p_{ijk}, c_{ijk}	Reliability and cost associated with option k of the cost-reliability data set for component j in subsystem i; $k \in \{1,2,\dots,K_{ij}\}$
x_{ijk}	Binary variable that is 1 if option k of cost-reliability data set for component j in subsystem i is selected, 0 otherwise
r_{ij}	Reliability of component j in subsystem i
f_i	Reliability of subsystem i
R	Reliability of the system
R_S	Minimum required system reliability

Note that p_{ijk} and c_{ijk} are given constants representing (respectively) the reliability and cost associated with a specific choice for a specific component in a specific subsystem, while

- r_{ij} is a function of $\{x_{ijk}\};\ \forall k \in \{1,2,\dots,K_{ij}\}$, i.e., of the specific data point selected
- f_i is a function of $\{r_{ij}\};\ \forall j \in \{1,2,\dots,n_i\}$, i.e., reliability of components in subsystem *i*
- *R* is a function of $\{f_i\};\ \forall i \in \{1,2,\dots,L\}$, i.e., reliability of the subsystems

Thus, the system reliability is ultimately a function of the binary variables $\{x_{ijk}\}$, while f_i and r_{ij} are intermediate quantities that are used as a matter of notational convenience. Note that the only true decision variables are the x_{ijk}.

3.2 Problem Formulations

A formulation of the reliability allocation problem **A**(***G***) for an arbitrary system ***G*** where the system cost is to be minimized subject to some minimum requirement on the system reliability, is as follows:

PROGRAM A(*G*):

$$Min \sum_{i=1}^{L}\sum_{j=1}^{n_i}\sum_{k=1}^{K_{ij}} c_{ijk}x_{ijk} \qquad \textbf{3.1}$$

$$s.t.\quad R_G = g(r_{ij}) \geq R_S \qquad \textbf{3.2}$$

$$x_{ijk} \in X$$

where

R_G is the reliability of the system

$$X = \left\{ x_{ijk} |\, x_{ijk} \in \{0,1\} \;\; \forall (i,j,k); \sum_{k=1}^{K_{ij}} x_{ijk} = 1 \;\; \forall\, (i,j) \right\} \quad \textbf{3.3}$$

$$r_{ij} = \sum_{k=1}^{K_{ij}} x_{ijk} p_{ijk} \;\; \forall (i,j); \quad \textbf{3.4}$$

and $g(.)$ is a function that depends on the specific system configuration.

Note that in (3.2) the system reliability R_G is expressed as a function of the reliability of its components r_{ij} via the function $g(r_{ij})$. The specific form of this function will depend on how the components are configured. The set *X* is defined in (3.3) as all 0-1 vectors that correspond to exactly one of the K_{ij} different options being selected for the j^{th} component of the i^{th} subsystem. The expression for r_{ij} in (3.4) simply assigns to it the value equal to the reliability of this selected component. Note that the definitions of the r_{ij} and the set *X* are identical for any configuration and are therefore not repeated when the formulations are stated; the reader is requested to refer back to the above definitions.

In this research we mainly concentrate on formulations for Series-Parallel and Parallel-Series system configurations. It is important to justify the selection of these. The following proposition will provide the link between any arbitrary system *G* and the Series-Parallel (or Parallel-Series) system derived from *minimal cutsets* (or *minimal pathsets*). A *minimal cutset* is defined as a minimal set of components such that when they fail together, the system fails.

Similarly, a *minimal pathset* is defined as a minimal set of components such that when they work together, the system works. A system may have multiple *minimal cutsets* as well as multiple *minimal pathsets*. Figure 3 gives an example of a particular system along with its minimal *pathsets* and minimal *cutsets*.

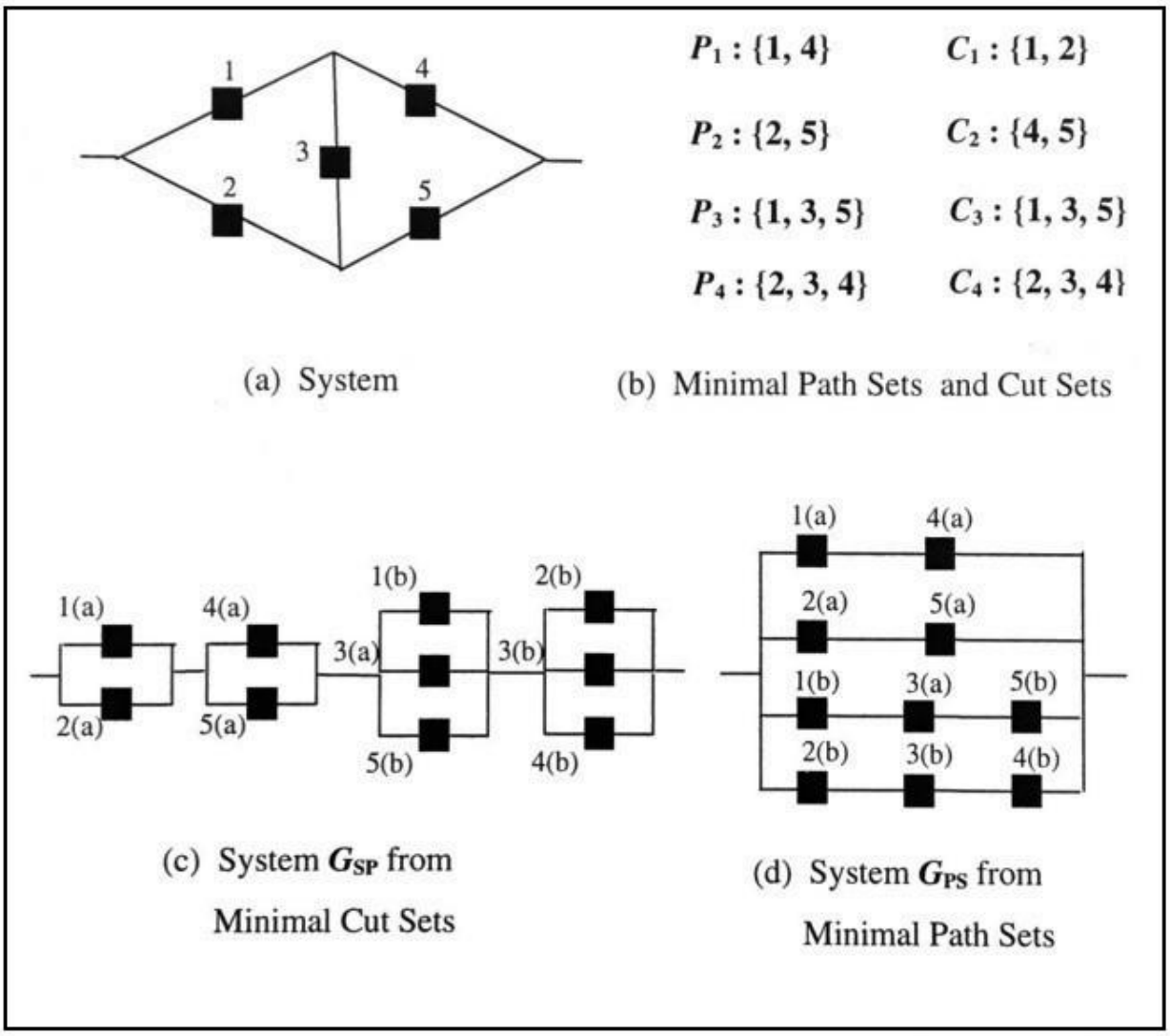

Figure 3: An Example of a system and the derived *SP*, *PS* systems

Consider a general system *G* in which all components are statistically independent. The collection of all minimal *cutsets* for system *G* is given by $\mathcal{C} = \{C_1, \dots, C_{|\mathcal{C}|}\}$ and the collection of

all minimal *pathsets* for system G is given by $\mathcal{P} = \{P_1, \dots, P_{|\mathcal{P}|}\}$. Then the system G can be represented as a series connection of subsystems $C_1, \dots, C_{|\mathcal{C}|}$ and it can also be represented as a parallel connection of subsystems $P_1, \dots, P_{|\mathcal{P}|}$. Consider a component of G that belongs to $C_{q_1}, \dots, C_{q_s}$, where $\{q_1, \dots, q_s\} \subseteq \{1, \dots, |\mathcal{C}|\}$. Since the same component appears in multiple cutsets, all these cutsets are statistically dependent on each other. Now assume that the (same) components appearing in $C_{q_1}, \dots, C_{q_s}$ are identical but statistically independent components. Assuming the same for every component that is repeated in multiple cutsets, the resulting *cutsets* are statistically independent. Let these *cutsets* be $C'_{r_1}, \dots, C'_{|\mathcal{C}|}$. We denote the system G_{SP} as series connection of $C'_{r_1}, \dots, C'_{|\mathcal{C}|}$. For example, consider Figure 3. The system in Figure 3 has the *cutsets* C_1 = {1,2}, C_2= {4,5}, C_3 = {1,3,5} and, C_4 = {2,3,4}. Here component 1 is common to both C_1 and C_3. Suppose we label the component 1 in C_1 as 1(a) and it is statistically independent of the component in C_3 which is denoted as 1(b). This is the same as duplicating component 1 for C_3. If we do the same for all components, then the resulting cutsets form a System G_{SP} as shown in Figure 3.

Similarly, now consider a component of G that belongs to $P_{q_1}, \dots, P_{q_s}$ where $\{q_1, \dots, q_s\} \subseteq \{1, \dots, |\mathcal{P}|$. Since the same component appears in multiple pathsets, all these pathsets are statistically dependent on each other. Assume once again that the (same) components appearing in $P_{q_1}, \dots, P_{q_s}$ are identical but statistically independent components. Assuming the same for every component that is repeated in multiple pathsets, the resulting *pathsets* are statistically independent. Let these pathsets be $P'_{r_1}, \dots, P'_{|\mathcal{P}|}$. We denote the system G_{PS} as a parallel connection of $P'_{r_1}, \dots, P'_{|\mathcal{P}|}$. For example, consider Figure 3. The system in Figure 3 has the *pathsets* P_1 = {1,4}, P_2 = {2,5}, P_3= {1,3,5} and, P_4 = {2,3,4}. The component 1 is common to both P_1 and P_3. If we assume that component 1 in P_1 as 1(a) and that it is statistically independent of component 1 in P_3 which is

denoted as 1(b), then this is the same as duplicating component 1 for P_3. If we do the same for all components, then resulting *pathsets* form a System G_{PS} as shown in Figure 3.

Proposition 1: Suppose $\boldsymbol{Z_G^*, Z_{SP}^*, Z_{PS}^*}$ are optimal solutions for Program A(G), Program A(G_{SP}) and Program A(G_{PS}) respectively; refer to Section 3.2 for the description of Program A(.). Then $\boldsymbol{Z_{SP}^* \leq Z_G^* \leq Z_{PS}^*}$.

Proof: Programs A(G), A(G_{SP}) and A(G_{PS}) differ only in the reliability constraint which may be written for each (respectively) as

$$R_G \geq R_S \qquad \text{for A}(G) \tag{i}$$

$$R_{SP} \geq R_S \quad \text{for A}(G_{SP}) \tag{ii}$$

$$R_{PS} \geq R_S \quad \text{for A}(G_{PS}) \tag{iii}$$

It is a well-known fact that $R_{PS} \geq R_G \geq R_{SP}$ (Leemis [1995]). Thus, it is clear that (ii) $\Rightarrow$(i) and (i) $\Rightarrow$(iii). Let X_{PS}, X_{SP} and X_G represent the set of all feasible solutions of A(G), A(G_{SP}) and A(G_{PS}) respectively. Then it is clear that $X_{SP} \subseteq X_G \subseteq X_{PS}$. Hence the proof.

The utility of this proposition is that it gives us an outline for a solution approach to any arbitrary system G. If we solve for G_{SP}, it will provide a feasible solution with an upper bound for the optimal cost for G. The formulation for G is not guaranteed to be linear but the formulation for G_{PS} serves as a relaxation (although nonlinear) for G. Any linear relaxation for G_{PS} will be an automatic linear relaxation for G. Hence it can be seen that the forthcoming Algorithm 1 (in Section 3.4) will also be useful to solve for arbitrary systems; however, more iterations will be needed than what would be required to solve for the optimum solution to G_{PS}.

In the foregoing analysis, one should note that for an arbitrary system, the number of *minimal cutsets* (or *minimal pathsets*) can be exponentially large and at times very difficult to obtain; clearly, this can be a major issue. However, the solution procedures given in this research will be useful for developing good solutions for any arbitrary systems. In the following sections, formulations for Series, Parallel, Series-Parallel, Parallel-Series, and *K-out-of-N* systems are developed and discussed. In all the models that follow X and r_{ij} are given by (3) & (4) respectively, and these expressions are therefore not repeated.

3.2.1 Simple Series Systems

For a simple Series system each component constitutes a "subsystem", so that $n_i = 1$ and $j \in \{1\}$; $\forall i$. The formulation is then as follows:

$$Min \sum_{i=1}^{L} \sum_{k=1}^{K_{i1}} c_{i1k} x_{i1k} \tag{3.5}$$

$$s.t. \quad R_G = \prod_{i=1}^{L} r_{i1} \geq R_S \tag{3.6}$$

$$x_{i1k} \in X$$

Note that the nonlinear system reliability constraint is readily linearized by taking logarithms and using the equivalence $r_{i1} \in \{p_{i1k}\} \equiv \ln(r_{i1}) \in \{\ln(p_{i1k})\}$. Then $\sum_{i=1}^{L} \sum_{k=1}^{K_{i1}} x_{i1k} \ln(p_{i1k}) \geq \ln(R_S)$ replaces constraint (3.6), and the problem reduces to a 0-1 integer linear programming problem in the variables $\{x_{i1k}\}$.

3.2.2 Simple Parallel Systems

For a simple Parallel system, we have a single "subsystem", so that *L*= 1 and i = 1 and we use n_1 to represent the number of components in parallel. The formulation is then as follows:

$$Min\ \sum_{j=1}^{n_1}\sum_{k=1}^{K_{1j}} c_{1jk}x_{1jk} \tag{3.7}$$

$$s.t.\quad R_G = 1-\prod_{j=1}^{n_1}(1-r_{1j}) \geq R_S \tag{3.8}$$

$$x_{1jk} \in X$$

Similar to the series system, the nonlinear system reliability constraint is easily linearized by taking logarithms and using the equivalence $r_{1j} \in \{p_{1jk}\} \equiv \ln(1-r_{1j}) \in \{\ln(1-p_{1jk})\}$. The constraint $R_G = 1-\prod_{j=1}^{n_1}(1-r_{1j}) \geq R_S$ i.e.; $\prod_{j=1}^{n_1}(1-r_{1j}) \leq 1-R_S$ can thus be replaced by $\sum_{j=1}^{n_1}\sum_{k=1}^{K_{1j}} x_{1jk}\ln(1-p_{1jk}) \leq \ln(1-R_S)$. Thus, for a simple parallel system the problem is once again a 0-1 integer linear programming problem in $\{x_{1jk}\}$. The next three subsections introduce more complex systems.

3.2.3 Series-Parallel (SP) Systems

For a Series-Parallel system, the formulation is as follows:

$$Min\ \sum_{i=1}^{L}\sum_{j=1}^{n_i}\sum_{k=1}^{K_{ij}} c_{ijk}x_{ijk} \tag{3.9}$$

[*SP*]: $s.t. \quad \prod_{i=1}^{L} f_i \geq R_S$

3.10

$$f_i = 1 - \prod_{j=1}^{n_i} (1 - r_{ij}) \; ; \; \forall i$$

3.11

$$x_{ijk} \in X$$

Note that f_i is the reliability of the i^{th} parallel subsystem. Unlike with simple series or simple parallel systems, there is no ready transformation whereby this program can be replaced by a linear equivalent.

3.2.4 Parallel -Series (PS) Systems

For a Parallel-Series system, the formulation is as follows:

$$Min \; \sum_{i=1}^{L} \sum_{j=1}^{n_i} \sum_{k=1}^{K_{ij}} c_{ijk} x_{ijk}$$

3.12

[*PS*]: $s.t. \quad 1 - \prod_{i=1}^{L} (1 - f_i) \geq R_S$

3.13

$$f_i = \prod_{j=1}^{n_i} r_{ij} \; ; \; \forall i$$

3.14

$$x_{ijk} \in X$$

Note that f_i is the reliability of the i^{th} series subsystem. Once again, unlike with simple series or simple parallel systems, there is no ready transformation whereby this program can be replaced by a linear equivalent.

3.2.5 *K-out-of-N* Systems

In a *K-out-of-N* system, the system works if at least *K* components out of a total of *N* components work. See Figure 1 in Chapter I for an example of a 2-*out-of*-3 system. Note that unlike in the case of an SP or a PS system, for a *K-out-of-N* system we do not have distinct subsystems. Hence, for convenience of notation, all components are assumed to belong to the same subsystem ($i \in \{1\}$ for all components). Now define *e* as some arbitrary set of *K* components out of the *N* components and define E_e as the event that all of the components in the set *e* work. Then $\mathrm{P}\{E_e\} = \prod_{j \in e} r_{1j}$. Since the system works as long as the event E_e occurs and there are a total of $\binom{N}{K}$ different possible events of this kind, the system reliability can be expressed as $R = \sum_e \Pr\{E_e\} - \Delta$, where Δ is a positive quantity involving joint probabilities and is a function of $\{r_{ij}\}$. The formulation of the allocation problem for this system is as follows:

$$Min \ \sum_{j=1}^{N} \sum_{k=1}^{K_{1j}} c_{1jk} x_{1jk} \qquad \textbf{3.15}$$

$$[\boldsymbol{T}]: \qquad s.t. \quad \sum_{\forall e} \left(\prod_{j \in e} r_{1j}\right) - \Delta \geq R_S \qquad \textbf{3.16}$$

$$x_{1jk} \in X$$

As with the PS and SP systems, there is no ready way to linearize (3.16).

Much of the discussion that follows is common to all three problem formulations (***SP, PS, T***) and we use the notation ***SP/PS/T*** to refer to the problem formulations described above. Once again, it is important to note that linearization of the system reliability constraint in ***SP/PS/T*** is not

possible as in the case of simple Series and simple Parallel systems. As such, the above problems are 0-1 integer nonlinear programming problems which are hard to solve directly; this calls for other approaches.

3.3 Linear Relaxations To NLIP Formulations

The primary difficulty in solving ***SP/PS/T*** stems from the fact that the reliability constraint is nonlinear. In order to solve ***SP/PS/T***, we introduce three new integer linear programs ***SP_0/PS_0/T_0***. Each of these will be shown to be a relaxation of the original problem.

3.3.1 Problem SP_0

$$Min \ \sum_{i=1}^{L} \sum_{j=1}^{n_i} \sum_{k=1}^{K_{ij}} c_{ijk} x_{ijk} \tag{3.17}$$

$$\textbf{[}SP_0\textbf{]}: \quad s.t. \quad \prod_{i=1}^{L} (1 - f_i) \leq (1 - R_S^{\frac{1}{L}})^L \tag{3.18}$$

$$f_i \geq R_S \ ; \ \forall i \tag{3.19}$$

$$f_i = 1 - \prod_{j=1}^{n_i}(1 - r_{ij}) \; ; \; \forall i$$

3.20

$$x_{ijk} \in X$$

Proposition 2: ***SP₀*** is a 0-1 linear program.

Proof: First from (3.20),

$$\ln(1 - f_i) = \sum_{j=1}^{n_i} \ln(1 - r_{ij}) = \sum_{j=1}^{n_i} \sum_{k=1}^{K_{ij}} (x_{ijk} \ln(1 - p_{ijk})) \; ; \; \forall i$$

3.21

where the last inequality follows from the equivalence $r_{ij} \in \{p_{ijk}\} \equiv \ln(1 - r_{ij}) \in \{\ln(p_{ijk})\}$.

Hence constraint (3.19), $f_i \geq R_S \Rightarrow (1 - f_i) \leq (1 - R_S)$

$$\Rightarrow \ln(1 - f_i) \leq \ln(1 - R_S)$$

$$\Rightarrow \sum_{j=1}^{n_i} \sum_{k=1}^{K_{ij}} (x_{ijk} \ln(1 - p_{ijk})) \leq \ln(1 - R_S); \; \forall i,$$

where the last clause follows from (3.21). These are all linear constraints in $\{x_{ijk}\}$.

Looking at constraint (3.18),

$$\prod_{i=1}^{L}(1 - f_i) \leq (1 - R_S^{\frac{1}{L}})^L \Rightarrow \sum_{i=1}^{L} \ln(1 - f_i) \leq L \ln(1 - R_S^{\frac{1}{L}})$$

$$\Rightarrow \sum_{i=1}^{L} \sum_{j=1}^{n_i} \sum_{k=1}^{K_{ij}} (x_{ijk} \ln(1 - p_{ijk})) \leq L \ln(1 - R_S^{\frac{1}{L}}),$$

which follows from (3.21). Once again, this is linear in $\{x_{ijk}\}$. Thus, all constraints are linear in $\{x_{ijk}\}$.

Proposition 3: Every feasible solution for ***SP*** is feasible in ***SP₀***.

Proof: Consider a feasible solution to ***SP***. First note that ***SP*** and ***SP₀*** differ only in the system reliability constraints with (3.18) & (3.19) replacing (10), namely $\prod_{i=1}^{L} f_i \geq R_S$. This clearly implies that $f_i \geq R_S;\ \forall i$ since $0 \leq f_i \leq 1\ ;\ \forall i$ which is (19).

Now the constraint $\prod_{i=1}^{L} f_i \geq R_S \Rightarrow\ (\prod_{i=1}^{L} f_i)^{1/L} \geq {R_S}^{1/L}$

(a)

Since the Arithmetic Mean (AM) of a set of nonnegative numbers is at least as large as their Geometric Mean (GM), it follows that $\frac{\sum_{i=1}^{L} f_i}{L} \geq (\prod_{i=1}^{L} f_i)^{1/L}$. Therefore, from (a)

$$\sum_{i=1}^{L} f_i \geq L\, R_S^{\frac{1}{L}} \Rightarrow \sum_{i=1}^{L} (1 - f_i) \leq L\,(1 - R_S^{\frac{1}{L}})$$

(b)

Once again by the AM-GM inequality $L(\prod_{i=1}^{L}(1 - f_i))^{\frac{1}{L}} \leq \sum_{i=1}^{L}(1 - f_i)$

(c)

From (b) and (c) it follows that $\prod_{i=1}^{L}(1 - f_i) \leq (1 - R_S^{\frac{1}{L}})^L$; which is the constraint (3.19) in ***SP₀***. Hence the result.

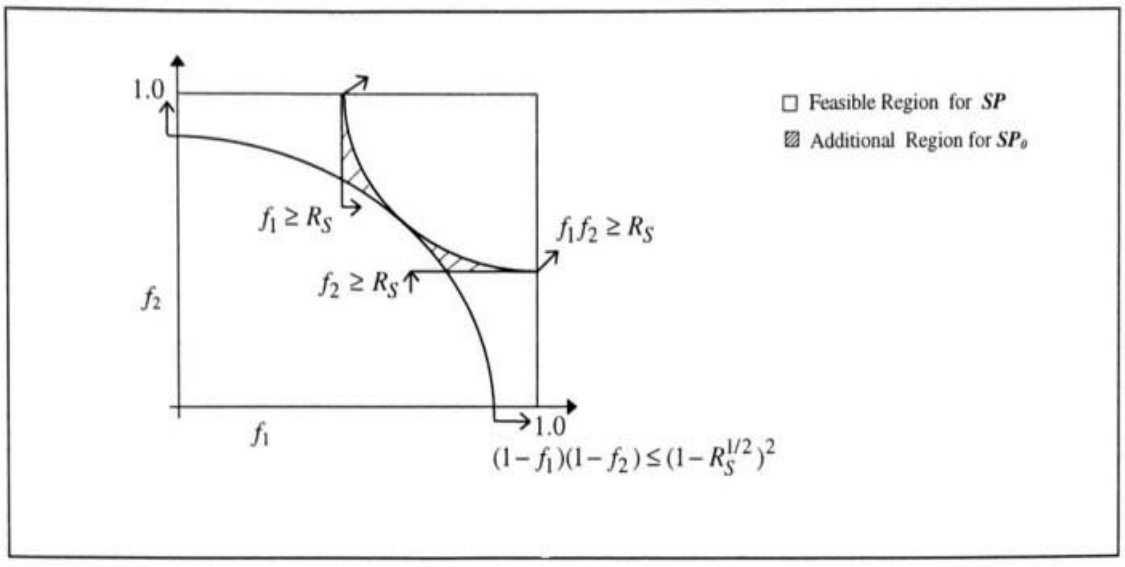

Figure 4: Feasible Regions for *SP* and *SP₀*

The feasible region for ***SP*** is thus contained entirely within that of ***SP₀***. Figure 4 depicts the same for an example of two subsystems. In summary, Propositions 2 and 3 thus show that ***SP₀*** is a linear relaxation of ***SP***.

