MODELS AND ALGORITHMS FOR RELIABLE FACILITY LOCATION PROBLEMS AND SYSTEM RELIABILITY OPTIMIZATION

By

ROGER LEZHOU ZHAN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2007

To my parents,

my wife, Ngana,

and my brother, Leping,

for their love and support

ACKNOWLEDGMENTS

I would never be able to adequately thank Dr. Zuo-Jun Max Shen, my supervisor and mentor, for helping me in developing my research topics. He allowed me to work freely, but he was always there when I needed advice or guidance. I want to thank him not only for his tremendous guidance and encouragement throughout my study, but also his endless trust and understanding.

Dr. Panos Pardalos deserves many sincere thanks. He recruited me to the University of Florida and was always very supportive through my years there.

I thank Dr. Joseph P. Geunes and Dr. Edwin Romeijn for their many good suggestions in this research as well as their wonderful courses I have taken with them. I also thank another member of my committee, Dr. Juan Feng, for her time and guidance in the last three years.

I am grateful to both of my co-authors Dr. Jiawei Zhang, and Dr. Mark Daskin for their incisive insights. I also thank Prof. H. Tuy for introducing his monotonic optimization methodology to me while he visited Gainesville in August, 2003.

I acknowledge all the help and financial support from the Department of Industrial and Systems Engineering. In particular, I thank Dr. Hearn for assigning me to teach various courses at the University of Florida.

My deepest appreciation goes to my entire family, including my wife, Ngana, my brother, Leping, my parents and in-laws. I could not have finished this research without their support, love, and encouragement. Most specially, I want to thank my wife for being with me in numerous late nights in Gainesville, Miami, and Greenwich, and helping to proofread my manuscript.

Last but not the least, I thank my friends, Leon, Lian, Shu, Gang, Bin, Jean, Chunhua, Jun, Altannar, Jie, Jonathon, Cindy, Silu, Junmin, and many others who made my experience at the University of Florida memorable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

MODELS AND ALGORITHMS FOR RELIABLE FACILITY LOCATION PROBLEMS AND SYSTEM RELIABILITY OPTIMIZATION

By

Roger Lezhou Zhan

August 2007

Chair: Zuo-Jun Max Shen
Major: Industrial and Systems Engineering

Uncertainty is one of the elements that make this world so fascinating and dynamic. However, the existence of uncertainty also poses a great challenge to reliable system design. Our study uses various models and algorithms to address reliability issues in the context of (1) the uncapacitated facility location problem where facilities are vulnerable, and (2) the system reliability problem where components are subject to fail.

We first study the uncapacitated reliable facility location problem in which the failure probabilities are site-specific. The problem is formulated as a two-stage stochastic program and then a nonlinear integer program. Several heuristics that can produce near-optimal solutions are proposed for this computationally difficult problem. The effectiveness of the heuristics is tested through extensive computational studies. The computational results also lead to some managerial insights. For the special case where the failure probability at each facility is a constant (independent of the facility), we provide an approximation algorithm with a worst-case bound of 2.674.

Another part of our research is related to the development and application of a monotonic branch-reduce-bound algorithm, a powerful tool to obtain globally optimal solution to problems in which both the objective function and constraints possess monotonicity. We tailor the algorithm to solve a mixed integer nonlinear programming problem. Its convergence analysis and acceleration techniques are also discussed. The algorithm is then successfully applied to solve system reliability optimization problems

in complex systems, including the redundancy allocation optimization problem and the reliability-redundancy allocation optimization problem. Compared to the existing techniques, the monotonic branch-reduce-bound algorithm is not only versatile but also very efficient in dealing with different types of problems in system reliability.

We also develop several models that can be used to fortify the reliability of the existing facilities. They are the extensions to the models in the first part of the dissertation and offer insights on which facility to choose and to what extent it should be fortified. The properties and solution methodologies of the models are discussed. In particular, a monotonic branch-reduce-bound algorithm is used to solve one of these models. The efficiency of the algorithm is demonstrated in the computational results.

CHAPTER 1
INTRODUCTION

Our study focuses on reliability issues arising in facility location design problems and complex systems. In the former case, we consider mathematical models that minimize the sum of facility opening costs and expected service and penalty costs when facilities are subject to fail from time to time. These failures may come from disruptive events (e.g. labor strikes, supplier business failures, terrorist attacks), or natural disasters (e.g. hurricanes, earthquake). Facility-specific failure probabilities are explicitly considered in our models. To the best of our knowledge, these appear to be the first such models in the literature. These models help to make decisions in the system design phase. Several heuristics and an approximation algorithm are proposed for solving these models.

If facilities have been built but are still subject to fail, we consider models to fortify the reliability of the existing system given limited fortification resources. These models can be reduced to a special class of global optimization problems, called monotonic optimization, in which both the objective function and constraints possess monotonicity. A specialized monotonic branch-reduce-bound algorithm is developed to efficiently solve these problems.

We also examine reliability issues in complex industrial and military systems. The reliability of such a system is measured by the probability of successful operation. We address the issue of allocating unreliable components in the system to achieve the maximum probability of successful operation, a different objective from that used in the facility location model. The problem is generally categorized as a system reliability optimization problem, including the classes of redundancy allocation and reliability-redundancy allocation optimization problems. In redundancy allocation optimization, one is given the option to allocate the appropriate levels of redundancy to maximize reliability or minimize the cost of a system given the design constraints. For example, if a component of reliability level at 0.9 is assigned in parallel to backup another component of the same reliability level, then the overall reliability level is at 0.99, because the probability that both fail is 0.01 assuming both components are independently operational. The redundancy allocation problem can be formulated as a nonlinear integer programming problem. In the reliability-redundancy allocation optimization problem, one is given not only the option to choose different levels of redundancy, but also the reliability levels of each component in a system. For example, in the previously discussed setting, a redundancy level of 2 provides a reliability level of 0.99; one may choose a component whose reliability level is 0.99 at a different cost. The problem can be formulated as a nonlinear mixed integer programming problem. In our study, we show that system reliability optimization is a particular type of monotonic optimization problem. Therefore, it can be efficiently solved by the monotonic branch-reduce-bound algorithm as shown in our computational study for a number of representative problem instances.

The contributions of our study encompass theoretical developments, computational algorithms and practical applications. In particular, we make the following contributions:

- We develop several models to optimally choose facility locations and assign demands to facilities in order to minimize the sum of fixed costs, the expected service, and the penalty cost by taking into consideration that a facility may fail. These models are suitable in the system design phase when no facility has been built yet.
- Several heuristics are proposed to effectively solve the complicated models presented in our work along with the extensive computational tests.
- We present an approximation algorithm for a special case of the reliable facility location problem where the failure probability is not facility-specific. We show that the algorithm admits a worst-case bound of 2.674. This is the first such approximation algorithm for the reliable facility location problem.
- We effectively develop and apply the branch-reduce-bound algorithm for system reliability optimization. The algorithm can be easily adapted to solve various types of problems arising from system reliability optimization and provides some benchmark results for the existing examples in the literature.
- We develop two models to help decision makers to choose which facility to fortify and to what extent it should be fortified when they face the issue of enhancing the

existing system. Such models are among the earliest fortification models in the facility location literature. Several efficient algorithms are also provided.

The structure of our research is depicted in Figure 1-1. In Chapter 2, we present several models for the uncapacitated reliable facility location problem in which some facilities are subject to failure from time to time. These models are the foundation of our research. Besides the general scenario-based model, they include the case in which each facility has a site-specific failure probability, and the case in which each facility has multi-level failure probabilities. The properties and different formulations of the models are thoroughly discussed. Several heuristics are presented along with the computational results.

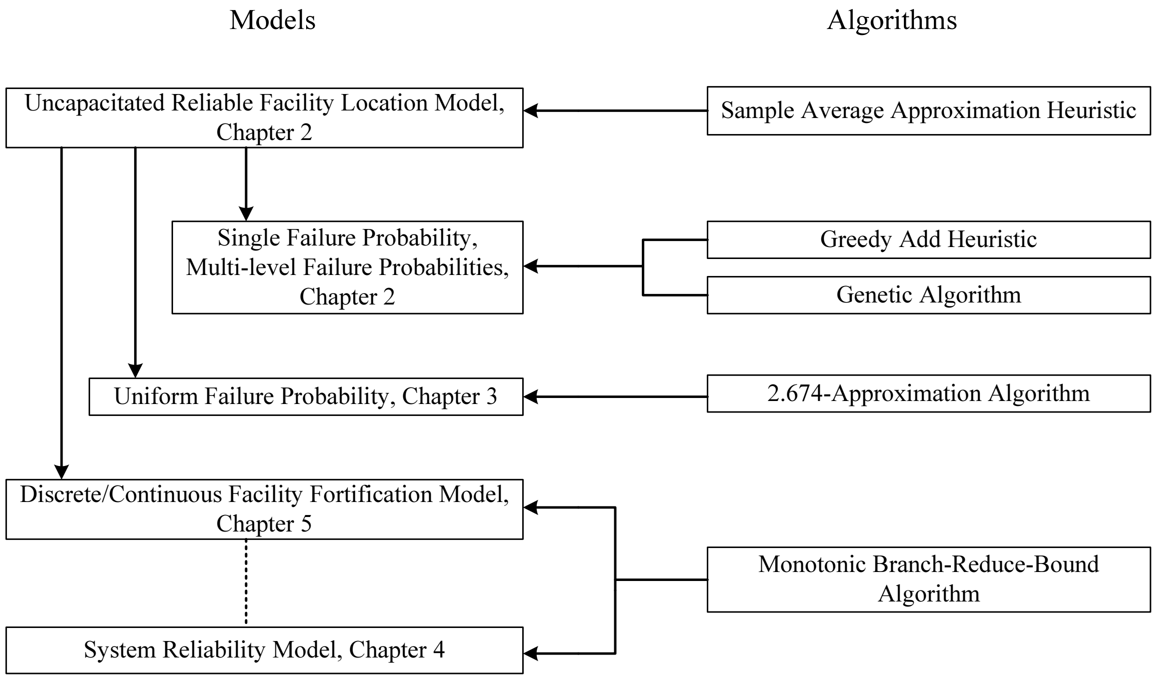

Models
Algorithms
Uncapacitated Reliable Facility Location Model, Chapter 2
Sample Average Approximation Heuristic
Single Failure Probability, Multi-level Failure Probabilities, Chapter 2
Greedy Add Heuristic
Genetic Algorithm
Uniform Failure Probability, Chapter 3
2.674-Approximation Algorithm
Discrete/Continuous Facility Fortification Model, Chapter 5
Monotonic Branch-Reduce-Bound Algorithm
System Reliability Model, Chapter 4

Figure 1-1. Research structure

In Chapter 3, we present a tighter approximation algorithm with a worst-case bound of 2.674 for a special case of the uncapacitated reliable facility location problem, where all failure probabilities are identical.

In Chapter 4, we present a monotonic branch-reduce-bound algorithm for a special case of the nonlinear mixed/pure integer programming problem where both the objective function and constraints possess monotonicity. Its convergence analysis and acceleration techniques are also discussed. The algorithm then is applied to solve system reliability optimization problems in complex systems, including the redundancy allocation optimization problem and the reliability-redundancy allocation optimization problem. The efficiency of the algorithm is demonstrated via the computational results.

Based on the models in Chapter 2, we develop two models that are used to fortify the reliability of the existing facilities. The properties and solution methodologies of the models are discussed. In particular, the monotonic branch-reduce-bound algorithm presented in Chapter 4 is used to solve one of these models. The efficiency of the algorithm is demonstrated through the computational results.

This dissertation is concluded in Chapter 6 with a discussion on future research directions.

CHAPTER 2
RELIABLE FACILITY LOCATION PROBLEM: MODELS AND HEURISTICS

2.1 Introduction

Facility location models have been extensively studied in the literature. Different kinds of facilities have been modeled, such as routers or servers in a communication network, warehouses or distribution centers in a supply chain, hospitals or airports in a public service system. Facility location models typically try to determine where to locate the facilities among a set of candidate sites, and how to assign 'customers' to the facilities, so that the total cost can be minimized or the total profit can be maximized ([34], [8], and [45]). Most models in the literature have treated facilities as if they would never fail; in other words, they were completely reliable. In this chapter, we relax this assumption to model a more realistic case.