3.3.2 Problem *PS₀*

$$Min \ \sum_{i=1}^{L} \sum_{j=1}^{n_i} \sum_{k=1}^{K_{ij}} c_{ijk} x_{ijk}$$

3.22

[*PS₀*]: $$s.t. \ \ f_i \geq e^{-Mz_i}\left(1 - \left(1 - R_S\right)^{\frac{1}{L}}\right); \ \forall i$$

3.23

$$\sum_{i=1}^{L} z_i \leq L - 1$$

3.24

$$f_i = \prod_{j=1}^{n_i} r_{ij} \; ; \; \forall i$$

3.25

$$x_{ijk} \in X, z_i \in \{0,1\}; \; \forall i$$

Here *M* is a sufficiently large positive constant. Notice that ***PS₀*** is also a 0-1 integer linear program in the variables $\{x_{ijk}\}$ and $\{z_i\}$; this follows from (25), (23) and the equivalence $r_{ij} \in \{p_{ijk}\} \equiv \ln(r_{ij}) \in \{\ln(p_{ijk})\}$, which yields $\ln(f_i) = \sum_{j=1}^{n_i} \ln(r_{ij}) = \sum_{j=1}^{n_i} \sum_{k=1}^{K_{ij}} x_{ijk} \ln(p_{ijk}) \geq -Mz_i + \ln(1-(1-R_S)^{\frac{1}{L}})$, which is linear in variables $\{x_{ijk}\}$ and $\{z_i\}$.

Proposition 4: Every feasible solution to ***PS*** is feasible in ***PS₀***.

Proof: Once again, ***PS*** and ***PS₀*** differ only in the system reliability constraint, with (3.23) and (3.24) replacing (3.13). The new constraints in ***PS₀*** in place of the reliability constraint of ***PS*** are:

$$f_i \geq e^{-Mz_i}(1-\left(1-R_S\right)^{\frac{1}{L}}); \; \forall i \quad \text{and} \sum_{i=1}^{L} z_i \leq L-1$$

Consider a solution to ***PS*** where (3.13) holds. Note that from (3.24), z_r = 0 for some *r*, so that the first constraint indicates that $f_r \geq (1-\left(1-R_S\right)^{\frac{1}{L}}); \; \forall r$. For the case where z_r = 1 for some *r*, a sufficiently large value of *M* reduces the first constraint to $f_r \geq 0$ which is always satisfied. Hence it is sufficient to prove that the original constraint (3.13) in ***PS***, namely $1-\prod_{i=1}^{L}(1-f_i) \geq R_S$ implies that $f_i \geq (1-\left(1-R_S\right)^{\frac{1}{L}}$ for at least one $i \in \{1,2,\ldots,L\}$.

To prove this, suppose that this constraint in ***PS*** holds and $f_i < (1-(1-R_S)^{\frac{1}{L}}); \; \forall i$.

Then $(1-f_i) > (1-R_S)^{\frac{1}{L}};\ \forall i \Rightarrow \prod_{i=1}^{L}(1-f_i) > (1-R_S) \Leftrightarrow 1-\prod_{i=1}^{L}(1-f_i) < R_S,$ which contradicts (3.13). Hence the proof.

Once again, from Proposition 4, it may be seen that ***PS₀*** is a linear relaxation of ***PS*** and contains the feasible region of the latter entirely within its own. The same is shown in Figure 5 for an example with two subsystems.

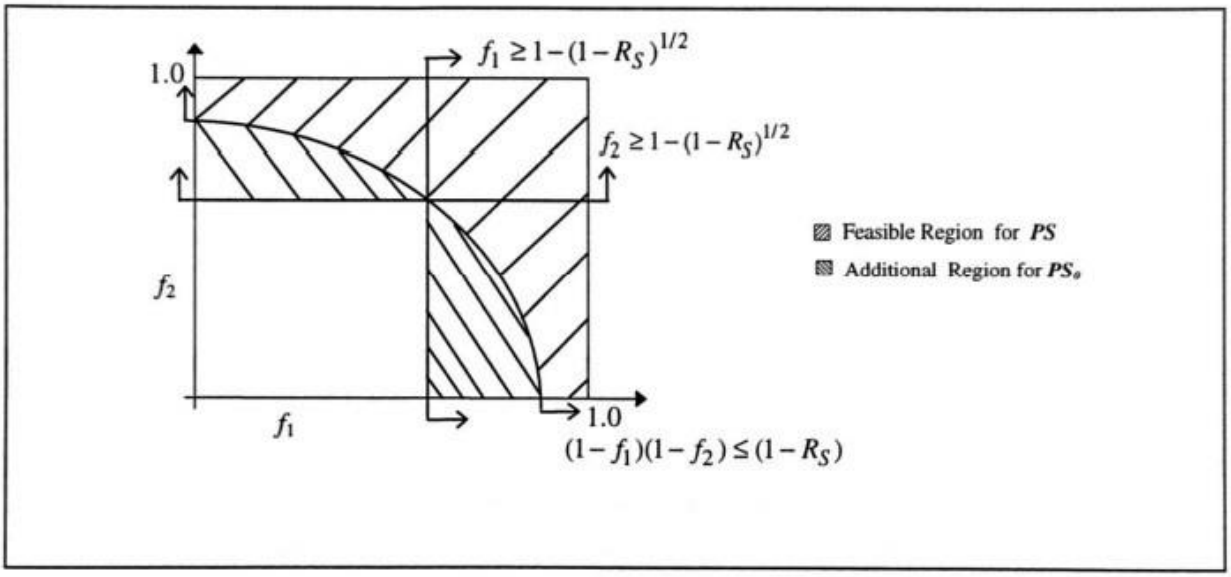

Figure 5: Feasible regions for *PS* and *PS₀*

3.3.3 Problem T_0

$$Min \sum_{j=1}^{N}\sum_{k=1}^{K_{1j}} c_{1jk}x_{1jk}$$

3.26

$$[T_0]: \qquad s.t. \ \sum_{j=1}^{N} r_{1j} \geq (R_S(K!))^{1/K}$$

3.27

$$x_{1jk} \in X$$

Again, note that ***T₀*** is an integer linear program because

$$\sum_{j=1}^{N} r_{1j} \geq \left(R_S(K!)\right)^{\frac{1}{K}} \Rightarrow \sum_{j=1}^{N} \sum_{k=1}^{K_{1j}} x_{1jk} p_{1jk} \geq \left(R_S(K!)\right)^{\frac{1}{K}}$$

which is clearly linear in $\{x_{1jk}\}$

Proposition 5: Every feasible solution to ***T*** is feasible to ***T₀***.

Proof: ***T*** and ***T₀*** differ only in the system reliability constraint with (3.27) replacing (3.16). First consider (3.16): $\sum_{\forall e}(\prod_{j\in e} r_{1j}) - \Delta \geq R_S \Rightarrow \sum_{\forall e}(\prod_{j\in e} r_{1j}) \geq R_S$. Now note that the coefficient of $\prod_{j\in e} r_{1j}$ in the expansion of $(\sum_{j=1}^{N} r_{1j})^K$ is exactly *K*! and that each such product term appears in this expansion. Furthermore, the expansion of $(\sum_{j=1}^{N} r_{1j})^K$ contains additional positive terms, e.g., $r_{11}^K, \ldots, r_{1N}^K$. Thus, it follows that

$$(\sum_{j=1}^{N} r_{1j})^K \geq \sum_{\forall e}\left(K!\left(\prod_{j\in e} r_{1j}\right)\right) \Rightarrow (\sum_{j=1}^{N} r_{1j})^K \geq (K!)R_S.$$

The last expression clearly implies (3.27) and thus ***T₀*** is once again a linear relaxation of ***T***. Hence the result.

3.4 An Algorithm To Solve For Optimal Solutions

Before outlining a cutting plane algorithm that can be used to solve ***SP/PS/T*** via the relaxations ***SP₀/PS₀/T₀*** we state the following proposition that holds for all three of these formulations and forms the basis for the cutting planes introduced by the algorithm.

Proposition 6: Let $\boldsymbol{x}^*$ be an optimal solution vector to ***SP₀/PS₀/T₀***. Define $I^* = \left\{\{i,j,k\}\middle| x^*_{ijk} = 1\right\}$, with cardinality $|I^*|=n_1+n_2+\ldots n_L$. If $\boldsymbol{x}^*$ is not feasible for ***SP/PS/T*** then the inequality $\sum_{\forall\{i,j,k\}\in I^*} x_{ijk} \leq |I^*| - 1$ eliminates $\boldsymbol{x}^*$ from the feasible set of ***SP₀/PS₀/T₀*** and is valid for ***SP/PS/T***.

Proof: Clearly, the vector $\boldsymbol{x}^*$ does not satisfy the proposed inequality and is therefore eliminated. Moreover $\boldsymbol{x}^*$ is the only point that is eliminated from the feasible set of ***SP₀/PS₀/T₀*** since it is the only 0-1 point that does not satisfy the inequality. At every other point at least one of the x_{ijk} = 1 when x^*_{ijk} = 0. Since the feasible region for ***SP/PS/T*** is contained entirely within that of ***SP₀/PS₀/T₀***, the inequality thus does not eliminate any feasible points for ***SP/PS/T*** and hence it is a valid inequality for the latter.

We now present an iterative algorithm for solving ***SP/PS/T*** to optimality.

Algorithm 1

1. Set p = 0 and solve ***SP₀/PS₀/T₀***.
2. If an integer solution vector $|\boldsymbol{x}^{p*}|$ to ***SP₀/PS₀/T₀*** is feasible to ***SP/PS/T***, stop; otherwise go to step 3.
3. Define $I^p = \{\{i,j,k\}|x^{p*}_{ijk} = 1\}$.
4. Add the inequality $\sum_{\forall\{i,j,k\}\in I^p} x_{ijk} \leq |I^p| - 1$ to ***SP₀/PS₀/T₀***. Note that the vector $\boldsymbol{x}^{p*}$ is now infeasible in ***SP₀/PS₀/T₀***.
5. Set $p = p + 1$. Solve ***SP₀/PS₀/T₀*** and return to Step 2.

Proposition 7: Algorithm 1 is finite and finds the optimal solution to ***SP/PS/T***.

Proof: By proposition 3/4/5, the feasible set of ***SP/PS/T*** is contained in the feasible set of ***SP₀/PS₀/T₀***. Hence, if the optimal solution to ***SP₀/PS₀/T₀*** at the p^{th} iteration ($\boldsymbol{x}^{p*}$) is feasible for

SP/PS/T, then it is also optimal for ***SP/PS/T***. Otherwise, the constraint added in Step 4 eliminates exactly one point, namely $\boldsymbol{x}^{p*}$ from the feasible set of $\boldsymbol{SP_0/PS_0/T_0}$ and nothing from that of ***SP/PS/T***. Since there are only a finite number of points which are feasible for $\boldsymbol{SP_0/PS_0/T_0}$ but infeasible for ***SP/PS/T*** and each iteration removes one such point, the algorithm is finite.

3.5 Examples And Observations

Algorithm 1 is illustrated via data sets in Table 1 and Table 2 for the three systems addressed in the earlier sections. Four reliability levels (0.85, 0.90, 0.95 and 0.99) are selected and four components considered. Costs for each component were randomly generated corresponding to these four reliability levels, while ensuring that these costs increased with reliability. The data are shown in Table 1 and Table 2. Note that an additional cost and reliability datum is added for each component (k=1), with c_{ij1} assumed to be zero and p_{ij1} assumed to be equal to a sufficiently small value ε. This is for the following reason: if x_{ij1} = 1 at the optimum for some i, j then this component j of subsystem i can be interpreted as being irrelevant and can therefore be removed from the system. The integer linear programs treated by the algorithm were solved using CPLEX-4.0 callable library.

Table 1: Reliability data matrix $\{p_{ijk}\}$

k	1	2	3	4	5
p_{ijk}	0.0	0.85	0.90	0.95	0.99

Table 2: Cost data matrix $\{c_{ijk}\}$ for example problems

Component No	*i*	*j*	*K*				
			1	2	3	4	5
1	1	1	0.00	251.05	339.80	440.45	597.70
2	1	2	0.00	354.00	449.50	572.75	703.30
3	2	1	0.00	248.55	347.90	463.75	609.40
4	2	2	0.00	276.70	370.20	495.15	628.50

Example 1: Series-Parallel system

We choose a Series-Parallel system with two subsystems. Components 1 & 2 are in parallel for the first subsystem, and Components 3 & 4 are in parallel for the second subsystem. We choose R_S = 0.97. Initially $\boldsymbol{SP_0}$ is formulated and solved. The optimal solution ($x_{115} = x_{121} = x_{215} = x_{222} = 1$) yields a value of 0.9677 for the system reliability, which violates the reliability constraint of $\boldsymbol{SP}$. Hence, the inequality $x_{115} + x_{121} + x_{215} + x_{222} \leq 3$ is added as per Proposition 5 and $\boldsymbol{SP_0}$ is reoptimized. The new optimal solution ($x_{115} = x_{121} = x_{215} = x_{221} = 1$) yields a system reliability of 0.9801, which satisfies the reliability constraint of $\boldsymbol{SP}$ and provides the optimum to $\boldsymbol{SP}$. The optimal cost is 1207.10

Example 2: Parallel-Series system

We choose a Parallel-Series system with two subsystems. Components 1 & 2 are in series for the first subsystem, and Components 3 & 4 are in series for the second subsystem. We choose R_S = 0.97. $\boldsymbol{PS_0}$ is formulated and solved. The resulting optimal solution ($x_{111} = x_{121} = x_{214} = x_{223} = 1$) yields a system reliability of 0.855, which violates the reliability constraint of $\boldsymbol{PS}$. Hence, the inequality $x_{111} + x_{121} + x_{214} + x_{223} \leq 3$ is added as per Proposition 5 and $\boldsymbol{PS_0}$ is reoptimized. After nine such iterations, we get an optimal solution ($x_{111} = x_{121} = x_{215} = x_{225} = 1$) with a

system reliability of 0.9801 which satisfies the reliability constraint of ***PS*** at an optimal cost of 1237.9

Example 3: *K-out-of-N* system

We consider a 2-*out-of*-3 system with components 1,3,4 from Table 2. For this system, we choose $R_S = 0.95$. Initially $\boldsymbol{T_0}$ is formulated and solved. The optimal solution ($x_{112} = x_{132} = x_{141} = 1$) yields a system reliability of 0.7225, which violates the system reliability constraint of ***T***. Hence the inequality $x_{112} + x_{132} + x_{141} \leq 2$ is added and $\boldsymbol{T_0}$ is re-solved. After seven such iterations, we get the optimal solution ($x_{113} = x_{132} = x_{142} = 1$) that gives a system reliability of 0.952 and satisfies the reliability constraint of ***T***. The optimal cost is 865.05

3.6 Strength of $\boldsymbol{SP_0}$ and $\boldsymbol{PS_0}$

Algorithm 1 presented earlier (Section 3.4) is guaranteed to solve the reliability allocation problems addressed. However, when the number of components is large, this algorithm may require a very large number of iterations. Let us use the term "gap" to denote the portion of the feasible space of the linear relaxation that does not intersect with the feasible space of the original nonlinear formulation with the integrality constraints ignored. At each iteration, the algorithm eliminates one infeasible integer solution from this gap. If the problem size is large, one could expect a large number of such infeasible integer solutions in the gap. Also, for the problems of ***SP*** and ***PS***, the size of this gap also depends on the value of R_S. For the case of ***SP*** consider two problem instances for the same system with identical cost-reliability data but with R_S^1 and R_S^2 as reliability target values where $R_S^1 > R_S^2 > 0.5$. Then the gap (see Figure 6) for the problem instance with R_S^1 as the specification will be smaller compared to that for the problem instance specified

with R_S^2. On the other hand, for ***PS***, the size of gap for the instance with R_S^1 will be larger compared to that of the problem instance with R_S^2 (see Figure 6).

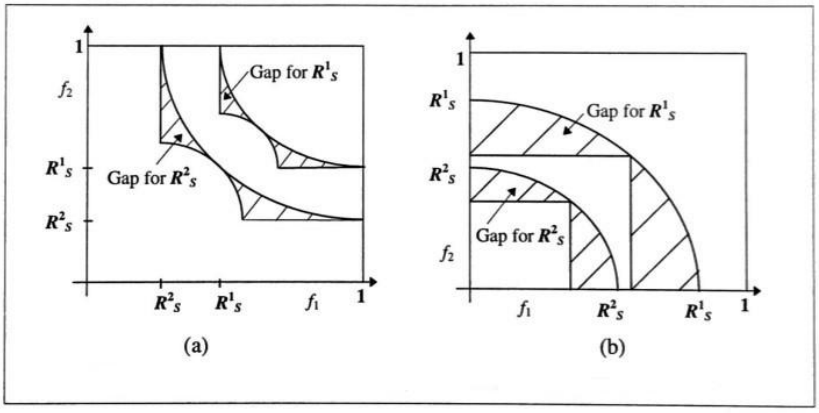

Figure 6: The gap between (a) *SP*$_0$ and *SP* (b) *PS*$_0$ & *PS*

Also note that the feasible solutions are discrete, and the number of such solutions in the gap (see Figure 6) also depends upon the discrete datasets. A larger gap does not necessarily mean more infeasible integer solutions in that gap, and a larger gap does not necessarily mean more iterations in the algorithm. A general comment is that the computational time of the algorithm depends on the number of infeasible solutions (in the order of increasing reliability) in the gap, which in turn depends upon two factors: (i) the number of components and number of discrete data points, and (ii) the target value of R_S.

Consider a twenty-component example (see Table 3 and Table 4 below) for ***SP*** and ***PS*** systems. First, we choose a Series-Parallel system with four subsystems. Each subsystem has five components in parallel. For R_S = 0.99, the optimal solution to the problem was obtained after 33 iterations of the algorithm described earlier (see Table 5 below and recall that x_{ijk} = 1 implies that the k^{th} choice is made for the j^{th} component in subsystem *i*). This configuration resulted in an overall system reliability of 0.990525 at a cost of 1139.05 units. However, when we tried to solve the same problem with R_S = 0.98; we could not do so. This point emphasizes the sensitivity of the

proposed algorithm to the problem parameters and thus the need to develop a solution procedure which is more robust. This issue can be expected to be even more pronounced as the size of our problem and the size of the cost-reliability data set (in terms of the number of options available) increases.

Table 3: Reliability Data Matrix $\{p_{ijk}\}$

k	1	2	3	4	5
p_{ijk}	0.001	0.85	0.90	0.95	0.99

Table 4: Cost data matrix $\{c_{ijk}\}$ for example problems

Component No.	i	j	K				
			1	2	3	4	5
1	1	1	0.00	64.35	280.65	427.35	511.15
2	1	2	0.00	120.75	160.25	419.45	650.25
3	1	3	0.00	88.85	291.75	479.15	731.35
4	1	4	0.00	117.45	238.85	388.75	710.55
5	1	5	0.00	121.55	277.35	379.85	703.85
6	2	1	0.00	104.25	172.15	446.55	515.35
7	2	2	0.00	98.35	195.35	304.35	651.45
8	2	3	0.00	148.15	203.65	498.95	671.15
9	2	4	0.00	115.95	173.95	494.65	734.55
10	2	5	0.00	148.65	159.25	321.15	682.95
11	3	1	0.00	94.45	207.45	430.25	525.35
12	3	2	0.00	138.25	230.75	482.85	623.75
13	3	3	0.00	101.75	189.15	474.15	562.45
14	3	4	0.00	75.15	154.35	451.25	680.35
15	3	5	0.00	50.45	271.85	329.35	568.85
16	4	1	0.00	79.65	190.35	430.75	691.25
17	4	2	0.00	52.15	215.95	394.15	657.55
18	4	3	0.00	129.85	268.65	452.95	622.85

19	4	4	0.00	123.95	279.25	400.85	745.95
20	4	5	0.00	53.45	239.45	464.55	718.55

Table 5: Optimum configuration for Series-Parallel system for R_S = 0.99

Subsystem No.	Component Choices
1	$x_{112} = x_{123} = x_{132} = x_{141} = x_{151} = 1$
2	$x_{212} = x_{222} = x_{231} = x_{242} = x_{251} = 1$
3	$x_{312} = x_{321} = x_{332} = x_{342} = x_{352} = 1$
4	$x_{412} = x_{422} = x_{431} = x_{441} = x_{452} = 1$

Unlike the Series-Parallel system, the problem formulation for the Parallel-Series system leads to a structure that is much more difficult to solve. While the algorithm described in Section 3.4 works well for relatively small systems, computational times start to increase rapidly as the problems grow in size. For a problem with the same data set from Table 3 and Table 4, i.e., four subsystems (all connected in a parallel setting) with five components each connected in series, and with the same value of R_S = 0.99, the algorithm was completely impractical. Hence, in order to solve problems of practical size (say 20 components); improved solution procedures are necessary.

3.7 Acceleration Schemes For The Algorithm

While we have provided an algorithm to solve the general IP formulations developed in this chapter, the algorithm is not capable of efficiently solving more large-scale problems that model general systems. Thus we have the option of either exploring enhancements to the algorithm that will accelerate its convergence to a solution, or to develop efficient heuristic procedures for larger

systems. We will explore the former option in the following sections of this chapter and then look at heuristic options in the next chapter.

The computational limitations of the algorithm described in Section 3.4 arise primarily out of the fact that we only eliminate a single infeasible integer point at each iteration. Hence any better solution procedure for large systems, should be aimed at either (i) eliminating more integer points at each iteration from the region within the relaxation that is infeasible for the original problem, or (ii) eliminating an entire section of this region rather than a single point or a set of integer points. Ideally, this elimination procedure should involve additional linear inequalities that can be easily handled. In the following sections, we identify a class of additional disjunctive systems for ***SP*** and ***PS***. For ***SP***, these disjunctive systems can yield additional valid inequalities which strengthen the linear relaxation by eliminating a region from the gap. For ***PS***, the disjunctive systems might be used as part of a better branching scheme with a branch-and-bound procedure.

3.7.1 Additional Disjunctive Inequalities To Strengthen LP Relaxation SP_0

Consider the reliability constraint in ***SP***:

$$\prod_{i=1}^{L} f_i = \prod_{i=1}^{L} \left\{1 - \prod_{j=1}^{n_i} (1 - r_{ij})\right\} \geq R_S$$

Clearly, this nonlinear constraint cannot be readily linearized. However, if we ignore the integrality of the binary variables, it is possible to define the feasible region of this constraint by means of a conjunction of finitely many disjunctive systems of linear inequalities. Furthermore, from the theory of disjunctive programming (Balas [1979]), it is possible to generate all valid inequalities for each such disjunctive system. We will show that the conjunction of these disjunctive systems defines the feasible region of the nonlinear reliability constraint of ***SP*** (ignoring integrality). To see how such disjunctive systems are represented, consider an example

with two subsystems. Define $0 < s_1, s_2 < 1$ such that $s_1 s_2 = R_S$. Now consider the ***SP*** system with reliability requirement $f_1 f_2 \geq R_S = s_1 s_2$. Then clearly all f_1, f_2 satisfying this will also satisfy at least one of $f_1 \geq s_1$, $f_2 \geq s_2$. That is, the disjunction $\{(f_1 \geq s_1) \mathrm{V} (f_2 \geq s_2)\}$ is valid for system ***SP***. This may now be generalized as follows:

Let, $0 < s_1, s_2, \ldots, s_L < 1$ such that $\prod_{i=1}^{L} s_i = R_S$. Then the disjunctive system $D(\boldsymbol{s}): \bigvee_{i=1}^{L} \{f_i \geq s_i\}$ for any such $\boldsymbol{s} = [s_1, s_2, \ldots, s_L]$ is valid for ***SP***. Notice that each element $\{f_i \geq s_i\}$ of such a system of disjunctions can be rewritten as $\sum_{j=1}^{n_i} \ln(1 - r_{ijk}) \leq \ln(1 - s_i)$, which in turn reduces to $\sum_{j=1}^{n_i} \sum_{k=1}^{K_{ij}} (x_{ijk} \ln(1 - p_{ijk})) \leq \ln(1 - s_i)$, a linear inequality constraint in $\boldsymbol{x}$. The nature of these disjunctive systems with respect to the reliability constraint can be readily appreciated when we look at an instance with two subsystems (i.e., with L=2).

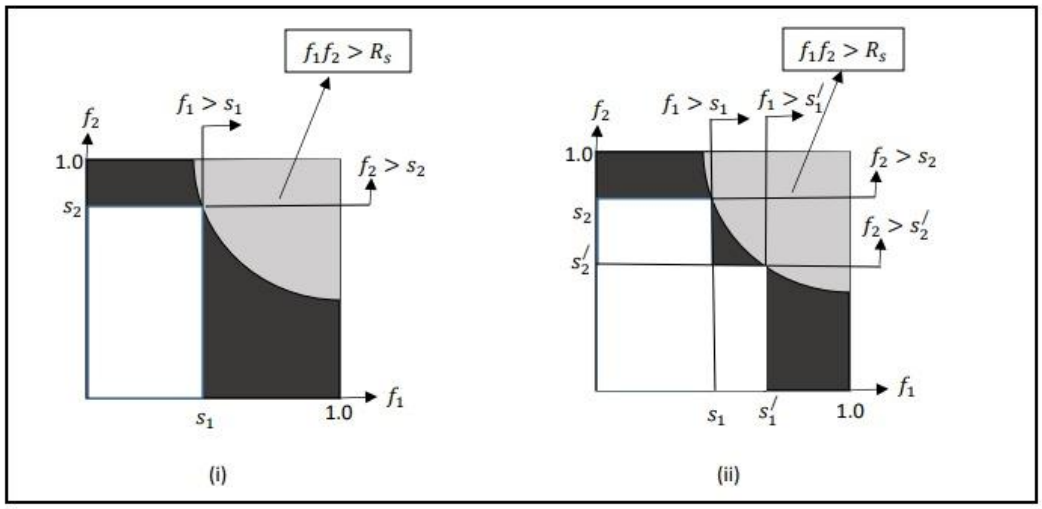

Figure 7 : Disjunctive systems for reliability constraint for *SP*

Referring to Figure 7(i) above, notice that disjunctive system $D(\boldsymbol{s}) = \{(f_1 \geq s_1) \mathrm{V} (f_2 \geq s_2)\}$ is a relaxation of the reliability constraint $f_1 f_2 \geq R_S$, but it contains additional points that violate $f_1 f_2 \geq R_S$, as depicted by the dark shaded area in Figure 7(i). However, consider when a second disjunctive system $D(\boldsymbol{s}') = \{(f_1 \geq s_1') \mathrm{V} (f_2 \geq s_2')\}$ is added to $D(s)$. Together, they contain less

infeasible region (that is shaded dark) in Figure 7(ii). Intuitively, it should be clear that if we have the collection of all such disjunctive systems, then they should have, in conjunction, the same feasible region as the reliability constraint. The following proposition proves this for the more general case.

Proposition 8: Let $\boldsymbol{\mathcal{D}}$ be the set of all disjunctive systems $D(\boldsymbol{s})$ such that $s_1, s_2, \dots, s_L \in (0,1)$ and $\prod_{i=1}^{L} s_i = R_S$. Then, the conjunction of the elements of the set $\boldsymbol{\mathcal{D}}$ is equivalent to the reliability constraint of ***SP***.

Proof: Since each disjunctive system $D(\boldsymbol{s})$ is valid for SP, the collection $\boldsymbol{\mathcal{D}}$ clearly represents a relaxation of the reliability constraint of ***SP***. Now, to prove that they both are equivalent, it would suffice for us to show that for any point infeasible in the reliability constraint there exists a disjunctive system in $\boldsymbol{\mathcal{D}}$ that is violated. Suppose $\{f_i = b_i;\ i = \{1,2, \dots, L\}\}$ is infeasible in the reliability constraint, so that $\prod_{i=1}^{L} b_i < R_S$. Now consider the following set of vectors $\left\{\boldsymbol{a} \in R^L : \prod_{i=1}^{L} a_i = R_S; b_i \le a_i \le \frac{R_S b_i}{\prod_{k=1}^{L} b_k}; \forall i\right\}$. Clearly a disjunctive system $D(\boldsymbol{a})$ exists in $\boldsymbol{\mathcal{D}}$ for each member $\boldsymbol{a}$ in the above set. Consider the subset of these elements $A = \left\{\boldsymbol{a} \in R^L : \prod_{i=1}^{L} a_i = R_S; b_i < a_i < \frac{R_S b_i}{\prod_{k=1}^{L} b_k}; \forall i\right\}$. Clearly A is a non-empty set and $\{f_i = b_i;\ \forall i\}$ violates all the disjunctive systems $\{D(\boldsymbol{a}) | \boldsymbol{a} \in A\}$. Hence the proof.

3.7.2 Valid Inequalities For *SP* From A Disjunctive System

Given a disjunctive system $D(\boldsymbol{s}): \bigvee_{i=1}^{L} \{f_i \ge s_i\}$ which is valid for ***SP***, the valid inequality as a consequence of $D(\boldsymbol{s})$ is derived as follows:

$$\bigvee_{i=1}^{L}\{f_i \geq s_i\} \equiv \bigvee_{i=1}^{L}\left\{\sum_{j=1}^{n_i}\sum_{k=1}^{K_{ij}}\left(x_{ijk}\ln\left(1-p_{ijk}\right)\right) \leq \ln(1-s_i);\ x_{ijk} \geq 0;\ \forall(i,j,k);\right\}$$

$$\equiv \bigvee_{i=1}^{L}\left\{\sum_{j=1}^{n_i}\sum_{k=1}^{K_{ij}}\left(x_{ijk}\left(\frac{\ln\left(1-p_{ijk}\right)}{\ln(1-s_i)}\right)\right) \geq 1;\ x_{ijk} \geq 0;\ \forall(i,j,k);\right\}$$

Using the theorem of Balas [1972], the valid inequality corresponding to the above system of disjunctions can be written as:

$$\sum_{i=1}^{L}\sum_{j=1}^{n_i}\sum_{k=1}^{K_{ij}}\left(x_{ijk}\left(\frac{\ln\left(1-p_{ijk}\right)}{\ln(1-s_i)}\right)\right) \geq 1$$

Proposition 9: Let $(q_1, q_2, \ldots, q_L)$ be a permutation of the numbers (1,2, …, L) and $\prod_{i=1}^{L} s_i = R_S$. Then for every such permutation the inequality

$$\sum_{i=1}^{L}\sum_{j=1}^{n_i}\sum_{k=1}^{K_{ij}}\left(x_{ijk}\left(\frac{\ln\left(1-p_{ijk}\right)}{\ln(1-s_{q_i})}\right)\right) \geq 1$$

is valid for ***SP***.

Proof: Notice that $\prod_{i=1}^{L} s_i = \prod_{i=1}^{L} s_{q_i} = R_S$. Thus, the stated inequality is a consequence of disjunctive system $\bigvee_{i=1}^{L}\{f_i \geq s_{q_i}\}$ and it must be valid for ***SP***. Hence the proof.

3.7.3 Additional Disjunctive Inequalities To Strengthen LP Relaxation PS_0

Similar to ***SP***, ***PS*** can also be described by a collection of disjunctive systems in place of its nonlinear reliability constraint. We now show that the conjunction of these disjunctive systems defines the feasible region of the nonlinear reliability constraint of ***PS***. To develop these disjunctive systems, consider an example with two subsystems. Define $0 < s_1, s_2 < 1$ such that

$(1-s_1)(1-s_2) = (1-R_S)$. Now consider a ***PS*** system with reliability requirement $(1-f_1)(1-f_2) \leq (1-R_S) = (1-s_1)(1-s_2)$. Then clearly for all feasible f_1, f_2 we must have at least one of $(1-f_1) \leq (1-s_1)$ or $(1-f_2) \leq (1-s_2)$. That is, the disjunction $\{(f_1 \geq s_1)\,\mathrm{V}(f_2 \geq s_2)\}$ is valid for system ***PS***. This may now be generalized as follows:

Let $0 < s_1, s_2, \ldots, s_L < 1$ such that $\prod_{i=1}^{L}(1-s_i) = (1-R_S)$.Then, the following disjunctive system $D(\boldsymbol{s}) = \bigvee_{i=1}^{L}\{f_i \geq s_i\}$ for any such $\boldsymbol{s} = [s_1, s_2, \ldots, s_L]$, is valid for ***PS***. Notice that similar to what we saw in the Section 3.7.1 each element $\{f_i \geq s_i\}$ of such a system can be written as a linear inequality $\sum_{j=1}^{n_i}\sum_{k=1}^{K_{ij}}(x_{ijk}\ln p_{ijk}) \geq \ln s_i$. Once again, the nature of these disjunctive systems with respect to the reliability constraint can be readily appreciated when we look at an instance with two subsystems (L=2) similar to the case of ***SP***.

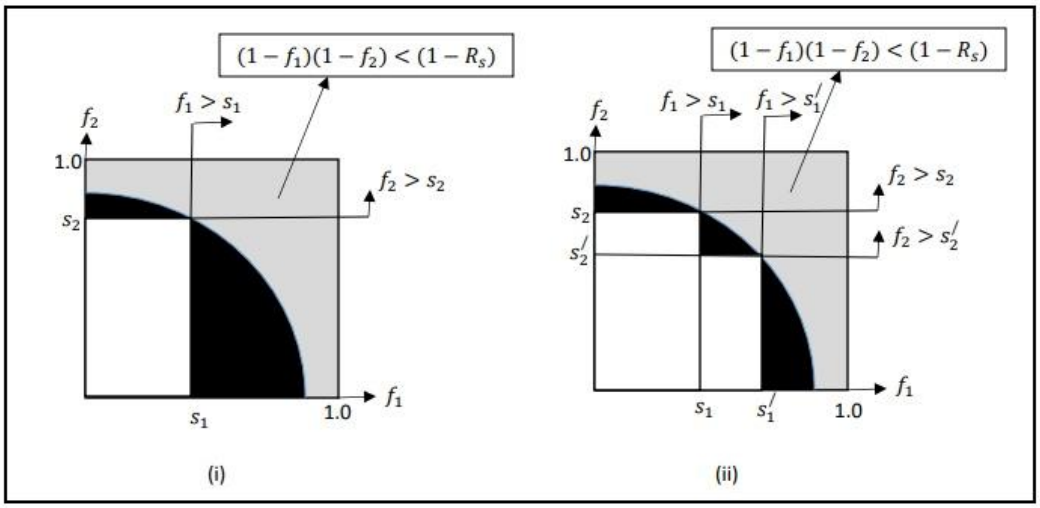

Figure 8: Disjunctive systems for reliability constraint for *PS*

Referring to Figure 8(i) above, notice that disjunctive system $D(\boldsymbol{s}) = \{(f_1 \geq s_1)\mathrm{V}(f_2 \geq s_2)\}$ is a relaxation of reliability constraint $(1-f_1)(1-f_2) \leq (1-R_S)$, but it contains additional points that are infeasible; this region is indicated by the shaded area (dark) in Figure 8(i). However, consider when a second disjunctive system $D(\boldsymbol{s}') = \{(f_1 \geq s_1')\mathrm{V}(f_2 \geq s_2')\}$ is added to $D(\boldsymbol{s})$.

Together, they contain less infeasible region (shaded dark) in Figure 8(ii). Intuitively, if we have the collection of all such disjunctive systems, then they should define, in conjunction, the same feasible region as the reliability constraint. The following proposition proves this for the more general case.

Proposition: Let $\boldsymbol{\mathcal{D}}$ be the set of all disjunctive systems $D(s)$ such that $s_1, s_2, \dots, s_L \in (0,1)$ and $\prod_{i=1}^{L}(1 - s_i) = (1 - R_S)$. Then, the conjunction of the elements of the set $\boldsymbol{\mathcal{D}}$ is equivalent to the reliability constraint of ***PS***.

Proof: Since each disjunctive system $D(\boldsymbol{s})$ is valid for ***PS***, the collection $\boldsymbol{\mathcal{D}}$ clearly represents a relaxation of the reliability constraint of ***PS***. Now to prove that they both are equivalent, it would suffice for us to show that for every infeasible solution to the reliability constraint there exists a disjunctive system in $\boldsymbol{\mathcal{D}}$ which is violated. Suppose $\{f_i = b_i;\ \forall i = \{1,2, \dots, L\}\}$ is an infeasible solution to the reliability constraint, so that $\prod_{i=1}^{L}(1 - b_i) > (1 - R_S)$. Now consider the following set of points $\left\{\boldsymbol{a} \in R^L : \prod_{i=1}^{L}(1 - a_i) = (1 - R_S); b_i \le a_i \le 1 - \frac{(1-R_S)(1-b_i)}{\prod_{k=1}^{L}(1-b_k)}; \forall i\right\}$. Clearly a disjunctive system $D(\boldsymbol{a})$ exists in $\boldsymbol{\mathcal{D}}$ for each point $\boldsymbol{a}$ in the above set. Consider the subset of these points $A = \left\{\boldsymbol{a} \in R^L : \prod_{i=1}^{L}(1 - a_i) = (1 - R_S); b_i < a_i < 1 - \frac{(1-R_S)(1-b_i)}{\prod_{k=1}^{L}(1-b_k)}; \forall i\right\}$. Clearly A is a non-empty set and the infeasible solution $\{f_i = b_i;\ \forall i\}$ violates all the disjunctive systems $\{D(\boldsymbol{a}) | \boldsymbol{a} \in A\}$. Hence the proof.

Proposition: Let $(q_1, q_2, \dots, q_L)$ be a permutation of the numbers (1, 2, …, L) and $\prod_{i=1}^{L}(1 - s_i)$ = (1-R_s). Then for every such permutation the disjunctive system

$$\bigvee_{i=1}^{L}\left(\sum_{j=1}^{n_i}\sum_{k=1}^{K_{ij}}\left(x_{ijk}\left(\frac{\ln p_{ijk}}{\ln s_{q_i}}\right)\right) \le 1\right) \text{ is valid for } \boldsymbol{PS}.$$

Proof: Notice that $\prod_{i=1}^{L}(1 - s_i) = \prod_{i=1}^{L}(1 - s_{q_i}) = (1 - R_S)$. Thus, the stated disjunction is a consequence of disjunctive system $\bigvee_{i=1}^{L}\{f_i \geq s_{q_i}\}$ and it must be valid for ***PS***. Hence the proof.

It should be noted that a single inequality cannot be readily written for the above disjunctive system. However, a branch and bound algorithm could be developed by using the above disjunction as part of a branching strategy.

In conclusion, we have provided an outline of possible acceleration strategies for an algorithm that seeks to solve the IP formulation exactly. In the next chapter we will explore the second alternative of developing appropriate heuristics to solve the problem.

4.0 Heuristic Approaches To Solving The Reliability Allocation Problem

Note: The work in this chapter was published in *Proceedings of the 5th Annual IE Research Conference* (Majety, Venkatasubramanian and Smith [1996]), and *Proceedings of the 6th Annual IE Research Conference* (Majety and Rajgopal [1997])

4.1 Introduction

Metaheuristics such as simulated annealing, tabu search, and evolutionary algorithms have been commonly adopted by researchers to solve many difficult problems, especially in combinatorial optimization. In this chapter we develop novel variants of two popular metaheuristic approaches to solve our reliability allocation problem.

Once again, we adopt 0-1 integer programming formulations where each binary variable corresponds to either a data point on the cost-reliability curve of a component, or a discrete option that is available to us, and our objective is to find the minimum-cost system that meets a minimum prespecified system level reliability. In general, the system reliability is a nonlinear function of all these variables. However, the primary goal of the methods described in this chapter is to find good solutions to the problem in a reasonable amount of time, even if they are not guaranteed to be optimal. It can be easily shown that the problem is NP hard by considering a particular case of ***SP*** system, viz. a series system. Hence, heuristic approaches make useful contributions to problems of this nature.

4.2 A Nested Simulated Annealing Algorithm

The first metaheuristic we develop is a variation of simulated annealing that we term "nested" simulated annealing (NSA). While we describe our algorithm in the context of a series-parallel (***SP***) system, the approach is readily generalized to other configurations. In traditional simulated annealing (SA), one starts with a feasible solution and identifies a feasible neighboring solution. If this new solution improves the objective function, it is immediately accepted, and a move is made to that solution. If not, the new solution is accepted (even though we are moving to an inferior solution) probabilistically based on some "annealing schedule." The latter term arises from an analogy with the annealing process for metals where the metal is heated to a certain temperature and then it is reduced gradually to a lower temperature. This temperature reduction may take place as a geometric progression, arithmetic progression or in some other appropriate way. Similarly in simulated annealing algorithms, for inferior solutions that are being considered for acceptance a probability of acceptance is defined based on the objective function value, and this probability is reduced gradually (like the temperature) in some systematic fashion as the algorithm progresses.

The efficiency of an SA algorithm depends on the definition of the feasible neighborhood of the current solution (Eglese, 1990). Unfortunately, in the context of our problem, it is not so easy to define a feasible neighborhood solution because the system reliability has to be explicitly computed for any point in order to verify whether it is feasible or not. We get around this by employing a novel approach where rather than restricting ourselves to only feasible solutions like a traditional SA algorithm, we also consider moves to infeasible solutions. For this we use two SA procedures nested within the same algorithm. The first SA focuses only on feasibility and

applies the principle of probabilistic acceptance with feasibility as the criterion, i.e., it is designed to screen solutions generated and allows us to probabilistically further consider any solution, whether feasible or infeasible. The second SA then focuses on cost and allows us to probabilistically accept a candidate solution advanced by the first SA, even if its cost is more than that of the previous solution. As the algorithm proceeds, it becomes increasingly difficult to accept infeasible solutions within the first SA, and it also becomes increasingly difficult to accept a solution that is not an improvement within the second SA; this is similar to a generic simulated annealing method.

The approach we use is based on the idea that when the problem has a difficult-to-satisfy constraint set, then rather than defining a feasible neighborhood, we could consider adding an exterior penalty function to the objective function. The notion of a penalty function is similar to what is used in the typical class of such methods for nonlinear optimization. However, we use a Nested SA algorithm instead of a penalty function. In principle, is quite similar to the penalty function approach. In a regular penalty function method, a penalty is applied by increasing the value of the objective function (assuming minimization) based on the degree of infeasibility. The rate of this penalty is increased as iterations progress so that algorithm will not move towards infeasible solutions because they become more expensive due to the penalty. In our approach, we have two nested SA algorithms. There is an acceptance probability in each SA. This probability is progressively reduced to make acceptance of infeasible solutions in the outer SA and acceptance of non-improving feasible solutions in the inner SA harder as the algorithm progresses through iterations. It is also worth noting that in our experiments with the Nested SA, we always obtained feasible final solutions (which is typically not the case with exterior penalty function methods where we typically have a final “refinement” step to obtain the optimum).

Before describing our nested SA algorithm, we define the "cooling" parameters $0 < \alpha_R, \alpha_Z < 1$ and the "temperature" parameters T_R, T_Z. The temperature parameter plays the role of defining a probability with which a move is accepted while the cooling parameter plays the role of altering the probability from one iteration to next iteration. Eventually the algorithm needs to stop with a well-defined stopping criterion. Examples of stopping criteria could be a fixed total number of iterations, if the improvement in the objective function in successive iterations falls below some threshold value, or a combination of both; other appropriate criteria might also be possible. The algorithm may now be specified is as follows:

STEP 0: Generate an initial feasible solution X with reliability R and cost Z. Define initial values for T_R, T_Z, and define an appropriate *stopping criterion*;

Set *accept = no*

STEP 1: If *stopping criterion* is met, **STOP**.

STEP 2: Generate a neighboring solution X' of X with reliability R' and cost Z'

i. If $R' \geq R_S$, then set *accept = yes*. If $R' < R_S$, then set *accept = yes* with probability $e^{-(R_S - R')/T_R}$;

ii. If *accept = yes* and $Z' \leq Z$, then set $X = X'$. If *accept = yes* and $Z' > Z$, then set $X = X'$ with probability $e^{-(Z' - Z)/T_Z}$;

iii. Set $T_R = \alpha_R T_R$; $T_Z = \alpha_Z T_Z$.

STEP 3: Set *accept = no*. Return to **STEP 1**.

Notice that the values of T_R, T_Z values are reduced through the iterations since $0 < \alpha_R, \alpha_Z < 1$. Due to this, the acceptance probabilities are also correspondingly reduced. This is analogous to temperature reduction in an annealing process. The probability of acceptance for an infeasible

neighboring solution is equal to $p_1 \times p_2$, where $p_1 = e^{-(R_S - R')/T_R}$ and either $p_2 = e^{-(Z' - Z)/T_Z}$ or $p_2 = 1$. In other words, the acceptance of a candidate solution is guaranteed if the solution is feasible and of lower cost than the previous solution. However, when one or both of these conditions are not true, we accept the solution probabilistically depending on the degree of infeasibility, the difference in cost, and the point on the cooling schedule at which the search currently resides. Probabilistic acceptance for violation of either condition becomes harder as the search proceeds.

4.2.1 Initial Feasible Solution

While an initial feasible solution might be generated in several different ways, we adopt the following approach:

i. Assume all f_i are equal for all subsystems *i*; compute $f_i = (R_S)^{1/L}$.

ii. Assume all r_{ij} are equal for all components *j* in each subsystem *i*; compute $r_{ij} = f_i^{1/n_i}$.

iii. Assign component *j* in subsystem *i* the value of p_{ijk} that satisfies $p_{ij(k-1)} < r_{ij} < p_{ijk}$, and assign it the corresponding cost c_{ijk}. If such a k does not exist, then a p_{ijk} value can be inserted into the data at an artificially high cost c_{ijk} so that this data point will never appear in the optimal solution.

This is a simple approach that guarantees that the resulting solution is feasible because the system with reliabilities of $\{r_{ij}\}$ is already feasible (with $R = R_S$) and each chosen component reliability is greater than or equal to r_{ij} thus $R \geq R_S$.

4.2.2 Neighboring Solution

It is difficult to arrive at a precise and comprehensive definition of a neighborhood for our problem; this can often be the case with some combinatorial problems. One approach might be to first fix the reliability of each subsystem at some value and then compute the individual component reliabilities. Another approach might be to arbitrarily pick some component (or components) and randomly increase or decrease the current reliability value to the next higher or next lower value in their reliability sets. Other options are also possible. While the neighborhood definition is complicated in and of itself, assuring feasibility (i.e., meeting the minimum system reliability level) for the neighboring solution further adds to this complexity. The feasibility of a neighboring solution can be ensured but not without considerable computational effort, which could be cumbersome. Furthermore, restricting the search to just the feasible region of a constrained problem also often results in inefficient convergence and suboptimal final solutions. Therefore, we are willing to accept an infeasible neighboring solution, but with some probability as described in the algorithm. The Nested SA process makes the acceptance of such infeasible solutions more unlikely as the search progresses.

We define our "neighborhood" as follows:

i. Define $k_{index}(i,j) = \{k \in (1,2,\dots,K_{ij} | r_{ij} = p_{ijk}\}$ Note that the $\{p_{ijk}\}$ are assumed to be ordered by *k*.

ii. Randomly select some (not necessarily the same number each time) components in the subsystem defined by (i,j) and randomly increase or decrease $k_{index}(i,j)$ by 1 for these components only. If $k_{index}(i,j)$ corresponds to the lowest (highest) value for a selected component then we only increase (decrease) its value.

The neighboring solution obtained in this fashion might be infeasible. However, it is easily seen that every solution can be reached from a given solution (feasible or infeasible) eventually with such selection of a neighboring solution. Hence the convergence to feasible solution is guaranteed in the outer SA eventually, while convergence to better solutions is guaranteed in the inner SA.

4.2.3 Examples And Results

We illustrate the algorithm with two example problems. We adopted the standard geometric cooling schedule for both the outer SA that considers feasibility and the inner SA that considers the objective. We use $\alpha_R = \alpha_Z = 0.99$ for both examples, T_R = 1000 for Example 1, T_R = 5,000 for Example 2; and T_Z = 10,000 for Example 1, T_Z = 25,000 for Example 2. The stopping criterion used was the total number of solutions considered, which was set to 20,000 for Example 1 and 30,000 for Example 2. The results are summarized below.

4.2.4 Example 1

We chose an ***SP*** system with nine components for Example 1. As a matter of convenience, we used the same reliability level options for all components in our data set. For each component, we assigned randomly generated cost values for each reliability level in the data set as shown in Table 6. We ensured an increasing trend of cost w.r.t. reliability while assigning costs. However, we did not assume any kind of a functional relationship (and in particular, any convex functional relationship); in fact, this is guaranteed due to our random generation of costs. We also account for the situation where a component might be irrelevant to the system reliability. To address this situation, for option k = 1, we specify a very low reliability (0.001) and zero cost. If a component

j in subsystem i is assigned the option k = 1 in the optimal solution, then that component is clearly redundant and hence can be removed from the system, that is, it is a 'blank.' The components are then randomly assigned to three subsystems of an ***SP*** system, with L_i equal to 3, 4 and 2 components for subsystems i = 1,2, and 3, respectively. The solution search space thus has a total of 12^9 candidate points.