The reliability issue we consider is under the framework of the so-called uncapacitated facility location problem (UFLP). In UFLP, we are given a set of demand points, a set of candidate sites, the cost of opening a facility at each location, and the cost of connecting each demand point to any facility. The objective is to open a set of facilities from the candidate sites and assign each demand point to an open facility so as to minimize the total facility opening and connection costs.

UFLP and its generalizations are NP-hard, i.e., unless $P = NP$ they do not admit polynomial-time algorithms to find an optimal solution. There is a vast literature on these NP-hard facility location problems and many solution approaches have been developed in the last four decades, including integer programming, meta-heuristics, and approximation algorithms. One common assumption in this literature is that the input parameters of the problems (costs, demands, facility capacities, etc.) are deterministic. However, such assumptions may not be valid in many realistic situations since many input parameters in the model are uncertain during the decision-making process.

The uncertainties can be generally classified into three categories: provider-side uncertainty, receiver-side uncertainty, and in-between uncertainty. The provider-side uncertainty may capture the randomness in facility capacity and the reliability of facilities, etc.; the receiver-side uncertainty can be the randomness in demands; and the in-between uncertainty may be represented by the random travel time, transportation cost, etc. Most stochastic facility location models focus on the receiver-side and in-between uncertainties ([47]). The common feature of the receiver-side and in-between uncertainties is that the uncertainty does not change the topology of the provider-receiver network once the facilities have been built. However, this is not the case if the built facilities are subject to fail (provider-side uncertainty). If a facility fails, customers originally assigned to it have to be reassigned to other (operational) facilities, and thus the connection cost changes (usually increases).

We focus on the reliability issue of provider-side uncertainty in this chapter. The uncertainty is modeled using two different approaches: 1) by a set of scenarios that specify which subset of the facilities will become non-operational; or 2) by an individual and independent failure probability inherent in each facility. Although each demand point needs to be served by one operational facility only, it should be assigned to a group of facilities that are ordered by levels: in the event of the lowest level facility becoming non-operational, the demand can then be served by the next level facility that is operational; and so on. If all operational facilities are too far away from a demand point, one may choose not to serve this demand point by paying a penalty cost. The objective is thus to minimize the facility opening cost plus the *expected* connection and penalty costs. This problem will be referred to as the uncapacitated reliable facility location problem (URFLP).

In particular, two variants of URFLP are considered in this chapter in terms of the characteristics of the failure probability at each facility. In the first one, we assume that there is only one site-specific failure probability at each facility. We call it the uncapacitated reliable facility location problem with a single-level failure probability(URFLP-SFP). In the other variant, we assume that there are multiple levels of failure probabilities that can be chosen at each facility. We call it the uncapacitated reliable facility location problem with multi-level failure probabilities(URFLP-MFP). Both of them can be modeled by a scenario-based stochastic programming approach and a nonlinear integer programming approach.

URFLP is clearly NP-hard as it generalizes UFLP. We propose several heuristics to solve URFLP. They include the sample average approximation heuristic for the scenario-based model, the greedy adding heuristics, the greedy adding and substitution heuristics, and the genetic algorithm for the nonlinear integer programming model.

The rest of this chapter is organized as follows. In Section 2.2, we review the related literature and provide some basic background for our models. The notation and acronyms are introduced in Section 2.3. In Section 2.4, a scenario-based model is proposed, which is followed by the nonlinear integer model for URFLP-SFP in Section 2.5. Section 2.6 contains the nonlinear integer model for URFLP-MFP. The three heuristics are presented in Section 2.7. In Section 2.8, we conduct computational studies on the performance of the heuristics. In Section 2.9, we conclude the chapter by suggesting several future research directions.

2.2 Literature Review

The importance of uncertainty in decision making has promoted a number of researchers to address stochastic facility location models (e.g., [38, 47]). However, as we pointed out in the Introduction, a majority of the current literature mainly deals with the receiver-side and/or in-between uncertainties. This includes [63], [10], [9], [7] and [42] among others.

The following two papers, [48] and [5], are closely related to this chapter. In [48], the authors assume that some facilities are perfectly reliable while others are subject to failure with the same probability. On the contrary, we assume that the failure probability is site-specific, a much more general case. They formulate their problem as a linear integer program and propose a Lagrangian relaxation solution method. Another related model is proposed in [5], which is based on the p-median problem rather than the framework of UFLP.

There is also a small strand of literature devoted to addressing the fortification of reliability for existing facilities, which includes [43], [44], and [49]. These models typically focus on the interdiction-fortification framework based upon the p-median facility location problem. They are generally formulated as bilevel programming models. Their main focus is to identify the *existing* critical facilities to protect under the events of disruption.

In our model, the failure probabilities are site-specific, which significantly complicates the problem when formulating it as a mathematical program. The model is further extended to URFLP-MFP, where each facility is allowed to have multiple levels of the failure probabilities. We propose two different modeling approaches: a scenario-based stochastic programming approach and a nonlinear integer programming approach. The scenario-based model is attractive due to its structural simplicity and its ability to model dependence among random parameters. But the model becomes computationally expensive as the number of scenarios increases. If the number of scenarios is too large, the nonlinear integer programming approach provides an alternative way to tackle the problem.

The sample average approximation method is widely used for solving complicated stochastic discrete optimization problems, e.g., [24], [42], and [61]. The basic idea of this method is to use a sample average function to estimate the expected value function. Thus the original problem is transformed to the one that can be efficiently solved.

Other types of the commonly used heuristics in optimization are the local search and iterative improvement algorithms ([13], [16], [4] and [3]). These heuristics start with an empty set and repeatedly consider adding a potential facility into the solution. Or they start with an non-empty set and repeatedly consider deleting or substituting a facility in the original set. Typically such algorithms are simple and easy to be incorporated in more sophisticated algorithms, such as Tabu search in [60].

We also apply a genetic algorithm (GA) based heuristic to solve the models we develop. GAs imitate the natural selection in biological evolution. As solution techniques, they maintain a large number of solutions, called the population, and allow each member of the population (called a chromosome) to evolve iteratively into good ones. Some good descriptions of GAs are provided in [11, 14, 39].

GAs have been used to solve many combinatorial optimization problems with success, including various facility location problems, e.g. [22] and [2]. In this chapter, we design a specialized GA for the models in which we are interested. A computational study on all three heuristics is also provided.

2.3 Notations and Acronyms

We first introduce some common notations that will be used throughout the chapter. Let D denote the set of clients or demand points and F denote the set of facilities. $|F|$ is the number of the facilities. Let f_i be the facility cost to open facility i, d_j be the demand of client j, and c_{ij} be the service cost if j is serviced by facility i. The service costs, c_{ij}, are assumed to form a metric, i.e., they satisfy triangle inequalities. For each client $j \in D$, if it is not served by any open and operational facility, then a penalty cost r_j will be incurred.

The acronyms listed in Table 2-1 are frequently used in this chapter.

Table 2-1. Acronyms

Acronym	Meaning
UFLP	Uncapacitated facility location problem
URFLP-SFP	Uncapacitated reliable facility location problem with a single-level failure probability
URFLP-MFP	Uncapacitated reliable facility location problem with multi-level failure probabilities
GA	Genetic algorithm
SAA-H	Sample average approximation heuristic
GAD-H	Greedy adding heuristic
GADS-H	Greedy adding and substitution heuristic

2.4 Uncapacitated Reliable Facility Location Problem: a Scenario-Based Model

We first discuss a scenario-based approach to model URFLP. Given a finite set of scenarios, where each scenario specifies the set of operational facilities, we can formulate URFLP as a two-stage stochastic program with recourse. The first stage decision is to determine which facilities to open before knowing which facilities will be operational. When the uncertainty is resolved, the clients (demand points) will be assigned to the operational facilities. These are the second stage decisions. In this model, we are not allowed to build new facilities in the second stage. In other words, no remedy can be made to the first stage decision, except for optimally assigning the clients to the operational facilities. The objective is to minimize the total expected cost which includes the first stage cost and the expected second stage cost. The expected cost is the sum of the cost of all scenarios times their specific probabilities.

Let $\mathcal{S}$ be the set of scenarios. For any $A \in \mathcal{S}$, let p_A be the probability that scenario A happens. Then URFLP can be formulated as the following two-stage stochastic program.

$$\text{minimize} \quad \sum_{i\in F} f_i y_i + \sum_{A\in\mathcal{S}} p_A g_A(y) \text{ subject to } y_i \in \{0,1\}, \tag{2–1}$$

$$\text{where} \quad g_A(y) = \min \sum_{j\in D}\sum_{i\in F} d_j c_{ij} x_{ij}^A + \sum_{j\in D} d_j r_j z_j^A \tag{2–2}$$

$$\text{s.t.} \quad \sum_{i\in F} x_{ij}^A + z_j^A = 1, \quad \forall j \in D \tag{2–3}$$

$$x_{ij}^A \le y_i, \quad \forall i \in F, j \in D \tag{2–4}$$

$$x_{ij}^A \le I_{A,i}, \quad \forall i \in F, j \in D \tag{2–5}$$

$$x_{ij}^A, z_j^A \in \{0,1\}. \tag{2–6}$$

In the above formulation, the binary variable y_i indicates if facility i is opened in the first stage. Parameter $I_{A,i}$ indicates if facility i is operational under scenario A, which is an input regardless of the value of y_i. Variable x_{ij}^A is the assignment variable which indicates whether client j is assigned to facility i in scenario A or not. Finally, the variable z_j^A indicates whether client j receives service at all or is subject to a penalty. The objective in the formulation, i.e. 2–1, is to minimize the sum of the fixed cost and the expected second stage cost. The objective of the second stage, i.e. 2–2, is to minimize the service and penalty cost. Constraints (2–3) ensure that client j is either assigned to a facility or subject to a penalty at each level k in Scenario A. Constraints (2–4) and (2–5) make sure that no client is assigned to an unopen facility or a nonfunctional facility respectively.

It is straightforward to show that the formulation (2–1) is equivalent to the following mathematical program.

$$
\begin{aligned}
&(\mathit{URFLP\text{-}SP})\\
&\text{minimize} \quad \sum_{i\in F} f_i y_i + \sum_{A\subseteq \mathcal{S}} p_A \left(\sum_{j\in D}\sum_{i\in F} d_j c_{ij} x_{ij}^A + \sum_{j\in D} d_j r_j z_j^A \right)\\
&\text{s.t.} \qquad \sum_{i\in F} x_{ij}^A + z_j^A = 1, \quad \forall j\in D, A\subseteq \mathcal{S}\\
&\qquad\quad x_{ij}^A \le y_i I_{A,i}, \quad \forall i\in F, j\in D, A\subseteq \mathcal{S}\\
&\qquad\quad y_i, x_{ij}^A, z_j^A \in \{0,1\}.
\end{aligned}
$$

One advantage of the scenario-based formulation is that it can easily capture the dependence of the failure probabilities of different facilities by properly defining the scenarios. If the number of scenarios is not too large, it is possible to solve URFLP-SP efficiently and effectively.

However, when the failure probabilities are independent, the possible number of scenarios can be extremely large. Therefore, the number of variables and constraints in URFLP-SP is exponentially large accordingly, which makes it extremely difficult to solve. Under this situation, we propose several alternative nonlinear integer programming formulations and efficient solution algorithms. We discuss these alternative formulations in the next two sections.

2.5 Uncapacitated Reliable Facility Location Problem with a Single-level Failure Probability

2.5.1 Nonlinear Integer Programming Model

In this section, we consider URFLP-SFP, assuming that the failure probabilities of the facilities are independent and each facility has only one failure probability. Let p_i denote the probability that facility i fails. Without loss of generality, we assume that $0 \leq p_i < 1$. The major difficulty here is to compute the expected service cost for each client. In order to overcome this difficulty, we extend Snyder and Daskin's formulation [48] to a more general setting. Comparing to [48], URFLP-SFP can be interpreted slightly differently as follows. Each client should be assigned to a set of facilities initially. The facilities assigned to any client can be differentiated by the levels: in case a lower level facility fails, the next level facility, if operational, will provide service instead.