From our experience in experimenting with the Nested SA, the best final solutions are obtained when the algorithm visits mostly infeasible solutions. We speculate that this might be because there are many more infeasible solutions in the search space than feasible ones and the optimal solution is likely "surrounded" by many of these infeasible solutions; thus allowing a wider search is better. For example, (see Table 8) for the best solution observed across 30 independent runs; the ratio of infeasible to feasible solutions (**IFR**) is 6.52. When we restricted this ratio to values under 1.5 in the algorithm, the best feasible solution cost from these runs was quite suboptimal (above 600 as compared to the optimum cost of 500.60). The Nested SA was run with no restrictions on **IFR**, and statistics are presented in Table 8 for 30 different independent runs. The best solution across all runs is within 6.65% of the global optimum solution (which we obtained by enumeration for the purpose of comparison). The search space considered by the SA (20,000 solutions) was a minuscule fraction of the total of 12^9 (0.0004% approx.).

4.2.5 Example 2

We chose Example 2 to demonstrate that the model suggested in this research addresses the redundancy issue as well. For this purpose, we added two more components to the above system in subsystem 1 (see Table 7), thus enlarging the search space to 12^{11} points, but without altering the optimal solution (with a value of 500.60). However, for these additional components (j=4, 5

for i=1) we intentionally generated higher costs for the option available at each "real" reliability levels (i.e., other than for k=1, which has a cost of 0). As in Example 1, the best solutions are obtained when the algorithm visited mostly infeasible solutions. For the best solution observed across 30 independent runs of the algorithm, the ratio of infeasible to feasible (**IFR**) solutions is 10.11. The Nested SA was run again with no restrictions, and statistics are presented in Table 8 for 30 different independent runs. This time the best solution across all runs is within 7.82% of global solution with the SA again considering only a tiny fraction of the search space (30,000 points of 12^{11}). It is interesting to note that in all the 30 different runs; the best solution always selected k=1 (cost of zero and reliability of almost zero) for the two additional components with only high-cost options available. Thus, the algorithm clearly indicated that these components are not necessary to the system and are 'blanks.' Or in other words, the optimal configuration of the system does not need these two components. This demonstrates that the problem formulation developed in this work also clearly address redundancy design.

Table 6: Cost and reliability data for Example 1 { c_{ijk} }

i	j	**RELIABILITY VALUES**											
		k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=11	k=12
		0.001	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	0.99
1	1	0	4.05	16.3	40.4	67.4	95.7	135.8	186.05	251.1	339.8	440.5	597.7
1	2	0	3.65	17.8	36	59.4	78.5	169.6	224.45	303.8	392	505.3	654.5
1	3	0	9.1	22.4	44.5	71.6	105.1	148.9	198.1	276.8	374.3	496.8	633.7
2	1	0	4.35	14.1	29.2	50.5	78.2	117.6	170.9	248.6	347.9	463.8	609.4
2	2	0	3.15	10.8	32	52	183.3	222.1	278.8	350.3	434.2	539.3	699.2
2	3	0	7.8	22.9	43.9	70.8	101.5	143.7	202.05	276.7	370.2	495.2	628.5
2	4	0	8.75	18.8	42.8	72.1	106.3	151.2	210.95	290	370	482.8	636.6
3	1	0	5.45	16.5	36.5	60.7	191.2	230.8	282	354	449.5	572.8	703.3
3	2	0	2.05	7.67	23.9	102	128.8	164.4	207	271.3	362.8	481	623.4

Table 7: Cost data for additional components in Example 2: { c_{ijk} }

i	j	RELIABILITY VALUES											
		k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10	k=11	k=12
		0.001	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	0.99
1	4	0	231.7	272.8	456	689.4	821	989.6	1354.5	2043.8	3492	4605	6754.5
1	5	0	219.1	232.4	364.5	591.6	735	978.9	1148.1	1676.8	2474.3	3697	5733.7

Table 8: Summary of results for Examples 1 and 2 with nested SA over 30 runs of each

	Max. Cost	Mean Cost	Min. Cost	No. of Iterations	Feasible Solutions	Best Configuration			Best Reliability	CPU time (sec/run)
Ex 1	662.1	572.7	534	20000	2660	4-5-4	2-5-3-4	5-8	0.85027	2.54
Ex 2	726.2	599.5	541	30000	2699	7-4-3-1-1	2-4-3-3	5-8	0.85092	5.70
Optimal (via enumeration)			501	-	-	3-6-5	4-3-2-3	5-8	0.85017	~6 hrs.

4.3 An Evolutionary Algorithm

The second heuristic approach we develop is an evolutionary algorithm. A common approach to dealing with problems that are explicitly constrained is the use of traditional penalty functions which penalize infeasible solutions found by the algorithm. Here, we generalize this penalty concept by also penalizing solutions that are not necessarily infeasible, but clearly undesirable. The rationale behind this approach is that for many problems, the general vicinity of the optimal solution is known. For example, in any optimization problem with a linear objective function, the optimal solution is known to be near the boundary of feasible region, and thus feasible solutions far from the boundary may be classified as undesirable. Here, we develop an evolutionary algorithm and demonstrate that penalizing undesirable feasible solutions (in addition to infeasible solutions) yields much better results and accelerated convergence.

Consider a penalty function approach to the following general optimization problem:

$$\min\ z(\boldsymbol{x})$$

$$\text{s.t. } \boldsymbol{x} \in X(E),\ \boldsymbol{x} \in X(D)$$

where, $X(E)$ defines a part of the feasible region determined by constraints that are "easy" while $X(D)$ defines the region determined by the "hard" constraints. Penalizing the hard constraints, the modified problem is:

$$\min\ z(\boldsymbol{x}) + \varphi[\boldsymbol{x}, d(D)]$$

$$\text{s.t. } \boldsymbol{x} \in X(E)$$

where $d(D)$ is a measure of the degree of infeasibility for the constraint set defining $X(D)$, and $\varphi[\boldsymbol{x}, d(D)]$ is a penalty function appended to the objective function. For example, the Lagrangian relaxation approach uses Lagrange multipliers to bring the difficult constraints into the objective function. In heuristic optimization methods most researchers (e.g., Coit et al. [1996], Bean and Hadj-Alouane [1992]) commonly adopt a penalty function based on the degree of infeasibility so as to discourage infeasible solutions from being selected. While the use of such a penalty drives solutions to feasibility it does not necessarily guarantee speedy convergence to the neighborhood of the optimum. We now define a different penalty function.

4.3.1 Penalty Function

We first introduce some notation: suppose *N* is some feasible neighborhood of the optimum solution. Then we term any point outside *N* as an "undesirable" solution. It should be noted that an undesirable solution might be feasible or infeasible. Clearly, the complement of *N* is a union of two disjoint regions (i) *I*: the infeasible region, and (ii) *F**N*: the feasible region excluding *N*. The modified problem may then be stated as:

$$\min\ z(\boldsymbol{x}) + \varphi[\boldsymbol{x} \in I, d(D)] + \varphi[\boldsymbol{x} \in F \backslash N, d(N)]$$

s.t.

$$\boldsymbol{x} \in X(E)$$

where, $\varphi[\boldsymbol{x} \in I, d(D)]$ is the usual penalty function applied to infeasible solutions, while $d(N)$ is a metric to measure the distance of the solution from the boundary of the feasible region ($d(N)$ =0 if $\boldsymbol{x} \in N$) and $\varphi[\boldsymbol{x} \in F \backslash N, d(N)]$ is a penalty function for feasible solutions that are not in *N*.

Our approach is predicated on the fact that for many difficult combinatorial problems the neighborhood of the optimal solution can be predicted. As an example, for an integer program with a linear objective, it is known that the optimum solution is in the vicinity of the boundary of the feasible region. For such problems the neighborhood *N* can be defined as all feasible solution within some distance of the boundary. The distance parameter is usually problem specific. It should be big enough so that the neighborhood defined by it contains the optimal solution; yet small enough that the penalty is applied to most feasible solutions that are undesirable. The selection of such a parameter can in general be quite difficult in most generic problem instances. We therefore suggest a dynamic procedure. In the initial stages, the distance parameter is large so that most feasible solutions escape this penalty and look more attractive than infeasible solutions. As the algorithm continues, the distance parameter is progressively reduced so that undesirable feasible solutions become increasingly unattractive, and the search focuses on finding the optimum. Our procedure is now described in detail and illustrated.

4.3.2 Evolution Strategy

Evolution strategy (ES) refers to an algorithm that tries to simulate the evolution process. A detailed description of the method may be found in Back et al. [1991]. Briefly, in the simulation

of an evolution process one tries to find links between the characteristics of an offspring and its parents. The usual procedure adopted is as follows: the problem is encoded in terms of an *n*-dimensional vector $\boldsymbol{x}$, and the objective is to find a vector that a minimizes a fitness function $f(\boldsymbol{x})$. Initially, a population of parent vectors $\{\boldsymbol{x}\}$ is generated randomly. Several offspring vectors $\{\boldsymbol{o}\}$ are generated from one, two or more parents by means of recombination and/or mutation strategies. These strategies constitute the heart of any evolutionary process, where recombination refers to the specific process used to generate offspring from the parent vectors and mutation refers to specific process used to produce random small changes in one or more characteristics of the offspring vectors. The specifics of these strategies could vary depending on the particular algorithm in use and particular application in question.

After the generation of the offspring vectors a selection process is conducted based on fitness function values. Depending on the algorithm in use, this selection could either be restricted to just the offspring set, or it could be from a combined set of offspring and parent vectors. The selected vectors are then given parent status and the procedure is repeated until a desired degree of convergence is achieved.

In this work, we adapt a (μ,λ)-ES (Back et al. [1991]). In this variant an initial population of μ vectors is generated, and from these parents, an offspring population of λ vectors is created by means of recombination and/or mutation. Then from the offspring population the best μ vectors are selected as the parent population for the next generation. Based upon extensive preliminary experimentation, a value of 7 was adopted in this work for the ratio λ/μ. The following sections describe in more detail the steps adopted for the reliability optimization problems at hand.

4.3.3 Encoding

As in the previous section we define $k_{index}(i,j) = \{k \in (1,2,\ldots,K_{ij} | r_{ij} = p_{ijk}\}$. By defining n_0 =0, the vector $\boldsymbol{x}$ is then defined such that the entries (n_i + 1) to (n_i + n_{i+1}) correspond to $k_{index}(i+1,1)$ to $k_{index}(i+1, n_i+1)$. For example, consider a system with three subsystems; the first subsystem has n_1 = 3 components, the second subsystem has n_2= 4 components, and the third subsystem has n_3= 2 components. Then the encoding {2,3,4, 4,5,6,7, 5,8} indicates that the three components in subsystem 1 take on the reliability values $p_{112}, p_{123}, p_{134}$, respectively, the four components in subsystem 2 take on the reliability values $p_{214}, p_{225}, p_{236}, p_{247}$, respectively and the two components in subsystem 3 take on the reliability values p_{315}, p_{328}, respectively. Each of the components with a reliability of p_{ijk} assumes a cost of c_{ijk}. Given the values for c_{ijk} and p_{ijk}, the cost and the reliability of the system may be obtained for the encoding.

4.3.4 Recombination

Back et al. [1991] describe two different recombination techniques that may be adopted for a problem: (i) discrete recombination and (ii) global recombination. In discrete recombination two parents are selected randomly and the offspring is created from these two parents. In global recombination two parents are selected randomly but with replacement to create each offspring vector. Within each recombination (discrete or global) the creation of an entry in the offspring vector is done in two ways: (i) discrete selection where the offspring entry is selected randomly from one of the two parent entries, and (ii) intermediate selection where the offspring entry is selected randomly as an intermediate value between the two parent entries. Thus, we have four

recombination procedures in all. In this work we consider only global intermediate recombination (GI) since it appeared to consistently produce the best results based on numerical experimentation.

4.3.5 Mutation

Mutation is induced by means of a vector of standard deviations $\boldsymbol{\sigma}$ (initially randomly generated). Each of the entries in the mutated offspring $\boldsymbol{o}$ follows a normal distribution with mean equal to the value of the offspring entry before mutation ($=o_i$), and standard deviation equal to σ_{o_i}. The offspring vector obtained from recombination may undergo mutation as defined by the following equation:

$$o_i \equiv o_i + int\left(\sigma_{o_i} * N(0,1)\right);\ \ \forall i$$

To understand the process, say that a mutation leads to an improved solution. Such a mutation is termed "successful." We adopt the 1/5-success rule as a method for controlling the mutations (Back et al. [1991]): if the ratio of successful mutations to all mutations is greater than 1/5 then the standard deviations of all entries in the offspring vector are increased by a factor 1/*b*, while if the ratio is less than 1/5 then all standard deviations are decreased by a factor of *b*. We use a value of b=0.82 based on trial and error. This ratio is measured for every 10n trials and then a multiplicative factor of $N(0,1)$ is applied, where N (0,1) is a standard normal variable.

Note that when the standard deviation is increased, the resulting offspring entry tends to be farther away from the original entry. If the successful mutation ratio is higher (more than 1/5), then mutation is encouraged with a mutated offspring entry that is relatively far away from the original value before mutation. On the other hand, if the successful mutation ratio is smaller, then mutation

is not encouraged because of the smaller standard deviation, and the resulting mutated offspring is likelier to resemble the original offspring more closely.

4.3.6 Structure Of Penalty For The Reliability Allocation Problem

We adopt a simple distance-based penalty function for infeasible solutions. This is defined as $\varphi[\boldsymbol{x} \in I, d(D)] = C(R_S - R(\boldsymbol{x}))$; where $R(\boldsymbol{x})$ is the reliability of the system corresponding to $\boldsymbol{x}$, and C is a constant whose value is suitably selected for a given problem. In addition to the penalty for infeasible solutions we also impose an additional dynamic penalty for undesirable feasible solutions. This is defined as $\varphi[\boldsymbol{x} \in F \backslash N, d(N)] = C'(R(\boldsymbol{x}) - R_S - \Delta)$. Here Δ is defined as the distance of the acceptance zone from the boundary of the feasible region into its interior (see Figure 9).

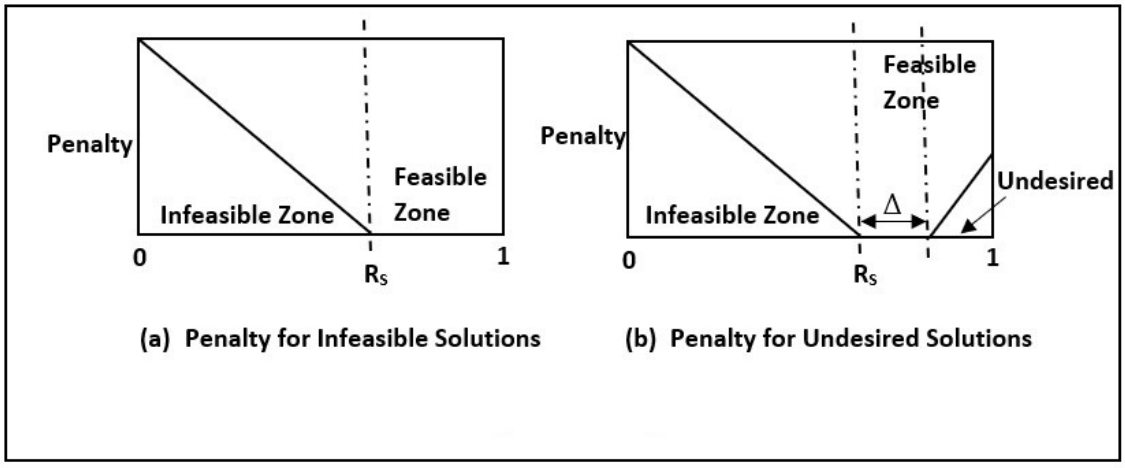

Figure 9: Penalty function

In general, the cost of a system increases as the desired system reliability increases. It is thus intuitively clear that the optimal solution will try to meet the reliability requirement exactly, or more likely (because of the discrete nature of the choices), exceed it by a minimal amount. Initially

a large value is assigned to Δ and its value is progressively reduced as the algorithm proceeds. We adopt a geometric reduction of Δ with a coefficient of $\alpha >$ 0.9. It is important to assign an appropriate value for α so that the distance between the boundary of the feasible region and the feasible penalty region decreases neither too quickly nor too slowly. The effect of this dynamic penalty on the undesired feasible region is now studied and the results contrasted with a traditional penalty function approach where none of the feasible solutions are penalized.

4.3.7 Examples And Results

Example-1: *SP* system:

We choose a system with nine components and three subsystems with 3, 4 and 2 components in each of these, respectively (Figure 10(a)).

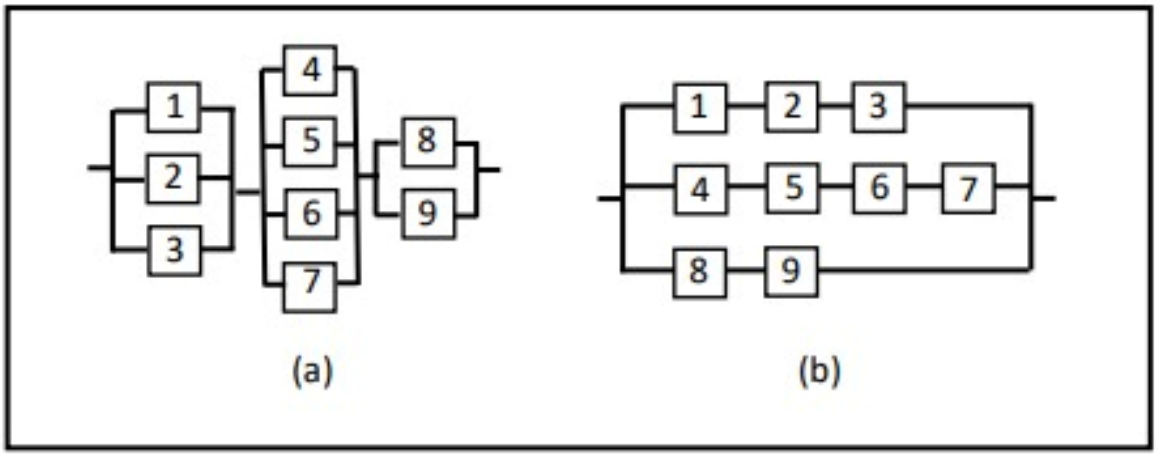

Figure 10: (a) *SP* system (b) *PS* system

Table 9: Reliability – Cost data for examples

i	*j*	Reliability Values											
		k=1	*k*=2	*k*=3	*k*=4	*k*=5	*k*=6	*k*=7	*k*=8	*k*=9	*k*=10	*k*=11	*k*=12
		0.001	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	0.99
1	1	0	4.05	16.30	40.40	67.35	95.70	135.75	186.05	251.05	339.80	440.45	597.70
1	2	0	3.65	17.75	36.00	59.35	78.50	169.60	224.45	303.80	391.95	505.30	654.45
1	3	0	9.10	22.35	44.45	71.55	105.1	148.85	198.10	276.75	374.25	496.80	633.65
2	1	0	4.35	14.10	29.15	50.45	78.20	117.55	170.90	248.55	347.90	463.75	609.40
2	2	0	3.15	10.80	31.95	52.00	183.25	222.10	278.80	350.30	434.15	539.30	699.15
2	3	0	7.80	22.85	43.85	70.80	101.45	143.70	202.05	276.70	370.20	495.15	628.50
2	4	0	8.75	18.80	42.80	72.05	106.25	151.20	210.95	289.95	370.00	482.75	636.60
3	1	0	5.45	16.45	36.45	60.70	191.20	230.75	282.00	354.00	449.50	572.75	703.30
3	2	0	2.05	7.67	23.87	101.90	128.81	164.35	207.00	271.25	362.80	480.95	623.40
Global Optimal Assignment for SP-system Example-1 = 3-6-5—4-3-2-3—5-8 with R = 0.8502 and Cost = 500.60													
Global Optimal Assignment for PS-system Example-2 = 3-3-3—2-2-2-2—10-10 with R = 0.8515 and Cost = 892.75													

For ease of exposition, we assume that reliability level options available for each component are the same, but with varying costs depending upon the component in question. For each component, we once again assign randomly generated cost values to the corresponding reliability values in the data set, while ensuring an increasing trend of cost vs. reliability. However, no other relationship is assumed. This data is given in Table 9. Notice that for $k = 1$ in the optimal solution, all components have very low reliability and zero cost. If any component (i,j) is assigned $k = 1$ in the optimal solution that component is deemed redundant and can therefore be removed from the system. Choices from the available option are randomly assigned to each component in the three subsystems in order to generate the initial population.

Option-1: Penalty for infeasible solutions only

For the example, the best parameter values based on experimentation were μ=60, λ=420 and C = 7500. The stopping criterion was 50 generations. Thus, we perform a total of 60 + 420×50 = 21060 fitness function evaluations, which corresponds to just 4.1×10^{-4} percent of the total combinatorial search space (=12^9). The algorithm converged to the same final solution (bottom of Table 9) in each of 10 different runs – irrespective of the random seed used – and showed convergence as displayed in Figure 11.

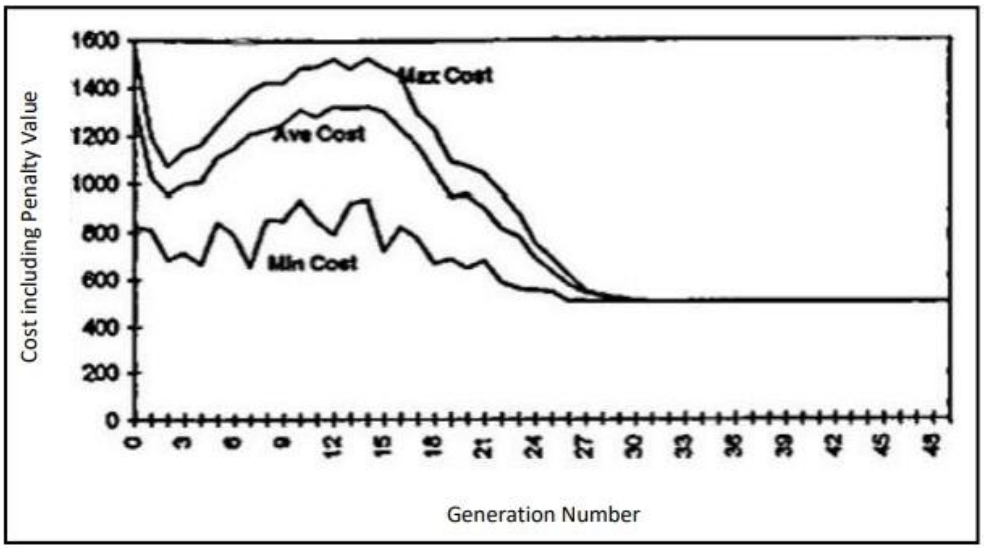

Figure 11: Convergence of ES for *SP* with option-1

However, the algorithm appears to be sensitive to the values of μ and C. At lower values of μ the algorithm converges to a feasible solution but to one that is distinctly inferior. Similarly, while for a value of C=7500 the algorithm performed adequately; for relatively low values of C, solutions with less cost are not sufficiently penalized and the algorithm converges to an infeasible solution with a very low cost. Conversely, if C is too high then the penalty is too high and the algorithm converges too quickly to a local optimum while missing nearby low-cost solutions.

Option-2: Penalty for undesirable solutions

For this option, we use the same values for μ and λ as with the first option. However, in this case we add the dynamic penalty function for undesirable feasible solutions in addition to the penalty for infeasible solutions. Based on our experiments we adopted $\alpha = 0.95$ and $C = 1200$ for the ***SP*** problem. Convergence to the optimal solution with this option is illustrated in Figure 12. It is clearly seen that the convergence is quicker than with option 1. It takes up to 32 generations to converge without the dynamic penalty whereas it takes only 23 generations when this option is adopted. This results in savings of 9×420 = 3780 function evaluations (≅17.9%). Also, it was found that the dynamic option converges to the optimum even with smaller values for μ (such as 40) with $\lambda=7\mu$. With μ=40 the savings in function evaluations is (32×420) -(23×280) = 7000 (≅33.2%).