Mathematically, we define two types of new binary variables x_{ij}^k, z_j^k to capture different level of facilities for a client j. In particular, $x_{ij}^k = 1$ if facility i is the k-th level backup facility of client j and otherwise, $x_{ij}^k = 0$. $z_j^k = 1$ if j has $(k-1)$-th backup facility, but has no k-th backup facility so that j incurs a penalty cost at level k.

Given the variables x_{ij}^k, z_j^k, one can compute the expected total service cost as follows. Consider a client j and its expected service cost at its level-k facility. Client j is served by its level-k facility only if all its assigned facilities at lower levels become non-operational. On the other hand, for any facility l, if it is on the lower level (i.e, less than k) for demand node j, then $\sum_{s=1}^{k-1} x_{lj}^s = 1$, otherwise $\sum_{s=1}^{k-1} x_{lj}^s = 0$. It follows that for client j, the probability that all its lower level facilities fail is $\prod_{l \in F} p_l^{\sum_{s=1}^{k-1} x_{lj}^s}$. If j is severed by facility i, as j's level-k backup facility, then facility i has to be operational which occurs with probability $(1 - p_i)$. Therefore, the expected service cost of client j at level k is $\sum_{i \in F} d_j c_{ij} x_{ij}^k (1 - p_i) \prod_{l \in F} p_l^{\sum_{s=1}^{k-1} x_{lj}^s}$. Similarly, we can calculate the expected penalty cost of client j at level k, which is $\prod_{l \in F} p_l^{\sum_{s=1}^{k-1} x_{lj}^s} d_j r_j z_j^k$.

The above discussion leads to a nonlinear integer programming formulation for URFLP-SFP.

$$\text{minimize} \quad \sum_{i\in F} f_i y_i + \sum_{j\in D}\sum_{k=1}^{|F|}\sum_{i\in F} d_j c_{ij} x_{ij}^k (1-p_i) \prod_{l\in F} p_l^{\sum_{s=1}^{k-1} x_{lj}^s}$$

$$+\sum_{j\in D}\sum_{k=1}^{|F|+1}\prod_{l\in F} p_l^{\sum_{s=1}^{k-1} x_{lj}^s} d_j r_j z_j^k \tag{2–7}$$

$$\text{subject to} \quad \sum_{i\in F} x_{ij}^k + \sum_{t=1}^{k} z_j^t = 1, \quad \forall j \in D, k = 1, ..., |F|+1 \tag{2–8}$$

$$x_{ij}^k \le y_i, \quad \forall i \in F, j \in D, k = 1, ..., |F| \tag{2–9}$$

$$\sum_{k=1}^{|F|} x_{ij}^k \le 1, \quad \forall i \in F, j \in D \tag{2–10}$$

$$x_{ij}^k, z_j^k, y_i \in \{0,1\}. \tag{2–11}$$

The decision variables x_{ij}^k, z_j^k are defined earlier. The indicator variable $y_i = 1$ if facility i is open in the first stage; otherwise $y_i = 0$. The objective function (2–7) is the summation of the facility cost, the expected service cost, and the expected penalty cost. Constraints (2–8) ensure that client j is either assigned to a facility or subject to a penalty at each level k. Constraints (2–9) make sure that no client is assigned to an unopen facility. Constraints (2–10) prohibit a client from being assigned to a specific facility at more than one level. Note that constraints (2–9) and (2–10) can be tightened as

$$\sum_{k=1}^{|F|} x_{ij}^k \le y_i \quad \forall i \in F, j \in D. \tag{2–12}$$

2.5.2 Model Properties

In formulation (URFLP-SFP), we do not explicitly require that a closer open facility be assigned as a lower level facility to a particular demand point. However, according to the following proposition, it is true that the level assignments among the open facilities are based on the relative distances between the demand point and the facilities regardless of the failure probabilities.

Proposition 2.1. *In any optimal solution to URFLP-SFP, for any client j, if $x_{uj}^k = x_{vj}^{k+1} = 1$, then $c_{uj} \le c_{vj}$.*

Proof. We prove the proposition by contradiction. Suppose $c_{uj} > c_{vj}$, we will show that by "swapping" the assignment of u and v, the objective function will strictly decrease.

In particular, if we set $x_{uj}^{k+1} = 1$ and $x_{vj}^k = 1$ with the values of other variables unchanged, we can compute the new objective value. The difference between the new objective value and the original one is

$$\begin{aligned} & d_j\left(c_{vj}(1-p_v)\bar{p_k} + c_{uj}(1-p_u)\bar{p_k}p_v\right) - d_j\left(c_{uj}(1-p_u)\bar{p_k} + c_{vj}(1-p_v)\bar{p_k}p_u\right) \\ = \quad & d_j(c_{vj} - c_{uj})(1-p_v)(1-p_u)\bar{p_k} \\ < 0, & \end{aligned}$$

where

$$\bar{p_k} = \prod_{l \in F} p_l^{\sum_{s=1}^{k-1} x_{lj}^s}.$$

The last inequality holds because $\bar{p_k} > 0$ and it is assumed that $p_u < 1$, $p_v < 1$, and $c_{vj} < c_{uj}$. This is clearly a contradiction to the optimality of the original solution. Therefore, $c_{uj} \le c_{vj}$. □

An implication of Proposition 2.1 is that if the set of open facilities is determined, then it is trivial to solve the level assignment problem for each client: assigning levels according to the relative distances of different facilities to the client. If at some level the distance is beyond the penalty cost, then no facility will be assigned at this level (and higher ones) and the demand node simply takes the (cheaper) penalty.

We would like to point out the relationship between formulation (URFLP-SP) and formulation (URFLP-SFP). Since these two formulations are just two ways of modeling the same problem, they should have the same minimum cost as long as the inputs to the two models are consistent. In formulation (URFLP-SFP), each facility i has independent failure probability p_i. This implies that there are $2^{|F|}$ scenarios and the probability that each of the scenarios occurs can be easily calculated. The corresponding values serve as inputs to formulation (URFLP-SP). We also notice that, in formulation (URFLP-SP), it is straightforward to obtain an optimal second stage solution for a given first stage solution. In particular, at the second stage, every client will be assigned to and be served by an open and operational facility that is closest to the client at the lowest possible level; if the service cost is higher than the penalty cost, then it takes the penalty.

2.5.3 A Special Case: Uniform Failure Probabilities

Now we consider a special case of URFLP where all facilities have the same failure probability, i.e., $p_i = p, \forall i \in F$. This assumption simplifies formulation (URFLP-SFP) considerably based on the following observation. Because $p_i = p, \forall i \in F$, it is straightforward that $\prod_{l \in F} p_l^{\sum_{s=1}^{k-1} x_{lj}^s} = p^{k-1}$, which is independent of the values of x_{lj}^s. This property is implicitly used in a multi-objective formulation proposed in [48].

Based on the above observation, we are able to reduce formulation (URFLP-SFP) to a linear integer program as follows.

(*URFLP-IP*)

$$\begin{aligned} \text{minimize} \quad & \sum_{i\in F} f_i y_i + \sum_{j\in D}\sum_{k=1}^{|F|}\sum_{i\in F} d_j c_{ij} x_{ij}^k (1-p)p^{k-1} + \sum_{j\in D}\sum_{k=1}^{|F|+1} p^{k-1} d_j r_j z_j^k && \text{(2–13)} \\ \text{subject to} \quad & \sum_{i\in F} x_{ij}^k + \sum_{t=1}^{k} z_j^t = 1, \quad \forall j \in D, k = 1, ..., |F|+1 \\ & \sum_{k=1}^{|F|} x_{ij}^k \le y_i, \quad \forall i \in F, j \in D \\ & x_{ij}^k, z_j^k, y_i \in \{0,1\}. \end{aligned}$$

This special case can be solved to optimality by using commercial solvers such as CPLEX. We use this case as a bench mark in the computational tests conducted in Section 2.8.

2.6 Uncapacitated Reliable Facility Location Problem with Multi-level Failure Probabilities

In this section, we extend URFLP-SFP to URFLP-MFP so that each facility has multi-level failure probabilities. In this model, the decision makers can make some key facilities more sustainable than others by investing more if necessary. In URFLP-MFP, we model the failure probabilities as functions of the initial fixed investment. To do so, we introduce t to devote different investment levels. In addition, f_{ti} is denoted as the fixed cost at the facility i at the level t; y_{ti}, the decision binary variables for the facility i at the level t. That is, $y_{ti} = 1$, if a level t investment is put at facility i; otherwise, $y_{ti} = 0$. We assume $0 \leq t \leq U$.

Given different investment levels at facility i, the output of the failure probability at facility i, $\mathcal{P}'_i(y_{ti})$, is determined by y_{ti}:

$$\mathcal{P}'_i(y_{ti}) = \sum_t (P(f_{ti})y_{ti}) \tag{2–14}$$

with $f_{0i} = 0$, $P(0) = 1$, and $P(\cdot)$ is a decreasing function. $f_{0i} = 0$ and $P(0) = 1$ imply that if there is no investment at facility i, then it is completely nonfunctional.

After the failure probability at facility i is determined, URFLP-MFP is essentially no different from URFLP-SFP. In the following formulation, p_i in the objective function 2–7 of URFLP-SFP is replaced by $\mathcal{P}'_i(y_{ti})$.

(*URFLP-MFP-1*)

$$\text{minimize} \quad \sum_{i\in F}\sum_{t}(f_{ti}y_{ti}) + \sum_{j\in D}\sum_{k=1}^{|F|}\sum_{i\in F} d_j c_{ij} x_{ij}^k (1-\mathcal{P}_i'(y_{ti})) \prod_{l\in F}(\mathcal{P}_l'(y_{tl}))^{\sum_{s=1}^{k-1} x_{lj}^s}$$

$$+\sum_{j\in D}\sum_{k=1}^{|F|+1}\prod_{l\in F}(\mathcal{P}_l'(y_{tl}))^{\sum_{s=1}^{k-1} x_{lj}^s} d_j r_j z_j^k \tag{2–15}$$

$$\text{subject to} \quad \sum_{i\in F} x_{ij}^k + \sum_{t=1}^{k} z_j^t = 1 \quad \forall j, k = 1, ..., |F|+1 \tag{2–16}$$

$$x_{ij}^k \le \sum_t y_{ti} \quad \forall i \in F, j \in D, k = 1, ..., |F| \tag{2–17}$$

$$\sum_t y_{ti} \le 1 \quad \forall i = 1, ..., |F| \tag{2–18}$$

$$\sum_{k=1}^{|F|} x_{ij}^k \le 1 \quad \forall i \in F, j \in D \tag{2–19}$$

$$x_{ij}^k, z_j^k, y_{ti} \in \{0,1\}, \tag{2–20}$$

where constraints (2–18) ensure at most one investment level is allowed at each facility and all the other constraints are similar to the ones in URFLP-SFP. Note that Proposition 2.1 still holds in this model.

The above formulation, URFLP-MFP-1, is a binary model. URFLP-MFP can also be modeled as a regular integer model by reinterpreting the definition of y_i as the investment level at facility i. Thus, y_i is not binary any more; $0 \le y_i \le U_i$, where U_i is the highest level at which facility i can be possibly built. The corresponding failure probability and the fixed cost at facility i are denoted by functions $P_i(y_i)$ and $F_i(y_i)$ respectively. $P_i(y_i), i \in F$, are nonincreasing functions of y_i and $P_i(0) = 1$, whereas $F_i(y_i), i \in F$, are nondecreasing functions of y_i and $F_i(0) = 0$. Then URFLP-MFP can be modeled as follows.