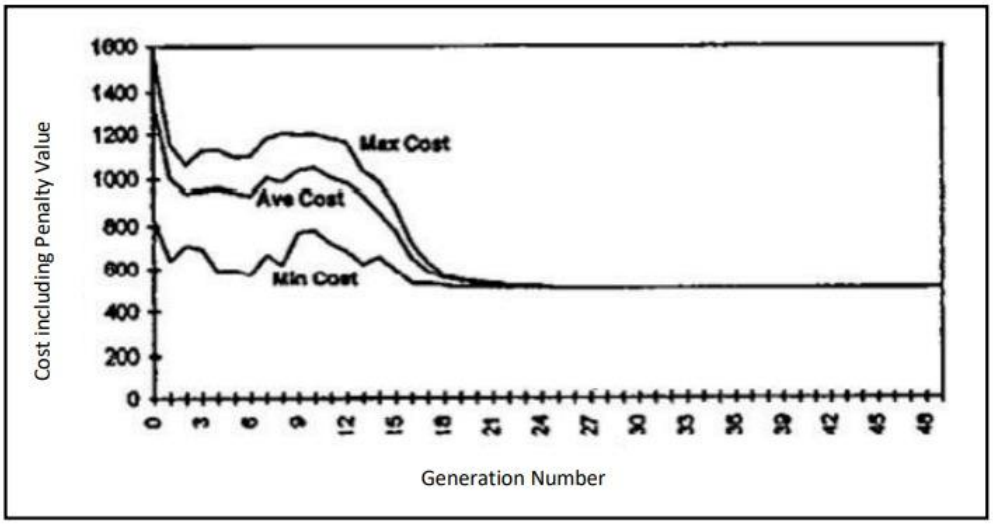

Figure 12: Convergence of ES for *SP* with option-2

Example-2: *PS* System:

For the ***PS*** problem, we used the same data as for the SP problem except that the three subsystems are connected in parallel as shown in Figure 10(b). Option-1 with penalty for infeasible

solutions alone resulted in non-convergence despite many trials with various values for the parameters. However, Option-2 with a penalty for undesired solutions resulted in convergence (see Figure 13) to the final configuration shown at the bottom of Table 9. This clearly emphasizes the utility of the additional penalty for undesired solutions.

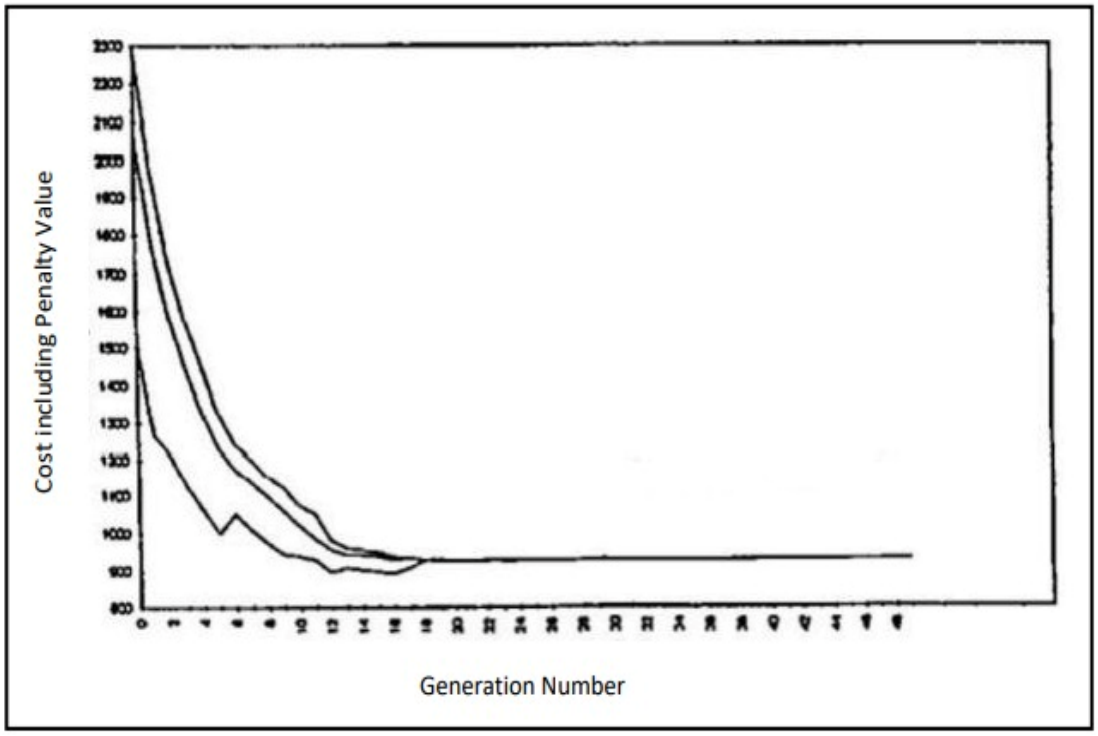

Figure 13: Convergence of ES for *PS* with option-2

4.4 Conclusions

In this chapter we address the fact that large problems cannot be solved optimally using the integer programming formulation of the previous chapter, thus necessitating the development of suitable heuristics. In particular, we develop two different heuristics to solve this *NP*-Hard combinatorial problem.

First, we developed a novel nested SA algorithm. The results are encouraging and the solution procedure clearly identifies near-optimal configurations of redundant components in the system, while maintaining feasibility of the final solution. Note that the Nested SA does not require the extensive tuning and parameter setting of many penalty function approaches. Only a cooling schedule needs to be specified, and a relatively straightforward geometric cooling schedule appears to work quite well. We did not consider other constraints such as weight or volume; however, it would be easy to extend the formulation of the problem to handle these additional constraints. While applying the Nested SA, one can include more SA procedures for the feasibility of these additional constraints. We believe that it is worthwhile investigating similar Nested SA algorithms for other combinatorial problems.

Second, we generalize the concept of a penalty function to include not just a penalty for infeasible solutions but also for undesirable solutions. This generalization is seen to result in accelerated convergence. The solutions developed in this chapter are well suited to address the reliability allocation problems addressed in this document. We adopted an evolution strategy to solve the combinatorial problem of reliability allocation. The initial results are very encouraging, and the technique offered optimal solutions to the example problems. An interesting and conclusive result is that the inclusion of penalty function for undesired feasible solutions enhanced the results in almost every case tested.

However, there are several open issues. First, a robust and problem-independent penalty function needs to be identified. For penalizing infeasible solutions, Coit et al. [1996] have suggested an adaptive penalty function for infeasible solutions. A similar functional form could possibly be adapted for the penalty for undesired feasible solutions as well. Secondly, the efficiency of such penalty functions depends on the definition of a neighborhood in a generic sense.

Ideally, the development of the penalty function should incorporate some user input in defining the neighborhood while maintaining its efficiency; this is a challenging task.

5.0 Optimum Test Plans

Note: The work in this chapter was published in *IIE Transactions* (Rajgopal, Mazumdar and Majety [1999])

5.1 Introduction

While the previous two chapters were focused on the design phase, this chapter focuses on the second broad topic of this dissertation, namely the testing of a designed system before it is deployed. System developers like to ensure that the systems that they design meet certain specified reliability levels. Therefore, it is essential for a system to be appropriately tested before deployment. When the tests are conducted, we require some criterion to use along with the test results so as to determine whether the system is deemed to be acceptable or unacceptable. Unfortunately, no statistical test can be perfect and every test has some probability of error associated with it. It is possible that based on the selected criterion a test might reject a good system on occasion; this is referred to as Type-1 error. Conversely, it is possible that a test might accept a bad system on occasion; this is referred to as Type-2 error. Thus, in addition to defining suitable values for the parameters associated with the test rule, a developer must also ensure that the probability of each type of error occurring is below some specified threshold value. Moreover, another important fact to keep in mind is that these tests incur costs and can turn out to be expensive. Therefore, while developing appropriate tests to demonstrate system reliability, we must also take test costs into consideration.

It should be obvious that in general, testing units of assembled systems is more complex and expensive than testing units of individual components. Thus, our preference would be to test only the components and make statements about the reliability of the overall system based on how these components are combined to form the system. This scenario has been studied extensively for various system configurations and under various assumptions (Easterling et al.[1991], Rajgopal and Mazumdar [1998], Rajgopal et al. [1999], and Rajgopal and Mazumdar, [2000], Rajgopal, J. and Mazumdar, M. (2001) etc.). However, in the presence of imperfect interfaces between components it will not be adequate to test only the components. This is because the interfaces introduce their own uncertainty and with imperfect interfaces it might not be possible to draw conclusions about the system simply by testing components; the only way to account for this is by testing the entire assembled system since we cannot "test" the interfaces by themselves. The approach studied in this chapter will be to test the system but to allow for the possibility of supplementing it with some component tests as well so as to minimize the total associated test costs. In particular, we address optimum test plans for a series system where interface reliabilities are also considered.

5.2 Notation

PARAMETERS

α	The maximum acceptable probability of Type 1 error (producer's risk)
β	The maximum acceptable probability of Type 2 error (consumer's risk)
n	Total number of components in the system
j	Index for component $j \in \{1,2,\dots,n\}$

λ_j Expected failure rate of component j;

Failure time of each component is an exponentially distributed random variable with a mean of λ_j^{-1}

λ_I Expected failure rate of interfaces;

Failure time of an interface is an exponentially distributed random variable with a mean of λ_I^{-1}

$\lambda_C = \sum_{\forall j} \lambda_j$ = The sum of the mean component failure rates

R_S System reliability = exp $(-\lambda_I - \lambda_C)$; assuming mission time is scaled to one unit of time.

R_1 Lower limit on reliability such that if $R_S \geq R_1$ then the system is considered to be definitely acceptable

R_0 Upper limit on reliability such that if $R_S \leq R_0$ then the system is considered to be definitely unacceptable;

$1 > R_1 > R_0 > 0$ and $\alpha + \beta < 1$

c_j Cost to test component j for one unit of time

c_S Cost of system level test for one unit of time

VARIABLES

t_j units of time for which to test component j with replacement of failed components

t_S units of time for which to test system with replacement of failed systems

X_j Number of failures observed for component j while testing for t_j units of time

X_S Number of system failures observed while testing for t_S units of time

X Total Number of failures observed = $\sum_j X_j + X_S$

m a number such that if $X \leq m$ the system is accepted; otherwise the system is rejected

Note that t_j, t_S and m are decision variables while X_j, X_S and X are random variables.

5.3 Problem Formulation

We test units of each component as well as of the system, with each test unit having its own test time and with replacement of failed units. Our acceptance criterion compares the total number of failures observed across all units tested (components as well as system) with some critical number, and if the number of observed failures is below this cutoff value we label the system as acceptable; this is similar in spirit to the criterion used with component testing or system testing alone. In the absence of any *a priori* information on component reliabilities or failure rates, it has been proven in several papers (Yan and Mazumdar [1986], Rajgopal and Mazumdar [1995] & [1996], Altinel [1992], etc.) that the optimal test times for all components are identical. We will simplify our formulation by denoting this common test time for each component as $t_C = t_1 = t_2 = \cdots = t_n$.

Then the generic problem formulation is as follows:

$$Minimize\ C = c_S t_S + \left(\sum_{j=1}^{n} c_j\right) t_C \qquad \textbf{5.1}$$

subject to

$$\Pr\left\{X_S + \sum_j X_j \le m | R_S \ge R_1\right\} \ge 1 - \alpha \qquad \textbf{5.2}$$

$$\Pr\left\{X_S + \sum_j X_j \le m | R_S \le R_0\right\} \le \beta \qquad \textbf{5.3}$$

$$t_S, t_C \geq 0$$

Note that (5.2) indicates that Type-1 error cannot exceed α while (5.3) ensures that Type-2 eror cannot exceed β. We now make the following important definition that will be needed to develop the mathematical basis for our plans.

Definition 5.1: $\phi_m(\gamma)$ is defined as the mean of a Poisson random variable Y that satisfies $\Pr(Y \leq m) = \gamma\,; 0 \leq \gamma \leq 1$.

Clearly the distribution function F_m of the variable Y in the above definition satisfies:

$$F_m\big(\phi_m(\gamma)\big) = \Pr(Y \leq m) = \gamma = \exp\big(-\phi_m(\gamma)\big)\left[1 + \phi_m(\gamma) + \frac{\big(\phi_m(\gamma)\big)^2}{2!} + \cdots + \frac{\big(\phi_m(\gamma)\big)^m}{m!}\right]$$

For given m and γ, the value of $\phi_m(\gamma)$ is easily computed by solving the above nonlinear equation using a simple technique such as the Newton-Raphson method. We computed the values of $\phi_m(\gamma)$ for various values of γ between 0 and 1 and for values of m ranging from 0 through 500. These values may be found in Appendix A (Figure 17 to 20). Note that $\phi_m(\gamma)$ decreases with γ (for fixed m) and increases with m (for fixed γ). We separate our discussion into two separate cases, the first where we have no knowledge about the interface reliability, and the second where we might have prior experience with the interface that allows us to bound its reliability.

5.3.1 Case 1: No Prior Information Available On Interface Reliability

First, we consider the case where there is no knowledge about λ_I relative to the $\lambda_j$values. Let us define $\delta = \frac{\lambda_I}{\lambda_C}$ as a measure of the relative magnitudes of the failure rates of the interfaces and

the components. Note that δ is some unknown positive constant. In such a situation, the following proposition is valid.

Proposition 5.1: The constraints in the generic formulation as given by (5.2) and (5.3) are respectively equivalent to the following:

$$\left\{Maximum\ \{(1+\delta)t_S + t_C\}\lambda_C, subject\ to\ \{(1+\delta)\lambda_C \leq\ -\ln R_1\ , \lambda_C \geq 0\}\right\} \leq \phi_m(1-\alpha) \quad \textbf{5.4}$$

$$\left\{Minimum\ \{(1+\delta)t_S + t_C\}\lambda_C, subject\ to\ \{(1+\delta)\lambda_C \geq\ -\ln R_0\ , \lambda_C \geq 0\}\right\} \geq \phi_m(\beta) \quad \textbf{5.5}$$

Proof: Note that the number of system failures (X_S) observed over time t_S is a Poisson random variable with mean $t_S(\lambda_I + \lambda_C)$, and the number of component failures of component j (X_j) observed over time t_j is a Poisson random variable with mean $t_C\lambda_j$. Hence $X_S + \sum_j X_j$ is also a Poisson random variable with parameter $\Lambda = t_S(\lambda_I + \lambda_C) + t_C\lambda_C = (t_S(1+\delta) + t_C)\lambda_C$.

Since $R_S = \exp(-\lambda_I - \lambda_C)$ it follows that

$R_S \geq R_1 \Longrightarrow \lambda_I + \lambda_C = (1+\delta)\lambda_C \leq\ -\ln R_1$, and

$R_S \leq R_0 \Longrightarrow \lambda_I + \lambda_C = (1+\delta)\lambda_C \geq\ -\ln R_0$

Using the Definition 5.1 for $\phi_m(\gamma)$ and the above; the constraints (5.2) and (5.3) can therefore be rewritten respectively as follows:

$$F_m(\Lambda) \geq F_m\left(\phi_m(1-\alpha)\right) \text{ for } \{, \lambda_C | (1+\delta)\lambda_C \leq\ -\ln R_1\ ; \lambda_C \geq 0\} \quad \textbf{5.6}$$

$$F_m(\Lambda) \leq F_m\left(\phi_m(\beta)\right) \text{ for } \{, \lambda_C | (1+\delta)\lambda_C \geq\ -\ln R_0\ ; \lambda_C \geq 0\} \quad \textbf{5.7}$$

Since the Poisson distribution function $F_m(\Lambda)$ is strictly decreasing in Λ, constraints (5.6) and (5.7) can respectively be rewritten as:

$$\Lambda \leq \phi_m(1-\alpha) \text{ for } \{, \lambda_C | (1+\delta)\lambda_C \leq -\ln R_1 \,; \lambda_C \geq 0\}$$

5.8

$$\Lambda \geq \phi_m(\beta) \text{ for } \{, \lambda_C | (1+\delta)\lambda_C \geq -\ln R_0 \,; \lambda_C \geq 0\}$$

5.9

Note that (5.8) and (5.9) are defined for nonnegative values of t_C and t_S. It is clear that these constraints reduce to solving the following two linear programming subproblems in λ_I and λ_C

$$\underline{\text{Subproblem 1}}: \ Maximize\ \Lambda, \ \text{subject to } \{0 \leq \lambda_C \leq -\ln R_1 \,/(1+\delta)\}$$

5.10

$$\underline{\text{Subproblem 2}}: \ Minimize\ \Lambda, \ \text{subject to } \{\lambda_C \geq -\ln R_0/(1+\delta) \,; \lambda_C \geq 0\}$$

5.11

and requiring the optimal objective value for Subproblem 1 to be $\leq \phi_m(1-\alpha)$ and that of Subproblem 2 to be $\geq \phi_m(\beta)$. Hence the result. □

Proposition 5.1: The optimum solutions to the two subproblems defined by (5.10) and (5.11) are given by $(-\ln R_1)(t_S(1+\delta)+t_C)/(1+\delta)$ and $(-\ln R_0)(t_S(1+\delta)+t_C)/(1+\delta)$, respectively.

Proof: Recall that $\Lambda = (t_S(1+\delta)+t_C)\lambda_C$ and the coefficient for λ_C is strictly positive. For Subproblem 1, it is clear that the objective is maximized when λ_C attains its maximum allowed value of $(-\ln R_1)/(1+\delta)$. Similarly, for Subproblem 2, the objective is minimized when when λ_C attains its minimum allowed value of $(-\ln R_0)/(1+\delta)$. The result then follows.

□

Now let us define $A(m) = \phi_m(1-\alpha)/(-\ln R_1)$; $B(m) = \phi_m(\beta)/(-\ln R_0)$. Then the optimization problem reduces to

$$Minimize\ C = c_S t_S + \left(\sum_{j=1}^{n} c_j\right) t_C$$

$$\text{subject to} \qquad B(m) \le t_S + t_C/(1+\delta) \le A(m)$$

5.12

$$t_S, t_C \ge 0$$

This optimization problem is feasible for any m that satisfies $B(m) \le A(m)$, i.e., $\phi_m(\beta)/(-\ln R_0) \le \phi_m(1-\alpha)/(-\ln R_1)$, i.e., $(-\ln R_1)/(-\ln R_0) \le \phi_m(1-\alpha)/\phi_m(\beta)$. Since $0 < R_0 < R_1 < 1$, the LHS of the last inequality above is strictly less than 1. Moreover, it has been shown by Rajgopal et al. [1994] that as long as $\alpha + \beta < 1$, the value of the ratio in the RHS of the same inequality, namely $\phi_m(1-\alpha)/\phi_m(\beta)$, is strictly increasing in m and approaches 1 as m approaches ∞. Thus, the problem is feasible for all $m \ge m^*$ where m^* is defined as:

$$m^* = Inf\ \left\{m \middle| \frac{(-\ln R_1)}{(-\ln R_0)} \le \frac{\phi_m(1-\alpha)}{\phi_m(\beta)}\right\}$$

5.13

Since the ratio $\frac{A(m)}{B(m)} = \left(\frac{\phi_m(1-\alpha)}{\phi_m(\beta)}\right) / \left(\frac{(-\ln R_1)}{(-\ln R_0)}\right)$, we may equivalently restate the above as

$$m^* = Inf\ \left\{m \middle| \frac{A(m)}{B(m)} \ge 1\right\}$$

5.14

If we now restrict ourselves to feasible values of m with $B(m) \le A(m)$, it is clear that $B(m) = t_S + t_c/(1+\delta)$ at the optimum, because if $B(m) < t_S + t_c/(1+\delta)$ then we could reduce the value of t_S without violating feasibility and improve the objective. Thus, we must have $(1+\delta)B(m) = t_S(1+\delta) + t_C$ at the optimum. Note that $A(m)$ is not important as far as the optimization is concerned and is of relevance only in terms of defining feasible m. Now since the

optimization problem is a minimization, from the structure of objective function it is clear that in general, at the optimum either t_S or t_C must be equal to zero depending on the magnitude of their coefficients in the objective function. The optimum solution is given by

(a) $t_S = 0, t_C = (1+\delta)B(m)$, with $C = \left(\sum_{j=1}^{n} c_j\right)(1+\delta)B(m)$ if $c_S \geq \left(\sum_{j=1}^{n} c_j\right)(1+\delta)$

(b) $t_C = 0, t_S = B(m)$, with $C = c_S B(m)$ if $c_S \leq \left(\sum_{j=1}^{n} c_j\right)(1+\delta)$

Moreover, we know that $\phi_m(\beta)$ is strictly increasing in m, and therefore, so is $B(m)$. Hence the optimum value of m is the smallest one that leads to a feasible problem, which is m^* as defined in (5.13).

From the above analysis, one may conclude that for a series system with imperfect interfaces and no specific *a priori* knowledge on the relative magnitude of the interface failure rate, the optimal policy calls for either only system testing or only component testing. Such a decision depends on the value of δ, which is obviously unknown. Because of this, these results are of somewhat limited practical use. We may consider two special cases here. If δ = 0, which leads to perfect interfaces, then we get the commonsense result that only component testing is warranted as long as the total testing cost per unit of test time across components is less than system test cost per unit time. On the other hand, when interfaces are unreliable (i.e., $\delta > 0$) the optimum policy suggests only system testing when δ is relatively large. This is intuitively sensible because it indicates that if interfaces are highly unreliable the only way to draw good inferences is by testing the entire assembled system.

5.3.2 Case 2: Using Prior Information On Interface Reliability

We next consider the case where some prior information on λ_I is available. In particular, suppose that δ is defined as an upper bound on the ratio λ_I/λ_C (rather than the exact value of this ratio), and that a (positive) value for δ is *given*.

By using similar arguments to the ones seen in **Case 1** in the previous subsection, we can respectively, reduce constraints (5.2) and (5.3) to (5.15) and (5.16) below:

$$\left\{Argmax\ \{\Lambda|\lambda_I + \lambda_C \leq\ -\ln R_1\ ; \lambda_I \leq \delta\lambda_C; \lambda_I, \lambda_C \geq 0\}\right\} \leq \phi_m(1-\alpha) \quad \textbf{5.15}$$

$$\left\{Argmin\ \{\Lambda|\lambda_I + \lambda_C \geq\ -\ln R_0\ ; \lambda_I \leq \delta\lambda_C; \lambda_I, \lambda_C \geq 0\}\right\} \geq \phi_m(\beta) \quad \textbf{5.16}$$

where once again, $\Lambda = t_S(\lambda_I + \lambda_C) + t_C\lambda_C = t_S\lambda_I + (t_C + t_S)\lambda_C$

Let us consider the subproblem in the LHS of (5.15). At the optimum solution, the first constraint $\lambda_I + \lambda_C \leq\ -\ln R_1$ must be binding. This is true because if we had a nonnegative vector which satisfied both constraints $\lambda_I + \lambda_C \leq\ -\ln R_1\ ; \lambda_I \leq \delta\lambda_C$ and the first one is inactive, then we could increase λ_C by the amount of the slack and increase the value of Λ while continuing to satisfy the second constraint. Given that λ_C has a larger objective coefficient, the optimum solution to this subproblem will have $\lambda_C =\ -\ln R_1\ ;\ \lambda_I = 0$, with a corresponding value of the objective given by $\Lambda^* = (t_C + t_S)(-\ln R_1)$. The second constraint $\lambda_I \leq \delta\lambda_C$ is redundant at the optimum.

Now let us consider the subproblem in the LHS of (5.16). The first constraint of this problem $\lambda_I + \lambda_C \geq\ -\ln R_0$ must be binding. This is true because if we had a nonnegative vector which satisfied both constraints $\lambda_I + \lambda_C \geq\ -\ln R_0\ ; \lambda_I \leq \delta\lambda_C$ and the first one is inactive, then we could simply decrease λ_I by the amount of the excess and improve the objective while continuing to

satisfy the second constraint. Thus, at the optimum we must have $\lambda_I = (-\ln R_0) - \lambda_C$. The subproblem then reduces to:

$$Minimize\ (-\ln R_0)t_S + t_C\lambda_C,\ \text{subject to}\ \{\lambda_C \geq\ -\ln R_0/(1+\delta)\,;\lambda_C \geq 0\}$$

It is clear that the optimum solution to this subproblem is given by $\lambda_C = -\ln R_0/(1+\delta)$ and hence $\lambda_I = (-\ln R_0)\delta/(1+\delta)$. The corresponding objective function value is given by $\Lambda^* = [-\ln R_0/(1+\delta)]t_C + (-\ln R_0)t_S$.

We may now restate the optimization problem as follows:

$$Minimize\ C = c_S t_S + \left(\sum_{j=1}^{n} c_j\right) t_C$$

subject to $(t_C + t_S)(-\ln R_1) \leq \phi_m(1-\alpha) \implies (t_C + t_S) \leq A(m)$

$$[-\ln R_0/(1+\delta)]t_C + (-\ln R_0)t_S \geq \phi_m(\beta) \implies \frac{t_C}{1+\delta} + t_S \geq B(m)$$

$$t_S, t_C \geq 0$$

where $A(m)$ and $B(m)$ are as defined before in the proof of Proposition 5.1.