(*URFLP-MFP-2*)

$$\text{minimize} \quad \sum_{i\in F} F_i(y_i) + \sum_{j\in D}\sum_{k=1}^{|F|}\sum_{i\in F} d_j c_{ij} x_{ij}^k \left(1 - P_i(y_i)\right) \prod_{l\in F} (P_l(y_l))^{\sum_{s=1}^{k-1} x_{lj}^s}$$

$$+ \sum_{j\in D}\sum_{k=1}^{|F|+1} \prod_{l\in F} (P_l(y_l))^{\sum_{s=1}^{k-1} x_{lj}^s} d_j r_j z_j^k \tag{2–21}$$

$$\text{subject to} \quad \sum_{i\in F} x_{ij}^k + \sum_{t=1}^{k} z_j^t = 1 \quad \forall j, k = 1, ..., |F|+1 \tag{2–22}$$

$$x_{ij}^k \le \min\{y_i, 1\} \quad \forall i \in F, j \in D, k = 1, ..., |F| \tag{2–23}$$

$$\sum_{k=1}^{|F|} x_{ij}^k \le 1 \quad \forall i \in F, j \in D \tag{2–24}$$

$$0 \le y_i \le U_i, \quad \forall i \in F, \tag{2–25}$$

$$x_{ij}^k, z_j^k \in \{0,1\}, y_i\text{: integer}, \tag{2–26}$$

where constraints (2–23) ensure no client is assigned to an unopen facility. That is, when $y_i = 0$, $x_{ij}^k = 0$, $\forall i, j, k = 1, ..., |F|$. Because of the binary constraints (2–26) on x_{ij}^k, the right hand side of constraints (2–23) can be replaced by y_i without affecting the feasible domain.

2.7 Solution Methodologies

2.7.1 Sample Average Approximation Heuristic

The Sample Average Approximation (SAA) method is widely used for solving complicated stochastic discrete optimization problems ([24], [42], and [61]). The basic idea of this method is to randomly generate samples, then use a sample average function to estimate the true expected value function. By doing so, the original problem is reduced to a relatively small problem that can be repeatedly solved. Such an approach has been used by various authors over the years. We apply the following procedures to solve model URFLP-SP.

The SAA Heuristic

- **Step 1:** Randomly generate a sample of N scenarios $\{A_1, \cdots, A_N\}$ and solve the following SAA problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{i\in F} f_i y_i + \sum_{s=1}^{N} \frac{1}{N} \left(\sum_{j\in D} \sum_{i\in F} d_j c_{ij} x_{ij}^s + \sum_{j\in D} d_j r_j z_j^s \right) \\ \text{s.t.} \quad & \sum_{i\in F} x_{ij}^s + z_j^s = 1, \quad \forall j \in D, s = 1, \dots N \\ & x_{ij}^s \le y_i I_{A_s,i} \quad \forall i \in F, j \in D, s = 1, \dots N \\ & y_i, x_{ij}^s, z_j^s \in [0,1] \end{aligned}$$

  Repeat this step M times. For each $m = 1, 2, \cdots, M$, let y^m and v^m be the corresponding optimal solution and its optimal objective value respectively. In view of Proposition 2.1, the level assignment decision (the second stage decision) can be solely determined by y^m. Compute the true objective value $\hat{v}^m$ using the formulation of URFLP-SFP for each y^m, $m = 1, 2, \cdots, M$.

- **Step 2:** Among the M solutions obtained in the first step, output the one with the minimum objective value, i.e., $\hat{v}^{min} = \min_{m=1,\dots,M} \hat{v}^m$.

Two remarks are in order. First, in a standard SAA approach ([42] and [61]), one additional independent sample is needed to estimate the true expected value $\hat{v}^m$. But in our case, an analytical formula is ready for estimating the true expected value. Second, the average of the v^m values, i.e. $\bar{v}^M = \sum_{m=1}^{M} v^m$, does not provide a statistical lower bound for the optimal value. This is different from the result in [42] where the uncertainty only comes from demand-side. In the current model, different samples may lead to different solution spaces of the problem, due to the provider-side uncertainty. Therefore the average value $\bar{v}^M$ is no longer unbiased to the true expected value. Nonetheless, $\bar{v}^M$ may still serve as a good indication of the quality of the solution from the SAA approach, as we will illustrate in the computational tests.

2.7.2 Greedy Methods

In this section, we propose two related heuristics to solve the general URFLP: Greedy Adding Heuristic (GAD-H), and Greedy Adding and Substitution Heuristic (GADS-H). These two heuristics based on greedy local search and iterative improvement algorithms.

In formulation (URFLP-SFP), Proposition 2.1 ensures that the level assignments can be easily derived for a given set of open facilities. Therefore, one can concentrate on selecting a set of open facilities without worrying too much on the decisions of level assignment. Let $v(T)$ denote the objective function value given by the set of open facilities, T. Let T^t be the set of open facilities at step t, and Φ be the empty set.

The Greedy Adding Heuristic

- **Step 1:** Initially the set of open facilities is empty. Set $t = 0$ and $T^t = \Phi$.
- **Step 2:** Choose a facility from the remaining candidates to open such that it can reduce the total cost the most. Add this facility to the facility set. That is,

$$\begin{aligned} & t = t + 1, \\ & j_t = \arg_{j \in F \setminus T^{t-1}} \min(v(T^{t-1} \cup \{j\})), \\ & T^t = T^{t-1} \cup \{j_t\}. \end{aligned}$$

- **Step 3:** Repeat Step 2 until the current solution cannot be improved further.

In general, as we can see from the computational tests later, the greedy adding heuristic is able to find a high quality solution very efficiently. The complexity of this heuristic is $O(n^4 \log n)$, where $n = |F|$. Given T^{t-1}, it takes $O(n \log n)$ to do the level assignments for each node, mainly because it involves a sort process that is in complexity of $O(n \log n)$. There is n such nodes, so it takes $O(n^2 \log n)$ to evaluate the value of $v(T^{t-1})$. In the worst case, it takes n such evaluations to get the most cost effective facility, j_t at step t. The greedy adding process iterates at most n times, which leads to the complexity of $O(n^4 \log n)$ for GAD-H.

After the greedy adding heuristic, we perform the following greedy substitution heuristic to further improve the solution: at each iteration, a substitute facility is chosen to replace the existing open facility if doing so reduces the total cost the most. This procedure is repeated until no substitute facility can be found to further reduce the total cost. The substitution can be a null facility. Replacing an open facility with a null facility means that we close the facility. After the substitution process, another greedy adding procedure is performed to further improve the solution. The whole process (a greedy adding procedure followed by a greedy substitution procedure then followed by another greedy adding procedure) is called the greedy adding and substitution heuristic (GADS-H).

2.7.3 Genetic Algorithm Based Heuristic

Genetic algorithm (GA) based heuristics have been widely used to solve combinatorial optimization problems. A GA imitates the mechanism of natural selection and natural genetics. Generally, a GA starts with an initial set of solutions called a *population*. Each member of the population is called a *chromosome*, representing an encoded version of a solution to the optimization problem at hand. The goal of an encoding is to translate a solution into a string of *genes* that make up the chromosome. There is a *fitness function* that evaluates the quality of a chromosome at each iteration, called a *generation*. The generation evolves through several operators: *crossover*, *reproduction*, *immigration*, and *mutation*. The crossover operator is a process to produce one or more offspring from the current generation. Reproduction is simply a process that copies the best solutions from the previous generation to the next. The immigration operator is to randomly create certain new chromosome in each generation. The mutation operator is to randomly change the genes in a chromosome to introduce randomness in each generation. The population size through each generation is kept constant. After several generations, the best solutions converge to an optimal or sub-optimal solution to the problem. Several comprehensive treatments of GAs are given in [11, 14, 39].

In principle, a GA can be applied to any optimization problem. But there is no generic GA since it requires many design decisions, such as the encoding of the chromosome, the selection of parents, the method of the crossover operator. In this section, we describe a GA that is suitable for URFLP.

The chromosome or the solution of the URFLP model is represented as a bit stream with one position for every candidate location. We will use *chromosome* and *solution* interchangeably. A “1” in position k is interpreted as that candidate site k is located to open, while a “0” indicates that it is not. Since it is optimal to assign demand nodes to facilities based on the distance between the demand node and the facility as indicated in Proposition 2.1, we do not need an explicit encoding of the demand-to-facility assignments. For model URFLP-MFP, an additional element is encoded to represent the level at which we invest in an open facility. Table 2-2 shows the encoding for a system with 10 candidate sites for model URFLP-MFP, for example, with open facilities at nodes 3, 4 and 8 at investment levels 2, 1 and 3 respectively.

Table 2-2. Sample chromosome for model URFLP-MFP

Candidate site	1	2	3	4	5	6	7	8	9	10
Open?	0	0	1	1	0	0	0	1	0	0
Investment level	0	0	2	1	0	0	0	3	0	0

Chromosomes are evaluated based on the value of objective function. A chromosome with a smaller objective value is fitter than one with a larger objective value. The following parameters are employed in our description of heuristic GA-H.

Table 2-3. GA-H parameters

Parameter	Notation
Population size	N_P
Maximum number of generations	N_G
Maximum number of generations without improvement	N_M
Number of reproduction	N_R
Number of immigration	N_I
Mutation probability	P_M

The initial population is randomly generated. For each solution of the population, sites are selected randomly and the solution is checked against all other solutions in the emerging population to ensure that the solution is unique. If it is, the solution is added to the emerging population; if it is not, the solution is rejected and a new random solution is generated. This process stops until N_P distinct solutions are populated.

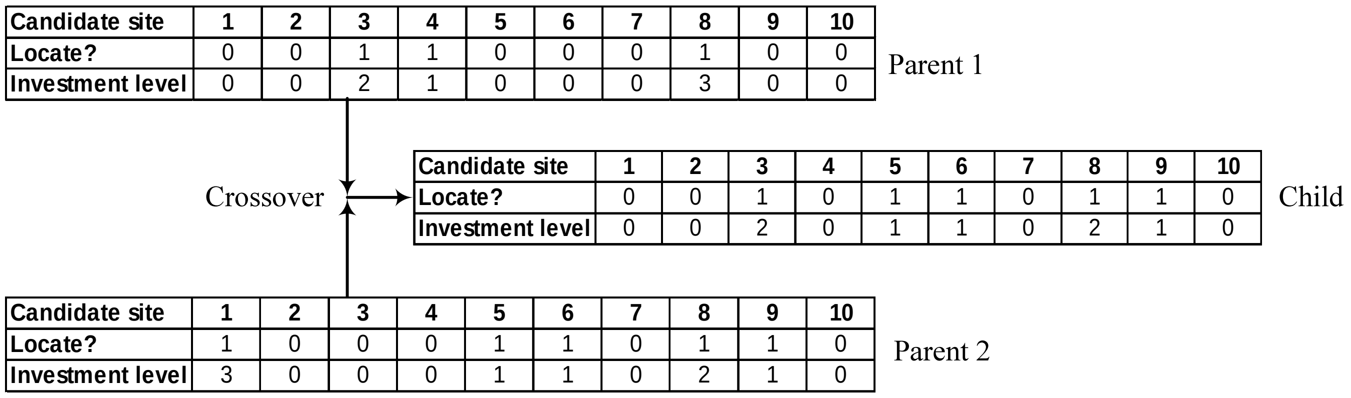
Once we have a generation of solutions or chromosomes, we employ several operators to create the next generation whose initial population size is zero. Reproduction carries forward the best N_R solutions from the current generation to the next one. Naturally, $N_R < N_P$. Immigration creates N_I solutions randomly, such that each new solution is different from any of the solutions already in the emerging population. This immigration process is identical to the process used for generating the initial solutions.

The main operator is crossover. Two solutions are selected at random from the population at the current generation, with a bias toward the better solutions. The probability of selecting the j^{th} best solution is given by $\frac{N_P+1-j}{N_P(N_P+1)/2}$. Note N_P is the number of solutions in the population. The denominator, $N_P(N_P+1)/2$, is the sum from 1 to N_P. The numerator, N_P+1-j, is the reverse order of j's fitness among all N_P solutions. These values are listed in Table 2-4 as the weight in probability evaluation. As we can see that the better a solution is, the higher weight it is assigned.

Table 2-4. Selection probabilities for a population with N_P solutions

Solution Rank	Weight in Probability Evaluation	Probability
1	N_P	$\frac{N_P}{N_P(N_P+1)/2}$
2	N_P-1	$\frac{N_P-1}{N_P(N_P+1)/2}$
$\vdots$	$\vdots$	$\vdots$
j	N_P+1-j	$\frac{N_P+1-j}{N_P(N_P+1)/2}$
$\vdots$	$\vdots$	$\vdots$
N_P	1	$\frac{1}{N_P(N_P+1)/2}$
Total	$N_P(N_P+1)/2$	**1**

After two non-identical parents are selected, a one-point crossover position in the list of genes (candidate sites) is randomly selected. A child solution is constructed using the genes (candidate sites) to the left of the crossover position from parent 1 and to the right of the crossover position from parent 2. This process is depicted in Figure 2-1 for two solutions with the one-point crossover position at 3. Encoding of the child solution in this example is the following: values of positions 1 to 3 are from those in the same positions as in parent 1; and value of positions 4 to 10 are from those in the same positions as in parent 2.