Note that $\frac{t_C}{1+\delta} < t_C$ and hence this optimization problem is only feasible if $B(m) \leq A(m)$, i.e. $\phi_m(\beta)/(-\ln R_0) \leq \phi_m(1-\alpha)/(-\ln R_1)$, which can be rewritten as $(-\ln R_1)/(-\ln R_0) \leq \phi_m(1-\alpha)/\phi_m(\beta)$. This is identical to the condition in Case 1. Hence we are guaranteed that the problem is feasible for all $m \geq m^*$ where m^* is defined via (5.13) or (5.14).

It is easy to see that at optimum, the constraint $\frac{t_C}{1+\delta} + t_S \geq B(m)$ must hold as an equality. Otherwise, we may decrease t_C to make the constraint hold as an equality and improve the objective, while continuing to satisfy the other constraint. Thus $t_C = (1+\delta)[B(m) - t_S]$.Since $t_C \geq 0$; it follows that $t_S \leq B(m)$. Using this information, the optimization problem can be restated as the following:

$$Minimize\ C = \left[c_S - (1+\delta)\left(\sum_{j=1}^{n} c_j\right)\right] t_S + (1+\delta)\left(\sum_{j=1}^{n} c_j\right) B(m)$$

subject to $t_S \geq \{(1+\delta)B(m) - A(m)\}/\delta$

$t_S \leq B(m)$

$t_S \geq 0$

Note that this problem is feasible for all m such that $\frac{A(m)}{B(m)} \geq 1$, i.e., for all $m \geq m^*$. We now exhaustively consider several different scenarios that we might encounter when we are solving the above optimization problem.

5.3.2.1 Scenario 1

$$\boldsymbol{c_S - (1+\delta)\left(\sum_{j=1}^{n} c_j\right) \leq 0}$$

Note that this expression is the coefficient of t_S in the objective. First, suppose we are given a feasible m. At the optimum the value of t_S will be at its upper bound which is determined by $t_S \leq B(m)$. Hence at the optimum, $\boldsymbol{t_S = B(m)}$ and $t_C = (1+\delta)[B(m) - t_S]$, i.e., $\boldsymbol{t_C = 0}$. This indicates that only the system is tested and the total optimal cost will be $\boldsymbol{C = c_S B(m)}$. Furthermore, since $B(m)$ is increasing in m the optimum value of m is given by its lowest feasible value which is $\boldsymbol{m = m^*}$ as defined via (5.13).

5.3.2.2 Scenario 2

$$\boldsymbol{c_S - (1+\delta)\left(\sum_{j=1}^{n} c_j\right) \geq 0} \text{ and } \boldsymbol{(1+\delta)B(m^*) - A(m^*) \leq 0}$$

Since its coefficient in the objective is nonnegative it is clear that given m, at the optimum, $t_S = Max\{0, [(1+\delta)B(m) - A(m)]/\delta\}$. Now, $(1+\delta)B(m^*) - A(m^*) \leq 0$ implies that $(1+\delta) \leq A(m^*)/B(m^*)$, and as discussed earlier, $A(m)/B(m)$ is strictly increasing in m. Thus, it is

clear that $(1+\delta) \leq A(m)/B(m)$ for all $m > m^*$, i.e., $[(1+\delta)B(m) - A(m)]/\delta \leq 0$ for all feasible values of m. This implies that for any feasible m, we must have $t_S = 0$ at the optimum and hence $t_C = (1+\delta)[B(m) - t_S] = (1+\delta)B(m)$. Moreover, the value of the objective is given by $(1+\delta)\left(\sum_{j=1}^{n} c_j\right)B(m)$ for all such m. But $B(m)$ increases with m and thus $\boldsymbol{m = m^*}$ being the smallest feasible m, it is also the optimum choice. So, in this scenario, only components are tested for $\boldsymbol{t_C = (1+\delta)B(m^*)}$ units of time. The optimal cost is given by $\boldsymbol{C = (1+\delta)\left(\sum_{j=1}^{n} c_j\right)B(m^*)}$.

5.3.2.3 Scenario 3

$\boldsymbol{c_S - (1+\delta)\left(\sum_{j=1}^{n} c_j\right) \geq 0}$ and $\boldsymbol{(1+\delta)B(m^*) - A(m^*) > 0}$

Note that here we have $1 \leq \frac{A(m^*)}{B(m^*)} < (1+\delta)$. Suppose again that we are given a feasible m. Once again because of its positive coefficient in the objective at the optimum $t_S = Max\{0, [(1+\delta)B(m) - A(m)]/\delta\}$. Since m is feasible it follows that $1 < \frac{A(m^*)}{B(m^*)} \leq \frac{A(m)}{B(m)}$.

However, there are two possibilities here: (a) $\frac{A(m)}{B(m)} < (1+\delta)$ and (b) $\frac{A(m)}{B(m)} \geq (1+\delta)$

We will consider each case separately.

Scenario 3a: Suppose $1 < \left(\frac{A(m)}{B(m)}\right) < (1+\delta)$, i.e., $(1+\delta)B(m) - A(m) > 0$. Therefore, at the optimum we will have $\boldsymbol{t_S = [(1+\delta)B(m) - A(m)]/\delta}$ and $t_C = (1+\delta)[B(m) - t_S]$, i. e., $\boldsymbol{t_C = \left(\frac{(1+\delta)}{\delta}\right)\{A(m) - B(m)\}}$. Thus, both the system and components are tested in this scenario. The optimum cost is given by:

$$\boldsymbol{C = (1+\delta)\left(\sum_{j=1}^{n} c_j\right)B(m) + \left[c_S - (1+\delta)\left(\sum_{j=1}^{n} c_j\right)\right]\frac{\{(1+\delta)B(m) - A(m)\}}{\delta}}$$

5.17

Scenario 3b: Suppose $\left(\frac{A(m)}{B(m)}\right) \geq (1+\delta)$. In this case $(1+\delta)B(m) - A(m) \leq 0$ and clearly, the situation reverts to that of Scenario 2, and we have the same solution as the one described under that scenario with $\boldsymbol{t_S = 0, t_C = (1+\delta)B(m)}$ and an objective value of $\boldsymbol{C = (1+\delta)\left(\sum_{j=1}^{n} c_j\right)B(m)}$.

Optimum value of m: The determination of the optimum m is more complicated with Scenario 3. First consider all m that satisfy the condition for Scenario 3b, and let us define m^1 as the smallest such m, i.e.,

$$m^1 = Inf\ \{m | m > m^*, \left(\frac{A(m)}{B(m)}\right) \geq (1+\delta)\}$$

5.18

Since the optimal objective value is $(1+\delta)\left(\sum_{j=1}^{n} c_j\right)B(m)$ and $B(m)$ is monotone increasing in m it is clear that the only candidate for the optimum m is m^1 with a corresponding objective value of $(1+\delta)\left(\sum_{j=1}^{n} c_j\right)B(m^1)$; all other values of m that satisfy the conditions for Scenario 3b yield higher values for the cost.

Next, consider all m that satisfy the conditions for Scenario 3a. Let us define the positive constants k_1 and k_2 via $k_1 = (1+\delta)\left(\sum_{j=1}^{n} c_j\right)$ and $k_2 = c_S - (1+\delta)\left(\sum_{j=1}^{n} c_j\right)$. Then we can rewrite (5.17) as $k_1\, B(m) + k_2\, \{(1+\delta)B(m) - A(m)\}/\delta = \left\{k_1 + k_2\left[(1+\delta) - \frac{A(m)}{B(m)}\right]\right\}B(m)$.

Recall that the value of $\frac{A(m)}{B(m)}$ increases monotonically so that $\left[(1+\delta)-\frac{A(m)}{B(m)}\right]$ is a positive quantity that decreases as m increases. Furthermore, $B(m)$ increases monotonically with m and with no bound from above. Hence the following Lemma holds:

Lemma 5.1: As long as $\left[(1+\delta)-\frac{A(m)}{B(m)}\right]>0$, the expression given by (5.17) will either increase monotonically with m, or it will initially decrease to some minimum value and then increase monotonically with m.

Thus, for this scenario we evaluate the expression in (5.17) at successively larger values of m starting with m^* and stop as soon as the value of this expression reaches a minimum at some optimum $m = m^0$ and starts to increase. The only exception is if we are at $m = m^1 - 1$ and the value of the expression is still decreasing, in which case the optimum value of m is given by $m^0 = m^1 - 1$ since this is the last value of m that meets the conditions for Scenario 3a. In our experience, when δ is relatively small, the very first value (m^*) will be the optimum one in most cases.

It is clear that the only candidates for the optimum solution are m^0 and m^1. However, it is not possible to make a general statement about when one will outperform the other because this depends on the relative magnitudes of δ, c_S and c_j. Thus, the objective value can be evaluated at m^0 using (6.17) with $m = m^0$ and at m^1 using $(1+\delta)\left(\sum_{j=1}^{n} c_j\right)B(m^1)$. The one yielding a smaller value for the objective is the optimum value for m. The two possibilities are shown in Figure 14. In Figure 14(a) the value of m^0 lies between m^* and m^1 and the optimum value is m^0. In Figure 14(b) $m^0 = m^*$ and the optimum value is m^1. One may note that if m^0 is optimum then we test both system and components according to the corresponding optimal test times. However, if m^1 is optimum then we test only the components for the corresponding optimal test time.

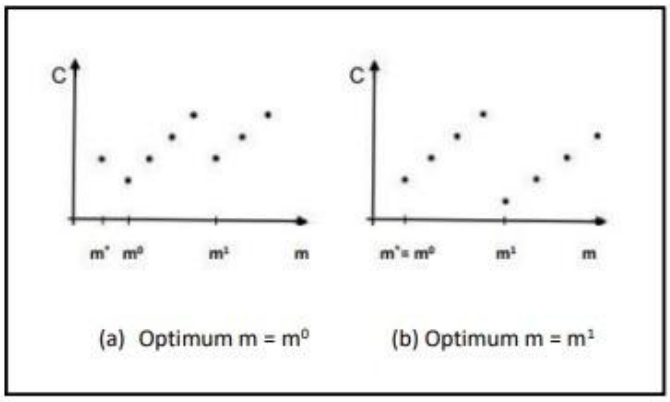

Figure 14: Minimum test costs corresponding to different values of *m*

5.3.3 Estimating Maximum Type 1 And Type 2 Error Probabilities

An issue that is of interest is the maximum Type 1 and Type 2 error probabilities that are associated with a test plan derived by the procedure above. This issue becomes particularly important in order to assess the degree of protection provided by sampling plans that are obtained when there is uncertainty in the value of δ. To compute these probabilities, first note the values of λ_I and λ_C that solve the subproblems are the ones that yield the maximum error probabilities. For Type 1 error these were given by $\{\lambda_C = -\ln R_1\,, \lambda_I = 0\}$ and for Type 2 error these were given by $\lambda_C = (-\ln R_0)/(1+\delta)\,, \lambda_I = (-\ln R_0)\delta/(1+\delta)$. Substituting these into the equation for Λ yields the following

$$Maximum\ Type\ 1\ Error\ Probability = 1 - F_m[(t_S + t_C)(-\ln R_1)] \quad \textbf{5.19}$$

$$Maximum\ Type\ 2\ Error\ Probability = F_m[\{t_S + \tfrac{t_C}{1+\delta}\}(-\ln R_0)] \quad \textbf{5.20}$$

These error probabilities hold for any plan with its corresponding values of t_S, t_C and m and thus (5.19) and (5.20) may be used to evaluate the performance of any such plan.

5.4 Example Problems

Three example problems and their solutions are presented to demonstrate all the different cases and scenarios that might encountered.

5.4.1 Example 1

In this example we fix the value of δ and see how the optimum policy varies as the system test cost changes relative to the component test costs. Here we consider a series system of five components to be tested with $R_0 = 0.80$ and $R_1 = 0.95$. The maximum acceptable probabilities of Type 1 and Type 2 errors are specified as $\alpha = \beta = 0.05$. We choose $\delta = 0.1$ and $\{c_j\} = \{10,15,5,5,2\}$ which gives $\sum_{j=1}^{5} c_j = 37$. For this data set, the value of the critical ratio for determining the smallest feasible m is given by $\frac{(-\ln R_1)}{(-\ln R_0)} = \frac{-ln\ 0.95}{-ln\ 0.80} = 0.23$. The smallest value of m for which $\frac{\phi_m(1-\alpha)}{\phi_m(\beta)}$ exceeds 0.23 is m =5 (with $\frac{\phi_5(0.95)}{\phi_5(0.05)} = \frac{2.613}{10.513} = 0.25$). Thus $m^* = 5$ with $A(m^*) = \frac{\phi_m(1-\alpha)}{-ln\ R_1} = \frac{2.613}{-ln\ 0.95} = 50.94$ and $B(m^*) = \frac{\phi_m(\beta)}{-ln\ R_0} = \frac{10.513}{-ln\ 0.80} = 47.11$.

(i) Suppose $\boldsymbol{c_S < (1+\delta)\left(\sum_{j=1}^{n} c_j\right)}$=40.7. This corresponds to Scenario 1 so that the optimum value of m is given by $m^* = 5$, $t_C = 0$, $t_S = B(5) = 47.11$ and total optimum test cost is $C = 47.11c_S$

(ii) Suppose $\boldsymbol{c_S > (1+\delta)\left(\sum_{j=1}^{n} c_j\right)}$=40.7. Here $\boldsymbol{(1+\delta)B(m^*) - A(m^*)} = (1.1)(47.11) - 50.94 > 0$. This corresponds to Scenario 3 of Case 2. Also $\frac{(-ln\ R_0)}{(-ln\ R_1)} = 4.35 > (1+\delta) = 1.1$. So, we need to find m^1 and m^0. Since $A(6) = \left(\frac{\phi_6(0.95)}{-ln\ 0.95}\right) =$

63.94, $B(6) = \left(\frac{\phi_6(0.05)}{-ln\ 0.80}\right) = 53.062$, we have $A(6)/B(6) = 1.205 > (1+\delta) = 1.1$. It follows that $m^1 = 6$ and the only candidate value for m^0 is 5. For $m^0 = 5$, we have $t_C = \{A(5) - B(5)\}\left(\frac{1+\delta}{\delta}\right) = 42.13$, $t_S = \{(1+\delta)B(5) - A(5)\}/\delta = 8.81$ and the total cost $C = c_S(8.81) + 37(42.13) = 8.81c_S + 1558.81$ units. For $m^1 = 6$, we have $t_C = (1+\delta)B(6) = 58.38$, $t_S = 0$ and the total cost $C = 37(58.38) = 2160.06$ units. Comparing the two values of C, it is clear that if $(40.7 <) c_S < 68.2$ then the optimum is defined by $m^0 = 5$, and if $c_S > 68.2$ then optimum solution is defined by $m^1 = 6$.

Similar results hold for other values of δ as well. It should be noted that for $\delta < 0.0813$ the value of $\{(1+\delta)B(5) - A(5)\}$ is negative and corresponds to Scenario 2, so that in this range, one would move from pure system tests to pure component tests directly, regardless of the value of c_S. The results are intuitively appealing. When c_S is relatively small, we test only the system. When c_S becomes larger, for intermediate values, both system and components are tested with total cost increasing as a piecewise linear function of c_S. When c_S becomes very large, only components are tested and system test cost is of little consequence. Figure 15 illustrates this for values of δ = 0.01, 0.1, 0.25 and 0.5.

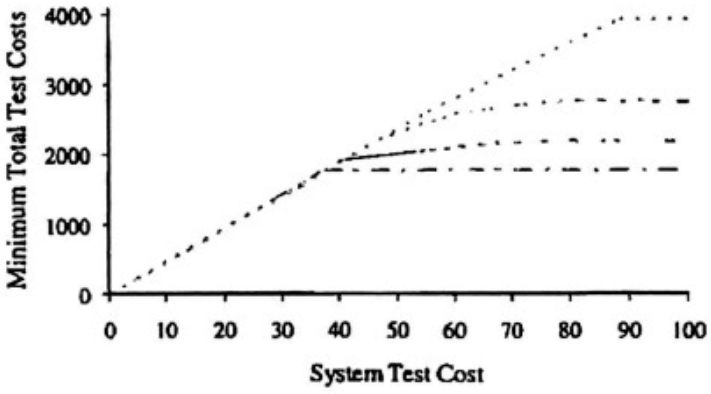

Figure 15: Minimum total test cost as a function of system test costs

5.4.2 Example 2

Here, we use the same data as the previous example with one change. We fix the value of c_S= 65 and study the effect of δ on the optimum cost. The rest of the parameters are maintained at the same values as before. We now study the three scenarios based on different values of δ.

(i) When $\delta > 0.7568$, it can be seen that $c_S - (1+\delta)\left(\sum_{j=1}^{n} c_j\right) \leq 0$. This corresponds to Scenario 1. The optimal solution is given by $m^* = 5$, $t_C = 0$ and $t_S = B(5) = 47.11$. The optimal cost is $C = 3062.15$ units.

(ii) When $\delta \leq 0.0813$, it can be seen that $c_S - (1+\delta)\left(\sum_{j=1}^{n} c_j\right) > 0$ and $(1+\delta)B(m^*) - A(m^*) \leq 0$. This corresponds to Scenario 2. The optimal solution is given by $m^* = 5$, $t_C = (1+\delta)B(m^*) = (1+\delta)47.11$ units and $t_S = 0$. The optimal cost is $C = 1743.07(1+\delta)$ units.

(iii) When $0{,}0813 \leq \delta \leq 0.7568$, it can be seen that $c_S - (1+\delta)\left(\sum_{j=1}^{n} c_j\right) > 0$ and $(1+\delta)B(m^*) - A(m^*) \geq 0$. This corresponds to Scenario 3. The optimal solution is

given by $m = m^0$ or $m = m^1$ depending upon the specific value of δ. That will dictate if it is economical to test both system and components or just components. The only way to know is to substitute the particular value of δ into the appropriate cost expressions and compare the two.

Figure 16 shows the minimum total test cost as a function of δ. It may be seen that there is an initial linear region for $\delta \leq 0.0813$, and a final flat region for $\delta \geq 0.7568$. In between, the optimum cost continuously increases with δ and the shape of the cost curve is irregular.

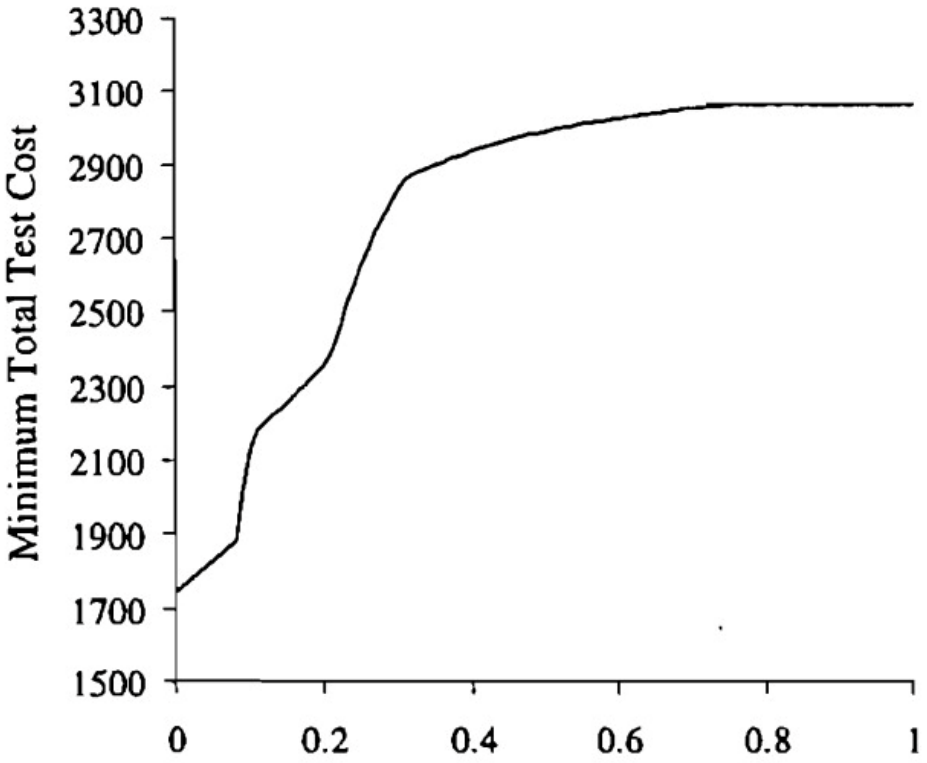

Figure 16: Minimum total test cost as a function of δ

5.4.3 Example 3

In this example, we illustrate the effect of uncertainty in the estimate for δ on the maximum Type 1 and Type 2 error probabilities associated with a plan. These probabilities depend on t_C, t_S and m. Here we consider the same data as in Example 2 above and let $\delta = 0.30$. It can be seen that this corresponds to Scenario 3 and the optimal solution is represented by $m = 6$, $t_C = 47.58$ and $t_S = 16.47$ with optimum total cost $C = 2831.01$. Using these we compute the maximum Type 1 and Type 2 errors to be equal to 0.05. Now, suppose that the value of δ is estimated wrongly and its true value is different and could vary in either direction of 0.3. We considered errors of 10%, 20%, 50%, 100% in either direction of 0.3, and compute solutions and Maximum Type 1 and Type 2 error probabilities for all these values of δ and tabulate the information in Table 10 below.

Table 10: Maximum Type 1 and Type 2 error probabilities for various δ

δ	t_C	t_S	m	*Max. Pr. Type 1 error*	*Max. Pr. Type 2 error*
0	47.11	0	5	0.115	0.005
0.15	61.03	0	6	0.05	0.027
0.24	56.73	7.32	6	0.05	0.040
0.27	51.65	12.40	6	0.05	0.045
0.30	47.58	16.47	6	0.05	0.05
0.33	15.44	35.50	5	0.115	0.025
0.36	14.47	36.47	5	0.115	0.028
0.45	12.34	38.60	5	0.115	0.038
0.60	10.21	40.73	5	0.115	0.056

It may be noted from Table 10 that as long as m does not change, Type 1 error remains unaffected because the maximum Type 1 error is not affected by δ. But when m changes, the Type

1 error probability increases. On the other hand, Type 2 error changes since it involves δ in its expression. However, the maximum Type 2 error is only slightly more than 0.05. If the δ value is unknown, then Type 1 and Type 2 error values may be computed for different values of δ to see how it affects the errors.

5.5 Some Comments

Let us look at the summary of results obtained in this chapter. When no prior information available ($\lambda_I = \delta\lambda_C$) on interface reliability the result is

(a) $t_S = 0$, $t_C = (1+\delta)B(m)$, $m = m^*$, $C = \left(\sum_{j=1}^{n} c_j\right)(1+\delta)B(m)$ if $c_S \geq \left(\sum_{j=1}^{n} c_j\right)(1+\delta)$

(b) $\boldsymbol{t_C = 0}$, $\boldsymbol{t_S = B(m)}$, $\boldsymbol{m = m^*}$, $\boldsymbol{C = c_S B(m)}$ if $\boldsymbol{c_S \leq \left(\sum_{j=1}^{n} c_j\right)(1+\delta)}$

When prior information is available in the form of $\lambda_I \leq \delta\lambda_C$

(c) $\boldsymbol{t_S = B(m)}$, $\boldsymbol{t_C = 0}$; $\boldsymbol{m = m^*}$ with $\boldsymbol{C = c_S B(m)}$ if $\boldsymbol{c_S \leq (1+\delta)\left(\sum_{j=1}^{n} c_j\right)}$

(d) $t_C = (1+\delta)B(m^*)$, $t_S = 0$; $m = m^*$, $C = (1+\delta)\left(\sum_{j=1}^{n} c_j\right)B(m^*)$. if $c_S \geq (1+\delta)\left(\sum_{j=1}^{n} c_j\right)$ and $(1+\delta)B(m^*) - A(m^*) \leq 0$

(e) $\boldsymbol{t_S = [(1+\delta)B(m) - A(m)]/\delta}$ and $\boldsymbol{t_C = \left(\frac{(1+\delta)}{\delta}\right)\{A(m) - B(m)\}}$, $\boldsymbol{m = m^0 \text{ or } m^1}$,

$\boldsymbol{C = (1+\delta)\left(\sum_{j=1}^{n} c_j\right)B(m) + \left[c_S - (1+\delta)\left(\sum_{j=1}^{n} c_j\right)\right]\frac{\{(1+\delta)B(m) - A(m)\}}{\delta}}$; if $\boldsymbol{c_S \geq (1+\delta)\left(\sum_{j=1}^{n} c_j\right)}$; $\boldsymbol{(1+\delta)B(m^*) > A(m^*)}$ and $\boldsymbol{(1+\delta)B(m) > A(m)}$

(f) $t_S = 0, t_C = (1+\delta)B(m)$, $m = m^0 \text{ or } m^1$, $C = (1+\delta)\left(\sum_{j=1}^{n} c_j\right)B(m)$; if $c_S \geq (1+\delta)\left(\sum_{j=1}^{n} c_j\right)$; $(1+\delta)B(m^*) > A(m^*)$ and $(1+\delta)B(m) < A(m)$

Suppose a decision maker decides to choose avoiding system testing by any means and testing only components irrespective of what the solution indicates, then he will have to play with certain parameters to see how his decision affects him. Notice that system testing is warranted only in three cases (b), (c) and (e) above. For example, if there is no priori information and $c_S \leq \left(\sum_{j=1}^{n} c_j\right)(1+\delta)$**;** the solution recommends system testing. In order to avoid the system testing, the decision maker may increase the estimate of cost for one unit of system testing (c_S) so that $c_S > \left(\sum_{j=1}^{n} c_j\right)(1+\delta)$. By doing this, the decision maker may avoid system testing. However, since there is no prior information on δ as well, decision maker will choose more expensive option just to avoid the system testing.

In the case of prior information being available as suggested in this chapter ($\lambda_I \leq \delta\lambda_C$), then just by increasing the estimate of cost for one unit of system testing (c_S) is not enough to ensure only component testing. Suppose, $c_S \leq \left(\sum_{j=1}^{n} c_j\right)(1+\delta)$ which warrants system testing as optimal solution. In order to avoid the system testing, decision maker may choose to increase c_S. But then the optimal solution may fall into either the case of (e) or (f). If it falls into (f) then no system testing is necessary. But if it falls into the case of (e) then system testing is still warranted. In order to avoid this, the decision maker has to look at the condition of $(1+\delta)B(m) > A(m)$. Since for a given set of values for α, β, R_0, R_1; the values of $A(m), B(m)$ are fixed irrespective of which system. The decision maker would not venture to change the values of α, β, R_0, R_1. The only parameter that can be changed to alter the condition $(1+\delta)B(m) > A(m)$ is by reducing the estimate of δ so that the condition changes to $(1+\delta)B(m) < A(m)$. In this case, the decision maker has to choose the assumption that system interfaces are very strong and the interface failure rate is much smaller. Here the decision maker runs the risk of ignoring interface failure at the final deployment if in fact interfaces have higher failure rate. Simply testing only components for much

longer period of time will not give any better inference on the system reliability when interfaces are weaker.

One important point that one should notice is that in all the above case (a) to (f); the value of $m^* = Inf\ \left\{m \middle| \frac{(-\ln R_1)}{(-\ln R_0)} \leq \frac{\phi_m(1-\alpha)}{\phi_m(\beta)}\right\}$ is constant for a given set of $\{\alpha, \beta, R_0, R_1\}$. m^* does not depend upon the number of components in a system. A system of ten components and a system of one million components will have same m^* as long as $\{\alpha, \beta, R_0, R_1\}$ are the same. This needs to be investigated further in future studies.

5.6 Conclusions

Test plans for demonstrating reliability either recommend system testing or component testing. This research makes the first effort to combine tests using system as well components for systems with unreliable interfaces. From a practitioner's perspective, this is necessary and an important step in the direction of optimal test plans. The results obtained in this work are intuitive and are easily applicable to simple series systems. This analysis clearly gives conditions where only the system is tested or where only the components are tested and also gives conditions where both system and components are tested. For a given value of δ (an upper bound on interface reliability), optimal test plans are very easily determined. For unknown values of δ, one can experiment with various values to estimate maximum Type 1 and Type 2 errors to get a feel for the decisions one has to make. It is also worth noting that in instances where it is optimal to do only component testing, the test times with imperfect interfaces exceed the test times for a system with perfect interfaces by a factor of $(1 + \delta)$. This is a consequence of the fact that perfect

interfaces ($(\delta = 0)$ represent a special case of the more general case with imperfect interfaces. Finally, if the *a priori* information on the interface failure rate is available in the form of an absolute upper bound, i.e., of the form $\lambda_I \leq u$, it is easy to show that if $u \geq \ln R_0$ the bound is too large to be of use and the optimum policy calls for pure system tests with $t_S = B(m^*)$ units of time, and if $u < \ln R_0$, this is exactly equivalent to solving the problem with a value of $\delta = u/\{u + (-lnR_0\}$.

In example 1, we assumed that $\delta = 0.1$. Suppose there were a total of four failures observed, two at the system level and two at the component level (with $t_S = 8.81, t_C = 42.13$). Since we have fewer than $m = 5$ failures we are expected to accept the system. However, one would be reluctant to accept the system because of the fact that there are as many system failures as there are component failures. If we estimate $\lambda_C = \frac{2}{42.13} = 0.0475$ and $\lambda_I + \lambda_C = \frac{2}{8.81} = 0.227$, then $\frac{\lambda_I}{\lambda_C} = 3.78$ which is much bigger than 0.1 for δ. Clearly this implies that the estimate of 0.1 for δ is far too optimistic and one would have to revise this upward suitably and solve the problem with new value of δ.

6.0 Conclusions

We address two different reliability related problems in this dissertation. The first one addresses reliability allocation when designing a system and the second addresses testing a new system before deployment.

6.1 Reliability Allocation Problem

We formulate the problem as a 0-1 integer programming problem, and develop new optimal as well as heuristic algorithms to solve the problem. In particular, we develop (i) an integer programming formulation and define appropriate valid inequalities for our formulation, and (ii) two heuristics: a nested simulated annealing algorithm and an evolutionary algorithm that uses a penalty function analogy. In the integer programming approach, the original formulations are all nonlinear integer programs except for simple series and simple parallel systems (which are integer linear programs). An iterative procedure is proposed where a single infeasible integer solution is eliminated at each iteration until the optimal solution is found. In the simulated annealing method, a nested approach is proposed where the outer algorithm addresses feasibility while the inner algorithm nudges the solution to optimality. In the evolutionary algorithm, a dynamic penalty approach is used to also penalize feasible solutions that might be far away from the optimum and to accelerate the search process so that convergence to an optimal solution is quicker. The integer programming approach guarantees an optimal solution, whereas, the other two solutions offer near

optimal solutions, but with much quicker computational times. In the following subsections, we provide suggestions to improve upon these procedures.

6.1.1 Conclusions And Future Research Directions: Integer Programming

The algorithm presented in Chapter 3 eliminates a single infeasible integer solution at each iteration. Hence any better optimum-seeking procedure for large systems, should be aimed at to either (i) eliminating more infeasible integer solutions at each iteration from the region within the relaxation that is infeasible for the original problem, or (ii) eliminating an entire section of this region rather than a single point. Ideally, this elimination procedure should involve additional linear inequalities that can be easily handled. In sections 3.7 and 3.8, we identify a class of additional disjunctive systems for ***SP*** and ***PS***. The next step would be to use these inequalities to develop better procedures.

For ***SP***, these disjunctive systems can yield additional valid inequalities which strengthen the linear relaxation by eliminating a significant portion of the region from the relaxation that is not of interest. Since adding additional constraints increases the portion of this region that is eliminated but also adds to computational effort, it remains to be studied how one can decide on the appropriate number of valid inequalities to be added at each iteration, and develop an appropriate algorithm for solving our problem. For ***PS***, the disjunctive systems do not yield a straightforward linearization and the feasible region is nonconvex, so that an approach such as the one suggested for ***SP*** is not possible. However, it might be possible to use the inequalities developed in Section 3.8 to develop a better branching scheme within a branch-and-bound procedure, and this is an avenue open for further research.

6.1.2 Conclusions And Future Research Directions: Metaheuristics

In Chapter 4, the first metaheuristic described is a nested simulated annealing algorithm that we developed. In this procedure, two simulated annealing procedures are used. The outer SA addresses feasibility and allows the algorithm to move to an infeasible point (unlike with traditional SA) while gradually making such a move more and more difficult as the algorithm progresses. The rationale here is that a "neighborhood" is not easy to define for our problem and determining whether a neighboring point is feasible can be computationally inefficient. On the other hand, the inner SA takes the current point provided by the outer algorithm and drives the search towards an optimal solution. This type of an approach has general appeal and can be used for any problem where generation of feasible solutions is difficult. The end results are satisfactory, but in our attempts, optimal solutions are not found using this approach. In future research, one could vary the algorithm parameters to study how the algorithm performs with these changes to determine better settings for these, or to develop some specific parameter functions that suit the problems at hand so that optimal solutions may be found more consistently in the search. Given the relatively low computational times, such an effort is worthwhile.

The second metaheuristic developed is an evolutionary algorithm that uses a dynamic penalty function. The penalty function is novel in that it is applied to infeasible solutions as well as feasible solutions that are considered as being "undesirable." The dynamic penalty function used in this research can also be used to address other optimization problems. Our work shows that this type of a dynamic penalty function accelerates the solution procedure and at least in the case of our problem, drives the search to the optimal solution at every attempt. However, the parameters of the search algorithm needed to be tweaked in order to achieve such results. In future

research, one might investigate the nature of the parameters used in the algorithm in order to arrive at more robust and problem independent parameter setting.

6.1.3 Reliability Decay Functions

Every component in a system comes with a reliability value at the beginning. But the reliability diminishes over time for any such component. Such diminishing reliability can be modeled using a Decay function. Every component in a system has a definite life span. Even if the component is highly reliable at the time of first use, its eventual decline is certain. Mechanical systems, Civil structures etc. all decline in terms of reliability over time. In most system design problems, many addressed the problem as reliability allocation problem where individual component reliabilities are determined at minimum cost while making sure that certain system reliability criterion is met, and in some cases, redundant components are added to increase system reliability and tried to determine the optimal redundancy at each component level. However, all these design problems are valid but none of these works addressed an important aspect which is the decaying of the component and/or system. It is only guaranteed that system works for a certain design period but how healthy system functions during that process is not considered in design stages itself. We can define a decay function to highlight the process of decaying of each component as well as the system.

6.1.3.1 Definitions

Let R_{io} is reliability of a component at the time of initial use. Over time its reliability is reducing and the relationship between initial reliability and the reliability after certain time 't' is given below:

$\boldsymbol{R_{io}}$: Starting Reliability

$\boldsymbol{R_{it}}$: Reliability at time '**t**'

$\boldsymbol{R_{it} = R_{io}.e^{-D_i(t)}}$

Where $\boldsymbol{D_i(t)}$ is a function of time '*t*' and defined as 'Decay Function'. It is a positive and increasing function in time. The boundary conditions are $D_i(0) = 0$, $D_i(\infty) = \infty$

The exponential form of the equation is valid and it can be shown easily so.

Suppose $R_{it} = R_{io}.f(t)$ where $f(t)$ is some arbitrary function of time *t*. Then without losing generality, $f(t)$ can be written as $e^{ln(f(t))}$. In that case, $\ln(f(t))$ can be equated to $-D_i(t)$. Since the reliability decreases overtime, assumption of $D_i(t)$ being an increasing function of time *t* is valid.

when $\boldsymbol{t = 0}$; $\boldsymbol{R_{it} = R_{io}}$; and when $\boldsymbol{t = \infty}$; $\boldsymbol{R_{i\infty} = 0}$; since $\boldsymbol{\lim_{t \to \infty} D_i(t) = \infty}$.

For a better understanding of Decay Function, let us observe the definition of reliability. According to the Department of Defense: "*Reliability is the probability that an item will perform a required function without failure under stated conditions for a stated period of time*."

Suppose t_d is design time. Then traditional reliability definition says that $R_{i0} = \Pr(T \geq t_d)$. Here we define the reliability at time *t* as $R_{it} = \Pr(T \geq t + t_d)$. Based on the earlier definition using decay function and combining this definition gives:

$$R_{it} = R_{io}.e^{-D_i(t)} \Longrightarrow D_i(t) = -\ln\left(\frac{R_{it}}{R_{i0}}\right) = -\ln\left(\frac{\Pr(T \geq t+t_d)}{\Pr(T \geq t_d)}\right) = -\ln\left(\frac{1-F(t+t_d)}{1-F(t_d)}\right)$$

Where *T* is the random variable representing life time of component *i* and $F(t)$ is probability distribution function of *T*.

For example, if lifetime of a component is exponentially distributed with a parameter λ and is represented by $F(t) = 1 - e^{-\lambda t}$. Then clearly $D_i(t) = -\ln\left(\frac{e^{-\lambda(t+t_d)}}{e^{-\lambda t_d}}\right) = \lambda t$.

6.1.3.2 Future Work Using Decay Functions

Most systems are designed with a specific period in consideration such as warranty period. However, users tend to use the products well beyond that warranty period. For example, an automobile comes with a warranty of 7 years or 100000 miles whichever is earlier. But most drivers like to use the automobile well beyond that warranty. Some automobiles begin to falter as soon as they cross their warranty period while some other automobiles function much better well beyond their warranty period. This can be ascertained at the design stage itself if their decay functions are available. Products with severe decay may incur heavy maintenance costs while products with milder decay incur much lesser maintenance costs. Hence maintenance costs may play major role in deciding the selection of the final system and not just initial cost at purchase time. The decay functions can be strategically used to estimate the maintenance costs. Research in this direction is ongoing and we will publish that research in the near future.

6.2 Optimal Test Plans

The second reliability problem we consider is testing prior to deployment. Optimal test plans are developed in this work for a simple series system with imperfect interfaces where both system level testing and component level testing are considered simultaneously. The results are very intuitive and easy to understand, and show that the best approach depends on the ratio $\lambda_I/\lambda_C(=\delta)$ of the failure rates of the interface and the components, and what we know about the value of δ. In the absence of any information on interface reliability, one could experiment with reasonable values for δ to see what the optimal plans dictate. One could be very liberal in the beginning by

assuming a very low value for δ and progressively correct that estimate by making observations from some initial testing.

Even though this research makes an important first step, more research is warranted with consideration of (i) other distributions for failure times of components and interfaces such as Weibull or Gamma, (ii) other test rules instead of the "sum" rule we adopted and, (iii) other forms of *a priori* information such as simple upper bounds on interface failures as well as bounds on individual component failure rates.

Appendix

m	β												
	0.001	0.01	0.025	0.05	0.1	0.2	0.5	0.8	0.9	0.95	0.975	0.99	0.999
0	6.90775	4.60517	3.68888	2.99572	2.30259	1.60944	0.69315	0.22314	0.10536	0.05129	0.02532	0.01005	0.001
1	9.23334	6.63835	5.57163	4.74386	3.88972	2.99431	1.67835	0.82439	0.53181	0.35536	0.24221	0.14855	0.0454
2	11.22885	8.40595	7.22468	6.29579	5.32232	4.27903	2.67406	1.53504	1.10207	0.81769	0.61868	0.43605	0.19053
3	13.06223	10.04512	8.76727	7.75364	6.68078	5.51505	3.67206	2.29679	1.74477	1.36632	1.08987	0.82326	0.42855
4	14.79414	11.60463	10.24159	9.15351	7.99359	6.72098	4.67091	3.08954	2.43259	1.97015	1.62349	1.27913	0.73939
5	16.45474	13.10848	11.66833	10.51303	9.27467	7.90599	5.67016	3.90366	3.1519	2.61302	2.20189	1.78528	1.10715
6	18.06163	14.57062	13.05947	11.84239	10.53207	9.07539	6.66963	4.73366	3.89477	3.28532	2.81436	2.33021	1.52043
7	19.62618	15.99996	14.42268	13.14811	11.77091	10.23254	7.66925	5.57606	4.65612	3.98083	3.45383	2.90611	1.97099
8	21.1562	17.40265	15.76319	14.43464	12.99471	11.37977	8.66895	6.42848	5.43247	4.69524	4.11537	3.50745	2.4527
9	22.65737	18.78312	17.0848	15.70521	14.20599	12.51875	9.66871	7.28922	6.2213	5.42542	4.79539	4.1302	2.96091
10	24.13397	20.14468	18.39036	16.96222	15.40664	13.65073	10.66852	8.15702	7.02075	6.16903	5.49116	4.77125	3.49148
11	25.5893	21.48991	19.68204	18.20751	16.59812	14.77666	11.66836	9.0309	7.82934	6.92421	6.20058	5.42818	4.04243
12	27.02429	22.82084	20.96158	19.44257	17.78159	15.8973	12.66823	9.9101	8.64594	7.68958	6.92195	6.09907	4.61106
13	28.44453	24.13912	22.2304	20.66857	18.95796	17.01328	13.66811	10.79398	9.46962	8.46394	7.65393	6.78235	5.19543
14	29.84999	25.44609	23.48962	21.88648	20.12801	18.12509	14.66802	11.68206	10.29962	9.24633	8.39539	7.47673	5.79397
15	31.24212	26.74289	24.74022	23.09713	21.29237	19.23316	15.66793	12.57389	11.1353	10.03596	9.14538	8.18111	6.40532
16	32.62217	28.03045	25.983	24.30118	22.45158	20.33782	16.66785	13.46914	11.97613	10.83214	9.90313	8.89457	7.02834
17	33.99118	29.30961	27.21865	25.49923	23.60609	21.4394	17.66779	14.36748	12.82165	11.6343	10.66794	9.61634	7.66205
18	35.35008	30.58104	28.44776	26.69177	24.75626	22.53814	18.66773	15.26867	13.67148	12.44195	11.43924	10.34572	8.30559
19	36.69965	31.84537	29.67085	27.87924	25.9025	23.63427	19.66767	16.17248	14.52526	13.25465	12.21652	11.08213	8.95821
20	38.04058	33.10312	30.88838	29.06202	27.04507	24.72799	20.66762	17.0787	15.38271	14.07203	12.99933	11.82505	9.61925
21	39.37349	34.35476	32.10073	30.24044	28.18424	25.81946	21.66758	17.98718	16.24356	14.89374	13.78729	12.57401	10.28814
22	40.69892	35.6007	33.30826	31.41481	29.32024	26.90885	22.66754	18.89774	17.10758	15.7195	14.58003	13.32862	10.96435
23	42.01734	36.84132	34.51129	32.58538	30.45328	27.99629	23.6675	19.81026	17.97457	16.54904	15.37726	14.0885	11.64743
24	43.32921	38.07695	35.7101	33.7524	31.58354	29.0819	24.66747	20.72461	18.84432	17.38213	16.17869	14.85334	12.33695
25	44.63489	39.30788	36.90493	34.91608	32.71118	30.16579	25.66744	21.64068	19.71669	18.21855	16.98407	15.62284	13.03253
26	45.93476	40.53439	38.09602	36.07661	33.83637	31.24806	26.66741	22.55837	20.59152	19.05811	17.79318	16.39672	13.73384
27	47.22912	41.75672	39.28358	37.23416	34.95923	32.32881	27.66738	23.47759	21.46867	19.90064	18.6058	17.17476	14.44057
28	48.51828	42.97483	40.4678	38.3889	36.0799	33.40811	28.66736	24.39827	22.34802	20.74598	19.42176	17.95673	15.15243
29	49.80249	44.18945	41.64884	39.54097	37.19848	34.48603	29.66733	25.32031	23.22944	21.59398	20.24088	18.74243	15.86917
30	51.08201	45.40051	42.82687	40.69051	38.31508	35.56266	30.66731	26.24366	24.11285	22.44451	21.063	19.53167	16.59053

Figure 17: Φ_m (β) Values – Part 1

	β												
m	0.001	0.01	0.025	0.05	0.1	0.2	0.5	0.8	0.9	0.95	0.975	0.99	0.999
31	52.35706	46.60818	44.00203	41.83763	39.4298	36.63804	31.66729	27.16825	24.99815	23.29745	21.88798	20.32428	17.31631
32	53.62785	47.81261	45.17445	42.98245	40.54272	37.71225	32.66727	28.09402	25.88523	24.15269	22.71569	21.12011	18.0463
33	54.89457	49.01396	46.34427	44.12508	41.65393	38.78533	33.66725	29.02093	26.77403	25.01012	23.546	21.91902	18.78032
34	56.1574	50.21235	47.51159	45.26561	42.7635	39.85732	34.66724	29.94891	27.66447	25.86964	24.37879	22.72086	19.51818
35	57.4165	51.40792	48.67653	46.40413	43.8715	40.92829	35.66722	30.87792	28.55648	26.73117	25.21397	23.52551	20.25974
36	58.67203	52.60078	49.83917	47.54073	44.978	41.99827	36.66721	31.80792	29.44998	27.59462	26.05143	24.33286	21.00482
37	59.92413	53.79104	50.99963	48.67548	46.08306	43.06731	37.66719	32.73887	30.34493	28.45991	26.89107	25.1428	21.75331
38	61.17294	54.9788	52.15797	49.80846	47.18674	44.13543	38.66718	33.67073	31.24126	29.32697	27.73282	25.95523	22.50505
39	62.41858	56.16416	53.31428	50.93974	48.28908	45.20267	39.66716	34.60347	32.13892	30.19574	28.5766	26.77004	23.25994
40	63.66118	57.34722	54.46865	52.06937	49.39014	46.26908	40.66715	35.53705	33.03787	31.06615	29.42232	27.58716	24.01784
41	64.90083	58.52804	55.62113	53.19742	50.48997	47.33467	41.66714	36.47145	33.93804	31.93813	30.26992	28.40649	24.77867
42	66.13765	59.70672	56.7718	54.32395	51.58861	48.39948	42.66713	37.40662	34.83941	32.81164	31.11933	29.22797	25.5423
43	67.37174	60.88333	57.92072	55.449	52.6861	49.46354	43.66712	38.34256	35.74192	33.68662	31.97048	30.05151	26.30865
44	68.60318	62.05793	59.06795	56.57263	53.78248	50.52686	44.66711	39.27922	36.64555	34.56302	32.82332	30.87704	27.07762
45	69.83207	63.2306	60.21354	57.69489	54.87779	51.58948	45.6671	40.21659	37.55024	35.44079	33.6778	31.70451	27.84913
46	71.05848	64.4014	61.35755	58.81582	55.97207	52.65142	46.66709	41.15464	38.45597	36.31988	34.53385	32.53384	28.6231
47	72.2825	65.57039	62.50004	59.93547	57.06533	53.7127	47.66708	42.09335	39.36271	37.20027	35.39143	33.36498	29.39944
48	73.5042	66.73761	63.64104	61.05387	58.15763	54.77334	48.66707	43.0327	40.27042	38.0819	36.25049	34.19786	30.17809
49	74.72365	67.90314	64.7806	62.17106	59.24898	55.83336	49.66706	43.97267	41.17907	38.96473	37.11098	35.03245	30.95897
50	75.94092	69.06702	65.91877	63.28707	60.33942	56.89277	50.66706	44.91325	42.08863	39.84874	37.97287	35.86869	31.74202
60	88.00629	80.62456	77.23189	74.38963	71.19885	67.4574	60.66699	54.34855	51.22923	48.74638	46.66019	44.31203	39.67929
70	99.90837	92.0586	88.44079	85.4046	81.98995	77.9771	70.66695	63.82879	60.43815	57.73156	55.45162	52.87862	47.77909
80	111.6807	103.3946	99.56692	96.35003	92.72682	88.46116	80.66691	73.34469	69.7013	66.78623	64.32573	61.54313	56.00819
90	123.3467	114.6502	110.6253	107.2385	103.4193	98.91603	90.66688	82.88979	79.00885	75.89783	73.26757	70.28788	64.34332
100	134.9234	125.8385	121.6268	118.0793	114.0745	109.3464	100.6669	92.45936	88.35363	85.05715	82.26621	79.09993	72.76746
110	146.4239	136.9691	132.5798	128.8792	124.698	119.756	110.6669	102.0498	97.73021	94.25724	91.31338	87.96951	81.26769
120	157.8583	148.0499	143.4906	139.6438	135.2938	130.1473	120.6668	111.6584	107.1344	103.4927	100.4027	96.88897	89.83396
130	169.2345	159.0867	154.3644	150.3774	145.8654	140.5228	130.6668	121.2829	116.5627	112.7592	109.5289	105.8523	98.45824
140	180.5591	170.0847	165.2055	161.0833	156.4156	150.8842	140.6668	130.9216	126.0126	122.0533	118.6879	114.8544	107.134
150	191.8376	181.0479	176.0172	171.7647	166.9465	161.2329	150.6668	140.5728	135.4817	131.372	127.8763	123.8914	115.8559
160	203.0745	191.9796	186.8024	182.4237	177.4601	171.5702	160.6668	150.2355	144.9681	140.7129	137.0912	132.9598	124.6194