Candidate site	1	2	3	4	5	6	7	8	9	10
Locate?	0	0	1	1	0	0	0	1	0	0
Investment level	0	0	2	1	0	0	0	3	0	0

Candidate site	1	2	3	4	5	6	7	8	9	10
Locate?	0	0	1	0	1	1	0	1	1	0
Investment level	0	0	2	0	1	1	0	2	1	0

Candidate site	1	2	3	4	5	6	7	8	9	10
Locate?	1	0	0	0	1	1	0	1	1	0
Investment level	3	0	0	0	1	1	0	2	1	0

Figure 2-1. Example of crossover operation at position 3

After a child solution has been constructed in the manner outlined above, with probability P_M, the solution is mutated. Mutation is accomplished by randomly selecting two candidate sites: one at which a facility opens and one at which a facility closes; then swapping their states: from open (“1”) to close (“0”), and from close (“0”) to open (“1”). In the case of model URFLP-SFP, a randomly selected investment level is associated with the newly open facility site.

If the child solution generated in this manner differs from all other solutions in the emerging population, it is added to the population; if it does not, the entire process (of parent selection, crossover, and mutation) is repeated. We continue adding solutions to the population until the population contains N_P total solutions. In other words, the size of each generation is maintained to be the same.

The whole process is repeated until one of the following termination criteria is met: (1) the algorithm reaches N_G generations, or (2) it fails to improve the best-known solution in N_M generations.

2.8 Computational Results

In this section, we compare the computational performance of the four heuristics: the sample average approximation heuristic (SAA-H), the greedy adding heuristic (GAD-H), the greedy adding and substitution heuristic (GADS-H), and the genetic algorithm based heuristic (GA-H). In order to evaluate the performance of these four heuristics, we first apply them to solve URFLP-IP, where each facility has the identical failure probability. The reason is that URFLP-IP admits a linear integer programming formulation that can be solved to optimality by using commercial solvers such as CPLEX, so that we can compare the heuristic results with the exact solutions, which helps to better evaluate the performance of each heuristic.

The test dataset is generated as follows. Coordinates of the sites were drawn from $U[0,1] \times U[0,1]$, demand of each site was drawn from $U[0,1000]$ and rounded to the nearest integer, fixed facility costs were drawn from $U[500,1500]$ and rounded to the nearest integer, and penalty costs were drawn from $U[0,15]$. Further, the transportation cost c_{ij} is set to be the Euclidean distance between points i and j. The number of sites varies from 10 to 100. The dataset is available in Appendix B.

All the algorithms were coded in C++ and tested on a Dell Optiplex GX620 computer running the Windows XP operating system with a Pentium IV 3.6 GHz processor and 1.0 GB RAM.

2.8.1 Sample Average Approximation Heuristic

We first test how the sample size (N) affects (1) the quality of the solution; and (2) the efficiency of the program, for a 50-node dataset with the uniform failure probabilities varying from 0 to 1. Table 2-5 lists the objective values obtained from SAA-H when $M = 1$ and the sample size varies from 10 to 200.

It is clear from Table 2-5 that the solution quality can be improved by increasing the sample size. The ratios of the objective value obtained from SAA-H with sample sizes 10, 50, 100, 150, 200, and 250 to the optimal value are plotted in Figure 2-2.

Table 2-5. Objective values from SAA-H for the 50-node dataset

Failure Probability	N10	N20	N30	N40	N50
0	7197.27	7197.27	7197.27	7197.27	7197.27
0.1	7956.03	7956.03	7763.80	7763.80	7763.80
0.2	9429.49	8770.31	8425.99	8425.99	8425.99
0.3	9908.52	10059.90	9669.49	10201.50	10095.20
0.4	11546.90	11753.50	11984.00	13887.60	11937.60
0.5	18727.50	15128.1	13120.60	15052.80	13546.40
0.6	27429.00	18161.70	17946.40	17946.40	17946.40
0.7	32965.30	33139.20	27374.80	23659.30	23690.40
0.8	62876.90	46387.50	35743.60	35743.60	35743.60
0.9	84406.10	69255.40	54722.30	54728.10	55647.40
1.0	128009.00	128009.00	128009.00	128009.00	128009.00

Failure Probability	N100	N150	N200	N250	**Exact**
0	7197.27	7197.27	7197.27	7197.27	7197.27
0.1	7763.80	7763.80	7763.80	7763.80	7763.80
0.2	8425.99	8425.99	8425.99	8425.99	8425.99
0.3	9387.85	9387.85	9275.99	9275.99	9275.99
0.4	10848.20	10599.90	10566.30	10762.00	10253.90
0.5	13157.20	13041.50	13099.70	12134.20	11603.00
0.6	17762.80	15609.50	14559.10	14559.10	13416.80
0.7	20740.30	20671.60	19070.90	17289.00	16157.20
0.8	29218.10	26731.10	24066.80	23252.20	21500.70
0.9	46546.80	45061.00	42555.50	38316.90	35987.70
1.0	128009.00	128009.00	128009.00	128009.00	128009.00

Figure 2-2 shows that SAA-H obtains fairly good solutions with a small sample size when the failure probability is low ($p \leq 0.4$). In contrast, a large sample size is needed to achieve such quality of solutions for the cases with a higher failure probability. The following could be a possible explanation. Note that a sample is a collection of different scenarios. When the failure probability is low, the majority of the facilities are candidate sites for opening in each scenario, so the sets of candidate facilities are similar in different scenarios. Thus, any individual sample can capture the characteristics of the system pretty well, and the corresponding solution obtained from SAA-H is close to optimal. For the extreme case where $p = 0$, all facilities are available to open in each scenario, so the sets of available facilities are the same in each scenario and the SAA-H can produce the exact solution in this case. In the case of the higher failure probability, the candidate sites for

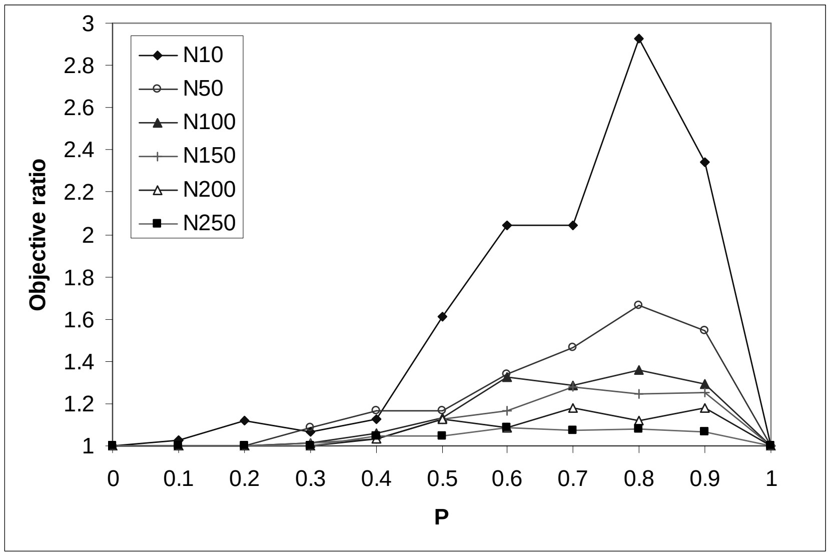


Figure 2-2. Objective ratio across different sample sizes for the 50-node dataset

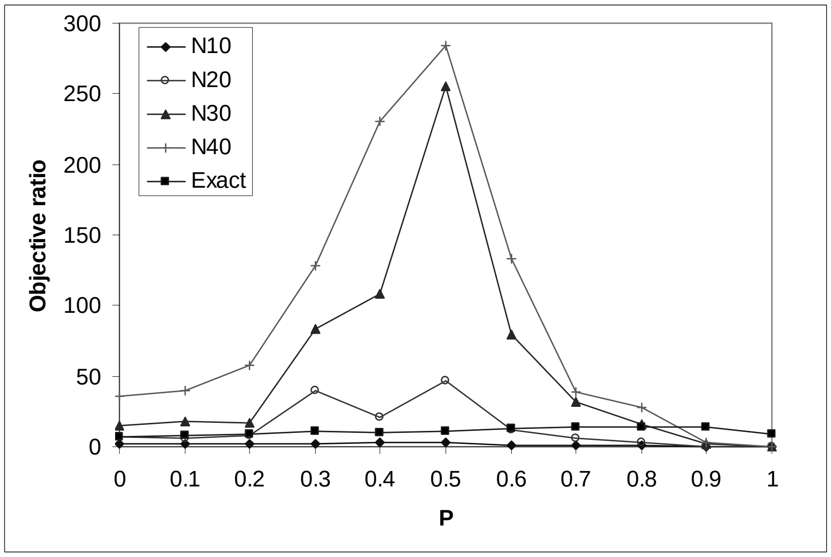


Figure 2-3. CPU time across different sample sizes for the 50-node dataset

opening in each scenario is relatively small and a scenario can be quite different from another one. Thus an increased size of sample can help to capture the characteristics of the system uncertainty.

Figure 2-3 depicts the computation time from the run with sample sizes 10, 20, 30, and 40. It shows that the case with $N = 10$ is the only one that requires slightly less time than the exact algorithm, whereas others require more time as the sample size increases. Another interesting pattern in Figure 2-3 is the tail effect of the CPU time in terms of the failure probability. SAA-H spends more time to obtain a solution when the failure probability is around 0.5. One possible way to explain this phenomena is the following: when the failure probability is around 0.5, the constraints $x_{ij}^s \leq y_i I_{A_s,i}$ among different samples are quite different. As a result, the problem size increases, so does the computational time.

Next, we examine the effect of the replication number (M) on the solution quality by fixing $N = 30$. Table 2-6 provides the objective values obtained when $M = 5, 10, 15, 20$. From the objective values obtained in different replication numbers, we can see that the increase of the replication number has not affected the solution quality too much. The gap in this table is defined as $\frac{\hat{v}^{min} - \bar{v}^M}{\bar{v}^M} \times 100\%$. The negative numbers in the "gap" column reveal that $\bar{v}^M$ is not always a lower bound for $\hat{v}^{min}$. However, it is a good indication of the quality of the solution from SAA-H. In this particular case, if the gap is within $\pm 10\%$, the obtained objective value is close to the optimal value.

In general, SAA-H is capable to produce a fairly good solution with a large sample size for the uniform case. But it also requires a tremendous amount of time to do so and may run out of memory due to the increase of problem size. We defer presenting the computational results of URFLP-SFP to Section 2.8.4.

2.8.2 Greedy Methods: GAD-H and GADS-H

In this section, we report the computational results of GAD-H and GADS-H on the URFLP-IP model.

Table 2-7 lists the computational results of a 50-node dataset when the failure probability varies from 0 to 1. The first column, P, is the failure probability at each facility. The "gap" column is defined as the percentage difference between the cost of the solution obtained by GAD-H and the optimal cost.