Figure 18: Φm (β) Values – Part 2

m	β												
	0.001	0.01	0.025	0.05	0.1	0.2	0.5	0.8	0.9	0.95	0.975	0.99	0.999
170	214.2735	202.8828	197.5635	193.0626	187.958	181.8972	170.6668	159.9085	154.4702	150.0741	146.3301	142.0567	133.4207
180	225.4379	213.7599	208.3026	203.683	198.4414	192.2148	180.6668	169.5909	163.9868	159.4537	155.591	151.1797	142.2565
190	236.5706	224.6131	219.0216	214.2865	208.9117	202.5236	190.6668	179.2821	173.5165	168.8502	164.8721	160.3266	151.1241
200	247.674	235.4442	229.722	224.8744	219.3698	212.8245	200.6668	188.9812	183.0584	178.2624	174.1718	169.4955	160.0208
210	258.7504	246.2549	240.4051	235.4477	229.8166	223.118	210.6668	198.6877	192.6116	187.689	183.4886	178.6849	168.9447
220	269.8015	257.0466	251.0723	246.0077	240.2529	233.4045	220.6668	208.4012	202.1753	197.1291	192.8216	187.8933	177.8938
230	280.8291	267.8205	261.7244	256.5551	250.6794	243.6847	230.6668	218.121	211.7488	206.5817	202.1694	197.1194	186.8663
240	291.8347	278.5779	272.3626	267.0908	261.0968	253.9588	240.6668	227.8469	221.3314	216.046	211.5312	206.3621	195.8608
250	302.8197	289.3197	282.9877	277.6154	271.5057	264.2273	250.6668	237.5784	230.9226	225.5214	220.9062	215.6203	204.876
260	313.7852	300.0469	293.6004	288.1297	281.9064	274.4905	260.6667	247.3152	240.5218	235.0071	230.2935	224.8932	213.9106
270	324.7324	310.7602	304.2015	298.6342	292.2995	284.7487	270.6667	257.057	250.1287	244.5026	239.6924	234.1799	222.9635
280	335.6623	321.4605	314.7916	309.1295	302.6854	295.0022	280.6667	266.8035	259.7428	254.0073	249.1023	243.4796	232.0337
290	346.5758	332.1485	325.3713	319.616	313.0645	305.2512	290.6667	276.5545	269.3637	263.5208	258.5227	252.7917	241.1203
300	357.4737	342.8247	335.9411	330.0943	323.4372	315.4959	300.6667	286.3098	278.991	273.0426	267.9529	262.1155	250.2225
310	368.3569	353.4898	346.5015	340.5646	333.8037	325.7366	310.6667	296.069	288.6245	282.5722	277.3925	271.4505	259.3394
320	379.2259	364.1442	357.053	351.0274	344.1643	335.9735	320.6667	305.8322	298.2639	292.1094	286.841	280.796	268.4705
330	390.0816	374.7886	367.5959	361.4831	354.5194	346.2067	330.6667	315.599	307.9088	301.6537	296.2981	290.1517	277.6149
340	400.9244	385.4232	378.1307	371.932	364.8691	356.4364	340.6667	325.3693	317.5591	311.2049	305.7634	299.5171	286.7722
350	411.7549	396.0487	388.6577	382.3743	375.2138	366.6627	350.6667	335.1429	327.2144	320.7626	315.2364	308.8917	295.9417
360	422.5737	406.6653	399.1772	392.8103	385.5535	376.8859	360.6667	344.9198	336.8747	330.3265	324.7169	318.2751	305.123
370	433.3812	417.2733	409.6896	403.2404	395.8886	387.1059	370.6667	354.6997	346.5396	339.8965	334.2045	327.667	314.3155
380	444.1779	427.8733	420.1951	413.6646	406.2192	397.3231	380.6667	364.4826	356.209	349.4722	343.699	337.0671	323.5188
390	454.9643	438.4654	430.6941	424.0834	416.5455	407.5374	390.6667	374.2683	365.8827	359.0535	353.2001	346.475	332.7325
400	465.7406	449.05	441.1867	434.4968	426.8676	417.7489	400.6667	384.0567	375.5606	368.6401	362.7075	355.8904	341.9563
410	476.5074	459.6274	451.6731	444.9051	437.1858	427.9579	410.6667	393.8478	385.2425	378.2318	372.221	365.3131	351.1896
420	487.2648	470.1978	462.1537	455.3085	447.5	438.1643	420.6667	403.6414	394.9282	387.8284	381.7405	374.7427	360.4322
430	498.0133	480.7614	472.6286	465.707	457.8106	448.3682	430.6667	413.4374	404.6176	397.4299	391.2656	384.1791	369.6837
440	508.7532	491.3185	483.0981	476.101	468.1176	458.5698	440.6667	423.2358	414.3106	407.0359	400.7961	393.622	378.9439
450	519.4848	501.8694	493.5622	486.4906	478.4211	468.7692	450.6667	433.0365	424.0071	416.6463	410.332	403.0711	388.2123
460	530.2082	512.4141	504.0212	496.8758	488.7213	478.9663	460.6667	442.8393	433.7069	426.2611	419.873	412.5264	397.4889
470	540.9239	522.953	514.4753	507.2569	499.0182	489.1613	470.6667	452.6443	443.41	435.88	429.419	421.9875	406.7733
480	551.632	533.4862	524.9245	517.634	509.312	499.3543	480.6667	462.4514	453.1162	445.5029	438.9697	431.4543	416.0652
490	562.3328	544.0139	535.3692	528.0071	519.6028	509.5452	490.6667	472.2604	462.8255	455.1298	448.5251	440.9266	425.3644
500	573.0265	554.5363	545.8093	538.3765	529.8906	519.7343	500.6667	482.0714	472.5376	464.7604	458.085	450.4043	434.6708

Figure 19: Φm (β) Values – Part 3

Bibliography

1. Agarwal, K.K., Gupta, J.S., and Misra, K.B., A New heuristic criterion for solving a redundancy optimization problem, IEEE Transactions on Reliability, v R-24, Apr. 1975, pp 86-87.
2. Alice Yalaoui, Chengbin Chu, Eric Chaˆtelet, Reliability allocation problem in a series–parallel system, Elsevier, Reliability Engineering and System Safety 90 (2005) 55–61
3. Altinel, I.K. (1992), "The Design of Optimum Component Test Plans in the Demonstration of a Series System Reliability," *Computational Statistics and Data Analysis*, **14**, 281-292.
4. Altinel, I.K. (1994), "The Design of Optimum Component Test Plans in the Demonstration of System Reliability," *European Journal of Operations Research*, **78**, 97-115.
5. Altinel, I.K. and Ozekici, S. (1997), "A Dynamic Model for Component Testing," *Naval Research Logistics*, **44**, 187-197.
6. Altinel, I.K. and Ozekici, S. (1998), "Optimum Component Test Plans for Systems with Dependent Components," *European Journal of Operations Research*, **111**, 175-186.
7. Back, T., Hoffmeister, F., and Shwefel, H.P., A Survey of Evolution Strategies, Proceedings of 4th International Conference on Genetic Algorithms, 1991
8. Balas, E. (1979). Disjunctive programming. *Annals of discrete mathematics*, *5*, 3-51.
9. Bean, J.C., and Hadj-Alouane, A.B., A Dual Genetic Algorithm for Bounded Integer Programs, Technical Report 92-53, University of Michigan
10. Brusco, M.J. and Jacobs, L.W., A Simulated Annealing Approach to the Cyclic Staff-Scheduling Problem, Naval Research Logistics, v 40, 1993, pp 69-84
11. Bulfin, R.L., and Liu, C.Y., Optimal Allocation of Redundant Components for Large Systems, IEEE Transactions on Reliability, 34, 1985, 241-247.

12. Cateanu, V.M., Popentiu, F., and Gheorgiu, M, A Reliability Optimization Computer Algorithm for Complex Systems, Microelectronics Reliability, v 26, no. 6, 1986, pp 1019-1023.

13. A. O. Charles Elegbede, Chengbin Chu, Member, IEEE, Kondo H. Adjallah, and Farouk Yalaoui, Reliability Allocation Through Cost Minimization, IEEE TRANSACTIONS ON RELIABILITY, VOL. 52, NO. 1, MARCH 2003, 106-111

14. Cheung, R. C., (1980), "A User-Oriented Software Reliability Model," *IEEE Transactions on Software Engineering*, **SE-6**, 118-125.

15. Coit, D. and Smith, A.E., Reliability Optimization of Series-Parallel Systems Using a Genetic Algorithm, IEEE Transactions on Reliability, 45, 1996, 254-260.

16. Coit, D. and Smith, A.E., and David M. Tate, Adaptive Penalty Functions For Constrained Combinatorial Problems, INFORMS Journal on Computing, v 8, No 2, Spring 1996.

17. Cruz J. A., Applicability and Limitations of Reliability Allocation Methods,Technical Report, National Aeronautics and Space Administration, Glen Research Center, Cleveland, Ohio, November 2016, [This report is available in electronic form at http://www.sti.nasa.gov/ and http://ntrs.nasa.gov/]

18. Easterling, R.G., Mazumdar, M., Spencer, F.W., and Diegert, K.V. (1991), "System Based Component Test Plans and Operating Characteristics: Binomial Data", *Technometrics*, **33**, 287-298.

19. Eglese, R.W., Simulated Annealing: A tool for Operational Research, European Journal of Operational Research, v-46, 1990, 271-281

20. El-Neweihi, E,. Proschan, F. and Setheraman, J, Optimal Allocation of Components in Parallel-Series and Series-Parallel Systems, Journal of Applied Probability, 23, 1986, 770-777.

21. El-Neweihi, E,. Proschan, F. and Setheraman, J, Optimal Assembly of Systems Using Schur Functions and Majorization, Naval Research Logistics, 34, 1987, 705-712

22. El-Neweihi, E,. Proschan, F. and Setheraman, J, Optimal Allocation of Multistate Components, Handbook of Statistics, 7, 1988, 427-432

23. Fyffe, D.E., Hines, W.W and Lee, N.K., System Reliability Allocation and a Computational Algorithm, IEEE Transactions on Reliability, Vol. R-17, No 2, 1968, 64-69

24. Gal, S. (1974), "Optimal Test Design for Reliability Demonstration," *Operations Research*, **22**, 1236-1242.

25. Glover, F., Tabu Search, ORSA Journal on Computing, v 1, 1989, pp 190-206

26. Glover, F., Tabu Search – Part II, ORSA Journal on Computing, v2, 1989, pp 4-32

27. Hu B., Xie K., and Tai H., Optimal Reliability Allocation of ±800 kV Ultra HVDC Transmission Systems, IEEE TRANSACTIONS ON POWER DELIVERY, VOL. 33, NO. 3, JUNE 2018

28. Inagaki, T., Inoue, K. and Akashi, H., Interactive optimization of system reliability under multiple objectives, IEEE Transactions on Reliability, v-27, 1978, 264-267

29. Jacobson, D.W. and Arora S.R., Simultaneous Allocation of Reliability and Redundancy Using Simplex Search, Proceedings of Annual Reliability and Maintainability Symposium, IEEE, 1996, 243-250

30. Jin, T. and Coit, D. (1999), "Allocation of Test Units to Minimize System Reliability Estimation Variability," Rutgers University Industrial Engineering Department, Working Paper 99-122.
31. Kanagaraj, G. and Jawahar, N., A Simulated annealing algorithm for optimal supplier selection using the reliability-based total cost of ownership model, HomeInternational Journal of Procurement Management, Vol 2, No 3, May 2009, 244-266
32. Kattele, J.D., Least-Cost allocation of reliability investment, Operations Research, vol. 10, 1967, 249-265
33. Kececioglu, D., Reliability Engineering Handbook, 1991, Prentice Hall, Engelwood Cliffs, NJ
34. Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P., Optimization by Simulated annealing, Science, v-220, 1983, 671-680
35. Kontoleon, J.M., Optimal Link Allocation of Fixed Topology Networks, IEEE Transactions on Reliability, 28, 1979, 145-147
36. Leemis, L.M., Reliability: Probabilistic Models and Statistical Methods, 1995, Prentice-Hall, Engelwood Cliffs, New Jersey
37. Majety, Subba Rao V., Milind Dawande, and Jayant Rajgopal. "Optimal reliability allocation with discrete cost-reliability data for components." *Operations Research* 47, no. 6, 1999, 899-906.
38. Majety, Subba Rao V. and Rajgopal, J., Dynamic Penalty Function for Evolutionary Algorithms with an application to reliability allocation, Proceedings of the Sixth International Industrial Engineering Research Conference, pp. 36-41. 1997.
39. Majety, Subba Rao V., Srikanth Venkatasubramanian, and Alice E. Smith., Optimal reliability allocation in series parallel systems from component's discrete cost–reliability

data sets: a nested simulated annealing approach, Proceedings of the Fifth International Industrial Engineering Research Conference, pp. 435-440. 1996.

40. Malaiya, Yashwant K., "Reliability allocation", *Encyclopedia of Statistics in Quality and Reliability* 4 (2008).
41. Malon, D., Optimal Consecutive-2-out-of-n:F Component Sequencing, IEEE Transactions on Reliability, 33, 1984, 414-418
42. Mann, N.R., Schafer, R.E., and Singpurwalla, N.D. (1974), *Methods for Statistical Analysis and Life Data*, New York: John Wiley.
43. Mazumdar, M. (1977), "An Optimum Procedure for Component Testing in the Demonstration of Series System Reliability", *IEEE Transactions on Reliability*, **R-26**, 342-345.
44. Mazumdar, M. (1980), "An Optimum Component Testing Procedure for a Series System with Redundant Subsystems", *Technometrics*, **22**, 23-27.
45. Mazumdar, M. and Rajgopal, J. (2000) "Minimum Cost Test Plans for a Series System with Imperfect Interfaces", in *Perspectives in Statistical Science*, (Basu, A.K., Ghosh, J.K., Sen, P.K. and Sinha, B.K., Eds.), Oxford University Press, New Delhi.
46. MIL-HDBK-781 (1987), Washington: Department of the Navy, Space and Naval Warfare Systems Command, Washington DC 20363, July 14, 1987.
47. Nakagawa, Y., and Nakashima, K., A Heuristics Method for Determining Reliability Allocation, IEEE Transactions on Reliability, v R-26, Aug 1977, pp 156-161
48. Poore, J. H., Mills, H. D., and Mutchler, D., (1993), "Planning and Certifying Software Systems Reliability," *IEEE Software*, 88-99.
49. Ppastavridis, S.G. and Sfakianakis, M., Optimal-Arrangement and Importance of the Components in a consecutive-k-out-of-n:F System, IEEE Transactions on Reliability, 40, 1991, 277-279

50. Raghavachari, M. (1998), “A Note on Optimal Component Test Plans for Series System Reliability With Exponential Failure Times,” *Technometrics*, **40**, 345-347.

51. Rajgopal, J., and Mazumdar, M. (1988), "A Type-II Censored, Log Test-Time Based Component Testing Procedure for a Parallel System", *IEEE Transactions on Reliability*, **37**, 406-412.

52. Rajgopal, J., Mazumdar, M. and Savits T. (1994), "Some Properties of the Poisson Distribution with an Application to Reliability Testing," *Probability in the Engineering and Informational Sciences*, **8**, 345-354.

53. Rajgopal, J., and Mazumdar, M., (1995), "Designing Component Test Plans for Series System Reliability via Mathematical Programming", *Technometrics*, **37**, 195-212.

54. Rajgopal, J., and Mazumdar, M., (1996) “A System Based Component Test Plan for a Series System, with Type-II Censoring,” *IEEE Transactions on Reliability*, **45**(3), 375-378.

55. Rajgopal, J., and Mazumdar, M., (1997) "System Based Component Test Plans for Reliability Inferences," Frontiers in Reliability, (Basu et al., Eds.), World Scientific Press, Singapore, pp. 295-302, 1998.

56. Rajgopal, J., and Mazumdar, M., (1997) “Minimum Cost Component Test Plans for Demonstrating Reliability of a Parallel System”, *Naval Research Logistics*, **44**, 401-418, 1997.

57. Rajgopal, J., Mazumdar, M., and Majety, S.V. (1999) "Optimum Combined Test Plans for Systems and Components," *IIE Transactions*, **31**(**6**), 481-490.

58. Rajgopal, J. and Mazumdar, M. (2001), “System-based component test plans for reliability demonstration: A review and survey of the state-of-the-art”, Handbook of Statistics, Volume 20, Chapter 25, 659-677

59. Rajgopal, Jayant, Mainak Mazumdar, Subba Rao Majety, and Vikram Talada. "Modular Test Plans for Certification of Software Reliability." *IEEE Trans. Software Eng.* (2001).

60. Saad Abbas Abed, Hatem Kareem Sulaiman, and Zahir Abdul Haddi Hassan, Reliability Allocation and Optimization for (ROSS) of a Spacecraft by using Genetic Algorithm, IOP Conf. Series: Journal of Physics: Conf. Series 1294 (2019) 032034

61. Sankar, S. and Vellaisamy, P. (2000), "Two-Stage Component Test Plans for Testing the Reliability of a Series System," Department of Mathematics, Indian Institute of Technology, Powai, Mumbai 400076, India.

62. Sharma, J., and Venkateswaran, K.V., A direct method for maximizing the system reliability, IEEE Transactions on Reliability, v R-20, Nov 1971, pp 256-259

63. Shubin Si, Mingli Liu, Zhongyu Jiang, Tongdan Jin, and Zhiqiang Cai , Member, IEEE System Reliability Allocation and Optimization Based on Generalized Birnbaum Importance Measure, IEEE Transactions on Reliability, Vol. 68, No. 3, September 2019, 831-843

64. Tillman, F.A., and Littschwager, J.M., Integer Programming formulation of constrained reliability problems, Management Science, v 13, July 1967, pp 887-899

65. Tillman, F.A., Hwang, C.L., Fan, L.T., and Balbale, S.A., Systems reliability subject to multiple nonlinear constraints, IEEE Transactions on Reliability, v R-17, Sept 1968, pp 153-157

66. Tillman, F.A., Hwang, C.L., and Kuo, W., Determining component reliability and redundancy for Optimal System Reliability, IEEE Transactions on Reliability, v R-26, Aug 1977, pp 162-165

67. Tillman, F.A., Hwang, C.L., and Kuo, W., Optimization of Systems Reliability, 1980, Marcel Dekker Inc., New York, NY

68. Way Kuo, and Rui Wan, Recent Advances in Optimal Reliability Allocation, IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, vol. 37, no. 2, march 2007, 143-156

69. Woodhouse, C.F., Optimal Redundancy Allocation by Dynamic Programming, IEEE Transactions on Reliability, Vol. R-21, No. 1, 1972, 60-62

70. Yan, J.H., and Mazumdar, M. (1986), "A Comparison of Several Component Testing Plans for a Series System", *IEEE Transactions on Reliability*, **R-35**, 437-443.

71. Yan, J.H., and Mazumdar, M. (1987b), "A Component Testing Plan for a Parallel System with Type II Censoring," *IEEE Transactions on Reliability*, **R-36**, 425-428.

72. Yan, J.H., and Mazumdar, M. (1987a), "A Comparison of Several Component Testing Plans for a Parallel System," *IEEE Transactions on Reliability*, **R-36**, 419-424.

73. Zuo, M.J., and Shen, J., System Reliability Enhancement Through Heuristic Design, OMAE-Volume II, Safety and Reliability, ASME, 1992, 301-304

74. Sang Hwa Jin, Yeong-Koo Yeo, Il Moon, Yonsoo Chung, and In-Won Kim, Equipment selection for the optimal system unavailability of jacketed reactors with discrete cost data, Elsevier-Journal of Loss Prevention in the Process Industries 16 (2003) 443–448

75. Benedict K. Nmah, Shorter Searches for Least-Cost Allocations of Redundancy, AMERICAN JOURNAL OF SCIENTIFIC AND INDUSTRIAL RESEARCH, 2015, 6(6): 116-122

76. Michel José Anzanello , A simplified approach for reliability evaluation and component allocation in three-state series and parallel systems composed of non-identical components, Gest. Prod., São Carlos, v. 16, n. 1, p. 54-62, jan.-mar. 2009

77. Harish D. Goel, Margot P. C. Weijnen, and Johan Grievink, Optimal Reliable Retrofit Design of Multiproduct Batch Plants, Ind. Eng. Chem. Res. 2004, 43, 3799-3811

78. Y. P. Aneja, R. Chandrasekaran, and K. P. K. Nair, Minimal-Cost System Reliability With Discrete-Choice Sets for Components, IEEE TRANSACTIONS ON RELIABILITY, VOL. 53, NO. 1, MARCH 2004, pp 71-76

79. Ahmad A. Moreb, ALLOCATING REPAIRABLE SYSTEM'S RELIABILITY SUBJECT TO MINIMAL TOTAL COST - AN INTEGER PROGRAMMING APPROACH, JOURNAL OF SYSTEMS SCIENCE AND SYSTEMS ENGINEERING (Dec 2007) 16(4): 499-506

80. B. Kaushik, N. Kaur, A.K. Kohli, Achieving maximum reliability in fault tolerant network design for variable networks, Elsevier Applied Soft Computing 13 (2013) 3211–3224

81. Baijnath Kaushik, Navdeep Kaur, and Amit Kumar Kohli, Improved Approach for Maximizing Reliability in Fault Tolerant Networks, Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol.17 No.1, 2013, pp 27-41

82. Wei-Chang Yeh, Chien-Hsing Lin, and Yi-Cheng Lin, A MCS Based Neural Network Approach to Extract Network Approximate Reliability Function, Conference Paper in Communications in Computer and Information Science, AsiaSim 2007, CCIS 5, pp. 287–297, 2007

83. Anuj Kumar, Sangeeta Pant and Mangey Ram, System Reliability Optimization Using Gray Wolf Optimizer Algorithm, Qual. Reliab. Engng. Int., 2017, 33, pp 1327–1335

84. Ganga Negi, Anuj Kumar, Sangeeta Pant and Mangey Ram, OPTIMIZATION OF COMPLEX SYSTEM RELIABILITY USING HYBRID GREY WOLF OPTIMIZER, Decision Making: Applications in Management and Engineering Vol. 4, Issue 2, 2021, pp. 241-256

85. Wei-Chang Yeh, Yi-Cheng Lin, Yuk Ying Chung, and Mingchang Chih, A Particle Swarm Optimization Approach Based on Monte Carlo Simulation for Solving the Complex Network Reliability Problem, IEEE TRANSACTIONS ON RELIABILITY, VOL. 59, NO. 1, MARCH 2010, pp 212-221

86. Sangeeta Pant, Anuj Kumar and Mangey Ram, Reliability Optimization: A Particle Swarm Approach, Advances in Reliability and System Engineering, Springer, Mangey Ram J. Paulo Davim Editors, Management and Industrial Engineering ISBN 978-3-319-48874-5, 2017, pp 163-187

87. W. C. Yeh, An interactive augmented max-min MCS–RSM method for the multi-objective network reliability problem, International Journal of Systems Science Vol. 38, No. 2, February 2007, 87–99

88. P. K. Kapur, A. K. Bardhan and P. C. Jha, Optimal Reliability Allocation Problem for a Modular Software System, OPSEARCH, Vol. 40, No. 2, 2003, pp 138-148

89. Kevin Y.K. NG and NGF Sancho, A Hybrid ‘dynamic programming/depth first search’ algorithm, with an application to redundancy allocation, IIE Transactions, 2001, VOL 33, pp 1047-1058

90. Kang, Keebom; Doerr, Kenneth H.; Apte, Uday; Boudreau, Michael, Decision Support Models for Valuing Improvements in Component Reliability and Maintenance, Military Operations Research, Vol. 15, No. 4, 2010, pp 55-68

91. A. Salmasnia; E. Ameri , A. Ghorbanian; and H. Mokhtari , A multi-objective multi-state degraded system to optimize maintenance/repair costs and system availability, Scientia Iranica, Transactions E: Industrial Engineering 24 (2017) 355-363

92. U Dinesh Kumar, J E Ramı´rez-Ma´rquez, D Nowicki, and D Verma, Reliability and maintainability allocation to minimize total cost of ownership in a series-parallel system, Proc. IMechE Vol. 221 Part O: J. Risk and Reliability, 2007, pp 133-140

93. Jing Wang &Mian Li, Redundancy Allocation for Reliability Design of Engineering Systems With Failure Interactions, Transactions of the ASME, Journal of Mechanical Design, MARCH 2015, Vol. 137, pp 031403-1 to 031403-8

94. G. Kanagaraj and N. Jawahar, A Simulated annealing algorithm for optimal supplier selection using the reliability-based total cost of ownership model, International Journal of Procurement Management, Vol. 2, No. 3, 2009, pp 244-266

95. Rashika Gupta and Manju Agarwal, Penalty guided genetic search for redundancy optimization in multi-state series-parallel power system, Journal of Combinatorial Optimization, 2006, vol 12, pp 257-277

96. M.H. Heydari, K.M.Sullivan and E.A.Pohl, Optimal Allocation of Testing Resources in Reliability Growth, Proceedings of 2014 Industrial and Systems Engineering Research Conference