As we can see from Table 2-7, GAD-H finds optimal or near-optimal solutions in most cases in less than 0.05 seconds. Compared to the exact method using CPLEX, it takes

Table 2-6. Runs from SAA-H for the 50-node dataset

P	$M = 5$			$M = 10$			Exact
	$\hat{v}^{min}$	$\bar{v}^5$	gap	$\hat{v}^{min}$	$\bar{v}^{10}$	gap	
0.0	7197.27	7197.27	0.00	7197.27	7197.27	0.00	7197.27
0.1	7763.80	7687.03	1.00	7763.80	7760.14	0.05	7763.80
0.2	8425.99	8315.65	1.33	8425.99	8436.50	-0.12	8425.99
0.3	9414.40	9054.28	6.79	9378.06	9112.31	3.32	9275.99
0.4	10872.60	9740.45	23.03	10479.80	9814.09	10.79	10253.90
0.5	11932.00	10457.00	25.47	11932	10497.60	13.66	11603.00
0.6	17825.90	11377.60	57.73	17335.50	11475.90	55.33	13416.80
0.7	27157.40	12758.80	114.56	23227.40	12816.20	111.90	16157.20
0.8	34912.50	14758.30	142.19	31284.20	14761.90	136.50	21500.70
0.9	54722.30	19703.50	177.73	54628.80	19428.60	181.66	35987.70
1.0	128009.00	128009.00	0.00	128009.00	128009.00	0.00	128009.00
P	$M = 15$			$M = 20$			Exact
	$\hat{v}^{min}$	$\bar{v}^{15}$	gap	$\hat{v}^{min}$	$\bar{v}^{20}$	gap	
0.0	7197.27	7197.27	0.00	7197.27	7197.27	0.00	7197.27
0.1	7763.80	7784.17	-0.26	7763.80	7768.5	-0.06	7763.80
0.2	8425.99	8484.24	-0.69	8425.99	8453.66	-0.33	8425.99
0.3	9378.06	9138.80	2.62	9275.99	9152.36	2.47	9275.99
0.4	10479.80	9826.75	6.65	10259.90	9842.41	6.48	10253.90
0.5	11932.00	10507.40	13.56	11932.00	10530.80	13.31	11603.00
0.6	17335.50	11523.50	50.44	17291.50	11563.00	49.92	13416.80
0.7	22894.20	12766.10	81.95	22894.20	12747.70	79.59	16157.20
0.8	31284.20	14736.80	112.29	31284.20	14746.00	112.15	21500.70
0.9	54628.80	19752.00	176.57	53343.90	19811.80	175.74	35987.70
1.0	128009.00	128009.00	0.00	128009.00	128009.00	0.00	128009.00

much less time. The greedy adding algorithm seems to perform better when the facility failure probability is high. It actually finds optimal solutions when the failure probability exceeds 0.5. This is in contrast to the performance of the SAA-H, which works better when the failure probability is low.

As we pointed out in Section 2.7.2, the solution quality of GAD-H can be further improved by GADS-H. This is clearly demonstrated in the following computational results. GADS-H actually finds the optimal solutions for all instances in Table 2-7 and the CPU times are comparable with those reported by GAD-H. The results are summarized in Table 2-8.

Table 2-7. Fifty-node uniform case: greedy adding and exact solution

	Greedy Adding Heuristic		Exact Algorithm		
P	Objective	Time (s)	Objective	Time (s)	gap(%)
0.0	7551.02	0.00	7197.27	6.94	4.92
0.1	8053.11	0.00	7763.80	7.61	3.73
0.2	8637.46	0.00	8425.99	8.94	2.51
0.3	9309.50	0.00	9275.99	10.62	0.36
0.4	10253.90	0.02	10253.90	10.38	0.00
0.5	11622.80	0.02	11603.00	10.86	0.17
0.6	13416.80	0.02	13416.80	12.84	0.00
0.7	16157.20	0.02	16157.20	13.47	0.00
0.8	21500.70	0.02	21500.70	14.08	0.00
0.9	35987.70	0.05	35987.70	14.27	0.00
1.0	128009.00	0.00	128009.00	9.27	0.00

Table 2-8. Fifty-node uniform case: GADS-H and exact solution

	GADS-H		Exact Algorithm			Gap
P	Objective	Time (s)	Objective	Open Facilities	Time (s)	(%)
0.0	7197.27	0.02	7197.27	15 31 40 41 48	6.94	0.00
0.1	7763.80	0.02	7763.80	15 22 31 40 41 48	7.61	0.00
0.2	8425.99	0.03	8425.99	15 22 31 40 41 48	8.94	0.00
0.3	9275.99	0.02	9275.99	15 22 31 40 41 48	10.62	0.00
0.4	10253.90	0.02	10253.90	15 22 31 40 41 42 48	10.38	0.00
0.5	11603.00	0.03	11603.00	15 22 31 35 40 41 42 48	10.86	0.00
0.6	13416.80	0.06	13416.80	2 12 15 22 31 35 40 41 43 48	12.84	0.00
0.7	16157.20	0.09	16157.20	2 12 14 15 19 22 31 35 40 41 43 45 48	13.47	0.00
0.8	21500.70	0.20	21500.70	2 12 14 15 19 20 21 22 26 31 35 36 40 41 43 45 48	14.08	0.00
0.9	35987.70	0.48	35987.70	1 2 10 11 12 14 15 17 19 20 21 22 23 24 26 27 31 35 36 40 41 43 45 48 49	14.27	0.00
1.0	128009.00	0.00	128009.00	No open facility	9.27	0.00

It is interesting to compare the sets of open facilities in Table 2-8. One might conclude that more facilities should be open as the facilities get more vulnerable, that is, when the failure probability increases. Although this claim is usually valid, it is not always true. An extreme case is when the failure probability is 1 so that no facility should open. One can also consider the following counter example where there is only one single facility to open. If $f_1 + d_{11}r_1p_1 < d_{11}r_1$, then this facility should be open. If p_1 increases to p_1' such that $f_1 + d_{11}r_1p'_1 > d_{11}r_1$, then this facility should not be open. In this example, when the failure probability increases, fewer facilities should be open.

2.8.3 Genetic Algorithm Based Heuristic

In all the GA-H tests, the values in Table 2-9 were used for the parameters in GA-H described in Section 2.7.3.

Table 2-9. Values of the GA-H parameters

Parameter	Value
Population size N_P	100
Maximum number of generations N_G	200
Maximum number of generations without improvement N_M	100
Number of reproduction N_R	10
Number of immigration N_I	10
Mutation probability P_M	0.1

Table 2-10 lists the computational results of the 50-node dataset when the failure probability varies from 0 to 1. The first column, P, is the failure probability at each facility. Because the GA heuristic is a probabilistic method, two trials are performed. We report the minimum objective it obtained and the average CPU time (in seconds) in the second and third column respectively. The "gap" column is defined as the percentage difference between the cost of the solution obtained by GA and the optimal cost.

Table 2-10. Fifty-node uniform case: GA and exact solution

	GA Heuristic (2 Trials)		Exact Algorithm		
P	Min Objective	Time (s)	Objective	Time (s)	gap(%)
0.0	7197.27	5.41	7197.27	6.94	0.00
0.1	7763.80	5.23	7763.80	7.61	0.00
0.2	8425.99	5.20	8425.99	8.94	0.00
0.3	9275.99	5.11	9275.99	10.62	0.00
0.4	10253.90	5.34	10253.90	10.38	0.00
0.5	11603.00	5.66	11603.00	10.86	0.00
0.6	13416.80	5.52	13416.80	12.84	0.00
0.7	16157.20	5.75	16157.20	13.47	0.00
0.8	21500.70	7.95	21500.70	14.08	0.00
0.9	35987.70	9.83	35987.70	14.27	0.00
1.0	128009.00	3.25	128009.00	9.27	0.00

As we can see, the GA heuristic performs quite well. It can produce the exact solution in 2 trials. While there is no guarantee that the GA solutions are optimal, it can easily find an optimal or near optimal one by running multiple times then taking the best solution it finds. Considering the fast speed for each run, a multi-trial GA is quite attractive.

Figure 2-4 depicts the evolution of the minimum objective value, and the average objective value in each generation when $p = 0.5$. The algorithm terminates at generation 176 after it finds the optimal solution at generation 76. In fact, in this example the GA quickly converges to a close optimal solution after just 20 generations as shown in Figure 2-4.

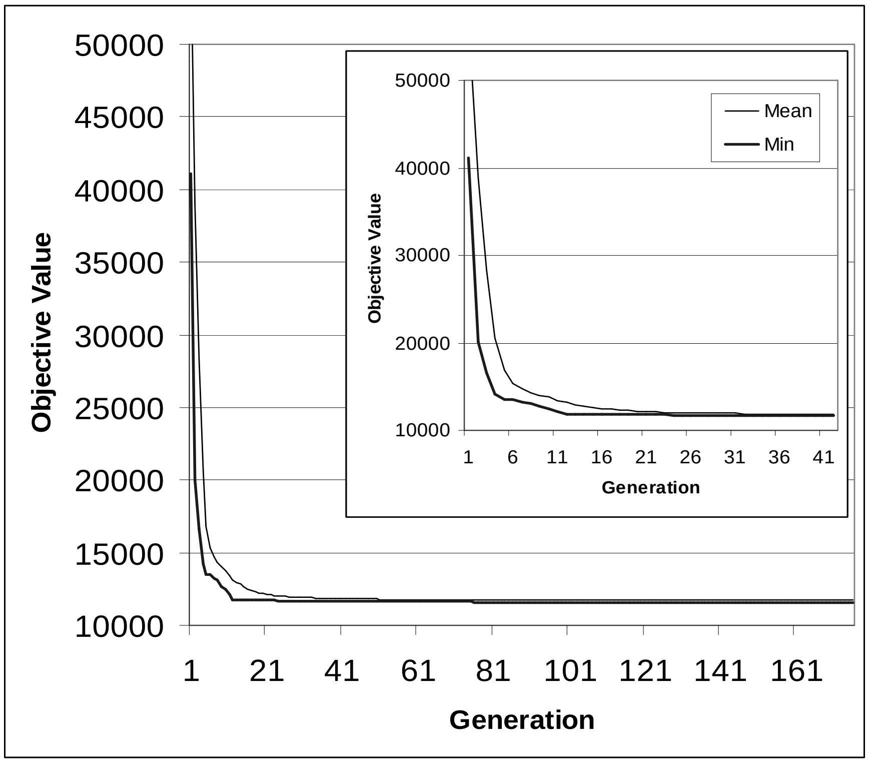


Figure 2-4. Evolution of the solutions from GA

2.8.4 Applying Heuristics to Solve URFLP-SFP

Dropping the uniformity assumption of facility failure probabilities introduces more challenges to solve URFLP exactly, due to the nonlinearity in formulation (URFLP-SFP).

The commercial solvers, such as CPLEX, lack such ability to solve nonlinear integer programming problems. In addition, other generic global optimization solvers seems not able to efficiently handle model URFLP-SFP. For example, BARON has difficulty in solving URFLP-SFP with 10 demand nodes and 10 candidate facilities. The major difficulties come from the binary constraints on the decision variables and the large number of nonlinear terms in model URFLP-SFP. In this section, we apply heuristics: SAA-H, GAD-H, GADS-H, and GA to solve such model and compare their performance. For some small size problems, we also provide globally optimal solutions from a simple enumeration method.

Several datasets are derived from the 100-node dataset in Appendix B: for example, dataset #10 is the first 10 lines from Table B-1 of Appendix B; it has 10 demand nodes and facility sites. The other datasets are derived in the same way. Table 2-11 lists the objective values obtained from SAA-H with different sample sizes. The column "Best Objective" lists the minimum objective value among the different sample sizes. Time is measured in seconds with results from the sample size of 100. "-" in Table 2-11 means that the program was out of memory due to the surge in the problem size. "-" in Table 2-12 means that the results of the enumeration method were not obtained due to the exponentially increased computational time. Table 2-12 summarizes the objective values of the solutions obtained by GAD-H, GADS-H, GA and the enumeration method, and their corresponding computational time. The results of GA are obtained through a single run.

Comparing the results from heuristics with the globally optimal solutions in small data sets (from #10, to #30), we can see that GADS-H and GA find optimal solutions, whereas SAA-H and GAD-H find optimal or near-optimal solutions. To evaluate the quality of the solutions found by these heuristics in all datasets, we plot the objective values in Figure 2-5. The values from SAA-H are the best ones available in Table 2-11 for each instance. Figure 2-5 shows that GADS-H and GA can find the best solutions in all datasets, whereas SAA-H and GAD-H can find either the best known solutions or

Table 2-11. Objective values obtained from SAA-H on URFLP-SFP with different sample sizes

Dateset #	N50	N100	N150	N200	Best Objective	Time (s) (N100)
10	5850.47	5576.00	5128.24	5128.24	5128.24	0.43
15	6074.06	5337.18	5337.18	5337.18	5337.18	2.11
20	8071.98	5761.79	5761.79	5761.79	5761.79	3.09
25	6749.39	6583.06	6583.06	6583.06	6583.06	12.14
30	7897.19	7622.22	7847.37	7847.37	7622.22	50.67
40	7474.92	7474.92	7474.92	7474.92	7474.92	54.20
50	8719.32	8641.28	8781.18	8641.28	8641.28	185.56
60	9357.37	9394.87	9357.37	9394.87	9357.37	159.48
70	10337.60	10391.80	10383.00	10383.00	10337.60	250.85
80	11054.30	11054.30	11054.30	11054.30	11054.30	320.82
90	12405.50	12405.50	12448.70	-	12405.50	659.57
100	13977.50	14028.10	-	-	13977.50	2164.03

Table 2-12. Computational performance on URFLP-SFP: GAD-H, GADS-H, GA and the enumeration method

Dateset	GAD-H		GADS-H		GA		Enumeration	
#	Objective	T (s)	Objective	T (s)	Objective	T (s)	Objective	T (s)
10	5128.04	0.00	5128.04	0.00	5128.04	1.62	5128.04	0.02
15	5305.04	0.00	5305.04	0.00	5305.04	1.81	5305.04	14.75
20	5761.79	0.00	5761.79	0.00	5761.79	1.81	5761.79	206.51
25	6439.87	0.00	6439.87	0.00	6439.87	1.82	6439.87	6806.57
30	7420.86	0.00	7382.04	0.00	7382.04	2.22	7382.04	188276.35
40	7474.92	0.00	7474.92	0.02	7474.92	4.09	-	-
50	8763.75	0.00	8641.28	0.00	8641.28	4.64	-	-
60	9357.37	0.00	9357.37	0.02	9357.37	5.14	-	-
70	10337.60	0.00	10337.60	0.03	10337.60	5.84	-	-
80	11054.30	0.00	11054.29	0.05	11054.29	5.56	-	-
90	13030.90	0.00	12405.50	0.06	12405.50	10.75	-	-
100	14463.40	0.02	13820.87	0.09	13820.87	11.23	-	-

close to the best known ones. In terms of computational efficiency, SAA-H takes much more time to achieve its solutions than all other three heuristics. Between GADS-H and GA, GADS-H spends considerably less CPU time than GA: GADS-H takes less than 0.1 seconds to get the best result in each instance. Overall, these results suggest that GADS-H is the best one among all four heuristics for model URFLP-SFP in terms of both solution quality and computational time.

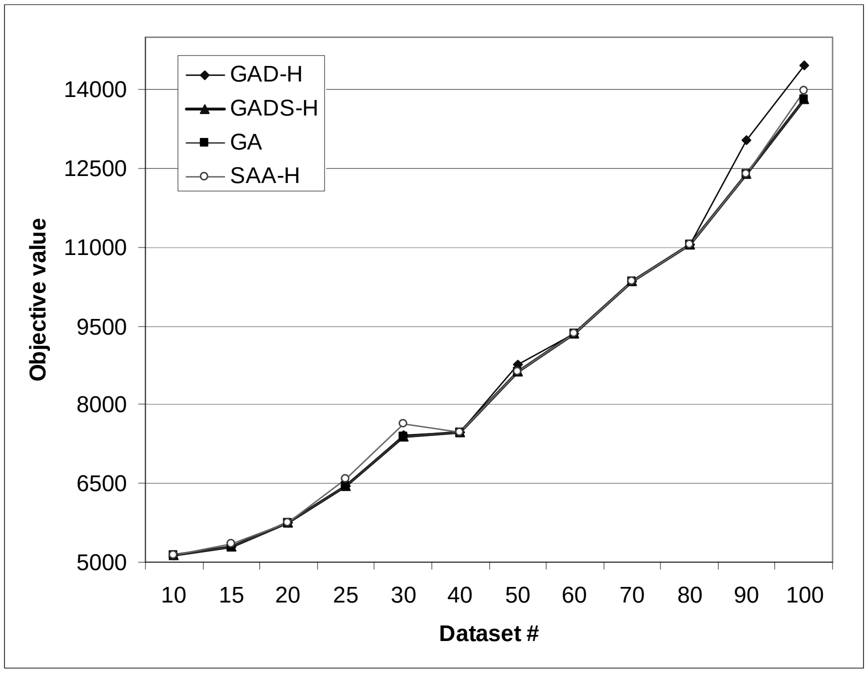


Figure 2-5. Comparison of objective values from GAD-H, GADS-H, GA, and SAA-H

2.8.5 URFLP-MFP: GADS-H vs. GA

In this part, we apply GADS-H and GA to solve URFLP-MFP, which is considerably more difficult than URFLP-SFP. The dataset is similar to the one used in the previous sections but with each facility having 3 levels of failure probability. The full dataset is presented in Table B-2 of Appendix B. The first half of Table 2-13 reports the results from GA, which has been run for 5 times with different random seeds. Eight datasets have been generated for testing. The best open sites and their optimal levels are listed in the second column. The best and worst results obtained in the 5 trials are listed in the third and fourth columns respectively. The average time in seconds are reported in the last column. The results of GADS-H are shown in the bottom half of Table 2-13. In only one case (the 80-node problem) does the GA find a better solution than GADS-H. In that case, the objective function value is 11782.85 compared to 11859.40, which represents only a 0.6% improvement. But GA takes more time than GADS-H does to get this small improvement. In all other cases, GADS-H finds the same solution as GA but with much less time. Overall, GADS-H is more favorable than GA in solving model URFLP-MFP.

Table 2-13. Computational results for URFLP-MFP using GADS-H and GA

Dataset #	GA, 5 Trials			
	Best Sites (Levels)	Best Result	Worst Result	Average Time (s)
20	2(2) 7(2)	5214.55	5214.55	3.13
30	2(2) 20(1) 21(2)	6484.74	6484.74	3.73
40	2(2) 20(1) 35(2)	7194.88	7194.88	4.52
50	2(2) 5(1) 20(1) 35(1)	8827.95	8827.95	5.71
60	2(2) 20(1) 35(2) 59(1)	9964.84	9964.84	7.20
70	2(2) 13(2) 20(1) 35(2)	10837.65	10837.65	9.06
80	2(2) 20(1) 35(2) 59(1) 76(1) 79(1)	11782.85	11867.01	10.81
90	2(2) 20(1) 35(2) 76(1) 79(1) 88(1)	12621.95	12717.84	11.13
100	2(2) 13(2) 20(1) 35(2) 67(2) 76(1) 88(1)	13713.42	13749.58	11.33
	GADS-H			
	Sites (Levels)	Result		Time (s)
20	2(2) 7(2)	5214.55		0.09
30	2(2) 20(1) 21(2)	6484.74		0.14
40	2(2) 20(1) 35(2)	7194.88		0.18
50	2(2) 5(1) 20(1) 35(1)	8827.95		0.27
60	2(2) 20(1) 35(2) 59(1)	9964.84		0.33
70	2(2) 13(2) 20(1) 35(2)	10837.65		0.39
80	2(2) 13(2) 20(1) 35(2) 76(1)	11859.40		0.57
90	2(2) 20(1) 35(2) 76(1) 79(1) 88(1)	12621.95		1.19
100	2(2) 13(2) 20(1) 35(2) 67(2) 76(1) 88(1)	13713.42		1.13

2.9 Conclusions

In this chapter, we have proposed several novel facility location models to deal with the uncertainty in facilities. The issue arises when a facility fails is that the customers originally assigned to it have to be reassigned to other facilities that are operational. The impact of such uncertainty is explicitly modeled in all of our models in order to build a reliable facility-customer network. In particular, we use a popular scenario-based technique to capture the uncertainty when the number of scenarios is relatively small. If the failure probability at each facility is independent, we propose several nonlinear integer models, URFLP-SFP and URFLP-MFP. These models greatly enrich the literature of facility reliability.

Four heuristics, SAA-H, GAD-H, GADS-H and GA, have been proposed to solve these problems. SAA-H is a specialized heuristic for the scenario-based model, whereas

GAD-H, GADS-H and GA can be used to solve URFLP-SFP and URFLP-MFP. Our computational studies show that (1) GADS-H is the best heuristics among all four in terms of the solution quality and the efficiency, and (2) GA is also able to find the best solution with a little more time than GADS-H.

There are several interesting research directions. We note that the major limitation of the current models is the assumption that the facilities are uncapacitated. Although the assumption itself is very common in the facility location models, it may be unrealistic in practice. In the capacitated case, customer of failed facilities can be assigned to the next level backup facilities only if they have sufficient capacity to satisfy the additional demand. This restriction may make the capacitated model very complex. It becomes a valuable topic of future investigation. In addition, some new measurements of the reliability concept in the facility location problem setting are worth pursuing.

CHAPTER 3
UNIFORM UNCAPACITATED RELIABLE FACILITY LOCATION PROBLEM: A 2.674-APPROXIMATION ALGORITHM

3.1 Introduction

In this chapter, we consider a special case of the uncapacitated reliable facility location problem in which failure probabilities are not facility-specific. We call it the uniform uncapacitated reliable facility location problem (UURFLP). Because the failure probabilities are the same across all facilities, URFLP is reduced to a linear integer programming problem. UURFLP is related to the model considered in [48], where the authors assume that some facilities are perfectly reliable while others are subject to failure with the same probability. They formulate their multi-objective problem as a linear integer program and propose a Lagrangian relaxation solution method. However, we consider penalty cost, a factor that is missed in [48].

UURFLP is clearly NP-hard as it generalizes UFLP. The focus of this chapter is to propose and analyze an approximation algorithm with a constant worst-case bound guarantee.

Designing approximation algorithms for the facility location problem and its variations has recently received considerable attentions from the research community. However, to the best of our knowledge, this chapter presents the first approximation algorithm for stochastic facility location problems with provider-side uncertainty.

The vast majority of approximation algorithms for the facility location problem mainly deal with deterministic problems, e.g. [17, 21, 32, 46]. Until very recently, approximation algorithms for UFLP with stochastic demand have been proposed; see the survey by Shmoys and Swamy [52]. Another related paper [6] proposes an approximation algorithm for a facility location problem with stochastic demand and inventory. Our approximation algorithm makes use of the ideas from several papers [17, 21, 32, 46]. In particular, this chapter is closely related to [17], which presents a 2.41-approximation algorithm for the so-called fault-tolerant facility location problem (FTFLP): every demand point must be served by several facilities where the number is specified, and a weighted linear combination is used to compute the connection costs. FTFLP has been motivated by the reliability issue considered in this chapter, but the failure probabilities of facilities are not explicitly modeled and penalty cost is not considered.

The remainder of this chapter is organized as follows. Several equivalent formulations for UURFLP are proposed in Section 3.2, which lead us to develop a 2.674-approximation algorithm in Section 3.3. The chapter is concluded in Section 3.4.

3.2 Formulations

The notation of this chapter follows that in Chapter 2. Recall the formulation of URFLP-SFP in Chapter 2:

(*URFLP-SFP*)

$$\begin{aligned}
\text{minimize} \quad & \sum_{i\in F} f_i y_i + \sum_{j\in D}\sum_{k=1}^{|F|}\sum_{i\in F} d_j c_{ij} x_{ij}^k (1-p_i) \prod_{l\in F} p_l^{\sum_{s=1}^{k-1} x_{lj}^s} \\
& + \textstyle\sum_{j\in D}\sum_{k=1}^{|F|+1}\prod_{l\in F} p_l^{\sum_{s=1}^{k-1} x_{lj}^s} d_j r_j z_j^k && (3\text{–}1)\\
\text{subject to} \quad & \sum_{i\in F} x_{ij}^k + \sum_{t=1}^{k} z_j^t = 1, \quad \forall j\in D, k=1,...,|F|+1 && (3\text{–}2)\\
& x_{ij}^k \le y_i, \quad \forall i\in F, j\in D, k=1,...,|F| && (3\text{–}3)\\
& \sum_{k=1}^{|F|} x_{ij}^k \le 1, \quad \forall i\in F, j\in D && (3\text{–}4)\\
& x_{ij}^k, z_j^k, y_i \in\{0,1\}. && (3\text{–}5)
\end{aligned}$$

Consider a special case of URFLP-SFP where all facilities have the same failure probabilities, i.e., $p_i = p, \forall i\in F$. This assumption simplifies formulation (URFLP-SFP) considerably based on the following observation. Because $p_i = p, \forall i \in F$, it is straightforward that $\prod_{l\in F} p_l^{\sum_{s=1}^{k-1} x_{lj}^s} = p^{k-1}$, which is independent of the values of x_{lj}^s. This property is implicitly used in a multi-objective formulation proposed in [48].

Based on the above observation, we are able to reduce formulation URFLP-SFP to a linear integer program as follows.

(*URFLP-IP*)

$$\text{minimize} \quad \sum_{i\in F} f_i y_i + \sum_{j\in D}\sum_{k=1}^{|F|}\sum_{i\in F} d_j c_{ij} x_{ij}^k (1-p)p^{k-1} + \sum_{j\in D}\sum_{k=1}^{|F|+1} p^{k-1} d_j r_j z_j^k \tag{3–6}$$

$$\text{subject to} \quad \sum_{i\in F} x_{ij}^k + \sum_{t=1}^{k} z_j^t = 1, \quad \forall j \in D, k = 1, ..., |F|+1$$

$$\sum_{k=1}^{|F|} x_{ij}^k \le y_i, \quad \forall i \in F, j \in D$$

$$x_{ij}^k, z_j^k, y_i \in \{0,1\}.$$

In the next section, we shall present an approximation algorithm for *URFLP-IP*. We find that it is more convenient to deal with a slightly different formulation. In the new formulation, we introduce a new set of decision variables θ_j^k to replace z_j^k. Define $\theta_j^k = \sum_{t=1}^k z_j^t$. Note that θ_j^k are not decision variables when $k > |F|$, $\theta_j^k = 1$ for all $k > |F|+1$. We prove that the following integer program is equivalent to formulation (URFLP-IP), as stated in Theorem 3.1. We refer the reader to Section 3.5 for the proof.

$$\text{minimize} \quad \sum_{i\in F} f_i y_i + \sum_{j\in D}\sum_{k=1}^{|F|}\sum_{i\in F} d_j c_{ij} x_{ij}^k p^{k-1}(1-p) + \sum_{j\in D}\sum_{k=1}^{\infty} d_j r_j \theta_j^k p^{k-1}(1-p) \tag{3–7}$$

$$\text{subject to} \quad \sum_{i\in F} x_{ij}^k + \theta_j^k = 1, \quad \forall j \in F, k = 1, ..., |F|$$

$$\sum_{k=1}^{|F|} x_{ij}^k \le y_i, \quad \forall i \in F, j \in D$$

$$x_{ij}^k, \theta_j^k, y_i \in \{0,1\}.$$

Theorem 3.1. *Formulation (3–6) and formulation (3–7) are equivalent.*

3.3 Approximation Algorithms

In this section, we aim to propose a 2.674-approximation algorithm for the special case where the failure probabilities are uniform. We call it a (R_f, R_c, R_p)-approximation algorithm, if the algorithm produces a solution with cost no more than

$$R_f F^* + R_c C^* + R_p P^*,$$

where F^*, C^*, and P^* are the optimal facility, transportation, and penalty cost, respectively.

We take advantage of several results for the fault-tolerant version of UFLP, where every demand point j must be served by k_j distinct facilities, a concept close to our level assignment. In [17], Guha *et al.* propose a couple of approximation algorithms for the fault-tolerant facility location problem using various rounding and greedy local-search techniques.

The fault-tolerant facility location problem can be formulated as the following integer program.

$(FTFLP)$

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in F} f_i y_i + \sum_{j \in D} \sum_{k=1}^{k_j} \sum_{i \in F} d_j c_{ij} w_j^k x_{ij}^k && (3\text{–}8) \\ \text{subject to} \quad & \sum_{i \in F} x_{ij}^k \geq 1, \quad \forall j \in D, k \leq k_j \\ & \sum_{k=1}^{k_j} x_{ij}^k \leq y_i, \quad \forall i \in F, j \in D \\ & x_{ij}^k, y_i \in \{0,1\}. \end{aligned}$$

The notation here follows that of model URFLP-IP with some subtle differences. w_j^k is the weighted factor at level k for demand node j. For demand j, the corresponding weights are assumed to be $w_j^1 \geq w_j^2 \geq \ldots \geq w_j^{k_j}$. x_{ij}^k denotes that demand j is assigned to facility i and facility i is the k^{th} closest open facility to j.

One of the key results on FTFLP from Guha *et al.* [17] is summarized below.

Lemma 3.1. *For any vector (x, y) satisfying the following inequalities (the dimension of (x, y) should be clear from the inequalities)*

$$\sum_{i \in F} x_{ij}^k \geq 1, \quad \forall j \in D, k \leq k_j$$

$$\sum_{k=1}^{k_j} x_{ij}^k \le y_i, \quad \forall i \in F, j \in D$$

$$x_{ij}^k \ge 0,$$

$$0 \le y_i \le 1,$$

and any $\alpha \in (0,1]$*, one can find an integer solution* $(\tilde{x}, \tilde{y})$ *satisfying the above inequalities so that*

$$\sum_{i\in F} f_i \tilde{y}_i + \sum_{j\in D}\sum_{k=1}^{k_j}\sum_{i\in F} w_j^k d_j c_{ij} \tilde{x}_{ij}^k \le \frac{\ln\frac{1}{\alpha}}{1-\alpha} f_i y_i + \frac{3}{1-\alpha}\sum_{j\in D}\sum_{k=1}^{k_j}\sum_{i\in F} w_j^k d_j c_{ij} x_{ij}^k.$$

In Lemma 3.1, α is a parameter that we can choose to control the quality of approximation. Indeed, the approximation ratios of the algorithms of Guha *et al.* are functions of α. One can then choose the best α to minimize the approximation ratio.

We are now ready to present our algorithm for UURFLP. We first solve a linear programming relaxation of formulation (3–7).

$$\begin{aligned}
\text{minimize} \quad & \sum_{i\in F} f_i y_i + \sum_{j\in D}\sum_{k=1}^{|F|}\sum_{i\in F} d_j c_{ij} x_{ij}^k p^{k-1}(1-p) + \sum_{j\in D}\sum_{k=1}^{\infty} d_j r_j \theta_j^k p^{k-1}(1-p) && (3\text{–}9)\\
\text{subject to} \quad & \sum_{i\in F} x_{ij}^k + \theta_j^k = 1, \quad \forall j \in D, k = 1, ..., |F| && (3\text{–}10)\\
& \sum_{k=1}^{|F|} x_{ij}^k \le y_i, \quad \forall i \in F, j \in D && (3\text{–}11)\\
& x_{ij}^k \ge 0, \theta_j^k \ge 0. && (3\text{–}12)
\end{aligned}$$

Assume that (x, y, θ) is an optimal solution to this linear program. Our algorithm rounds the fractional solution (x, y, θ) to an integer solution $(\bar{x}, \bar{y}, \bar{\theta})$ that is feasible to formulation (3–7).

The algorithm is based on a property of the optimal fractional solution (x, y, θ), which is formalized in the following lemma. This lemma enables us to utilize known algorithms and analysis for the fault-tolerant facility location problem.

Lemma 3.2. *For each* $j \in D$*, the following two statements are true.*

(i). $\theta_j^k \leq \theta_j^{k'}$ *for any* $1 \leq k \leq k'$.

(ii). If there exists k *such that* $0 < \theta_j^k < 1$, *then* $\theta_j^{k'} = 0$ *and* $\theta_j^{k''} = 1$ *for* $k' < k$ *and* $k'' > k$.

Proof. The proof is intuitive and similar to the proof of theorem 3.1, which is thus omitted here. □

We present our rounding procedure next. For each $j \in D$, assume k_j is the smallest integer such that $\theta_j^{k_j} > 0$.

The rounding procedure is carried out in two phases. The first phase rounds the optimal fractional solution (x, y, θ) to another fractional solution $(\hat{x}, \hat{y}, \hat{\theta})$, which is feasible to a linear programming relaxation of an appropriately defined fault-tolerant facility location problem. In the second phase, we use an algorithm for the fault-tolerant facility location problem to round the fractional solution $(\hat{x}, \hat{y}, \hat{\theta})$ to an integer solution $(\bar{x}, \bar{y}, \bar{\theta})$, which is feasible to formulation (3–7).

Phase I: Decomposition

- For every $j \in D$ and $i \in F$, let $\hat{\theta}_j^k = \theta_j^k$ and $\hat{x}_{ij}^k = x_{ij}^k$ for all $k \geq 1$ except for $k = k_j$.
- Choose a parameter $\delta \in (0, 1]$ whose value will be fixed later.
- Randomly generate a variable β that is uniformly distributed in $[0, \delta]$. For each $j \in D$ and $i \in F$, if $\theta_j^{k_j} \geq \beta$, then set

  $$\hat{\theta}_j^{k_j} = 1, m_j = k_j - 1, \hat{x}_{ij}^{k_j} = 0,$$

  otherwise set

  $$\hat{\theta}_j^{k_j} = 0, m_j = k_j, \hat{x}_{ij}^{k_j} = \frac{x_{ij}^{k_j}}{1 - \theta_j^{k_j}} = 1.$$

- For each $i \in F$, set

  $$\hat{y}^i = \max_{j \in D} \sum_{k=1}^{m_j} \hat{x}_{ij}^k.$$

Phase II: Solving Fault-Tolerant Problem

- For each $j \in D$ and $i \in F$, let $\bar{\theta}_j^k = 1$ and $\bar{x}_{ij}^k = 0$ for all $k > m_j$, and let $\bar{\theta}_j^k = 0$ for all $k \le m_j$, where m_j is defined in Phase I of the algorithm.
- Use the algorithm(s) in [17] with a parameter $\alpha \in (0, 1]$ to round the solution $(\hat{x}, \hat{y})$ to a feasible solution of a fault-tolerant facility location problem, where a set of facilities is open such that each client j is served by at least m_j distinct open facilities.
- For each $i \in F$, set $\bar{y}_i = 1$ if facility i is open, and set $\bar{y}_i = 0$ otherwise.
- For each $j \in D$, if i is the k^{th} closest open facility to client j, then let $\bar{x}_{ij}^k = 1$, where $k \le m_j$.
- Output the solution $(\bar{x}, \bar{y}, \bar{\theta})$.

This two-phase algorithm shall be referred to as Algorithm TP. It is obvious that the solution $(\bar{x}, \bar{y}, \bar{\theta})$ is feasible to formulation (3–7). We now establish a worst case approximation bound of our (randomized) algorithm, i.e., we shall show that the (expected) total cost is no more than a constant factor times the optimal cost. We bound the total penalty cost in Lemma 3.3, and bound the total facility and transportation costs in Lemma 3.4.

Lemma 3.3. *In Algorithm TP, the expected total penalty cost is bounded from above by*

$$\sum_{j\in D}\left(\frac{1}{\delta} d_j r_j p^{k_j-1}(1-p)\theta_j^{k_j} + \sum_{k=k_j+1}^{\infty} d_j r_j p^{k-1}(1-p)\theta_j^k\right).$$

Lemma 3.4. *For any $\alpha \in (0, 1)$, the expected facility cost plus the expected transportation cost is no more than*

$$\frac{\ln\frac{1}{1-\delta}}{\delta}\cdot\left(\frac{\ln\frac{1}{\alpha}}{1-\alpha}F_{LP} + \frac{3}{1-\alpha}C_{LP}\right),$$

where F_{LP} and C_{LP} are the total facility cost and the total transportation cost, respectively, corresponding to the solution (x, y, θ) in the linear programming relaxation of formulation (3–7).

We refer the reader to Section 3.5 for proofs of both lemmas. An immediate consequence of these two lemmas is the following corollary.

Corollary 3.1. *For any $\alpha \in (0,1)$ and $\delta \in (0,1)$, there is a $\left(\frac{\ln\frac{1}{1-\delta}}{\delta}\frac{\ln\frac{1}{\alpha}}{1-\alpha}, \frac{\ln\frac{1}{1-\delta}}{\delta}\frac{3}{1-\alpha}, \frac{1}{\delta}\right)$-approximation algorithm for the problem with uniform probabilities.*

By choosing $\alpha = 0.049787057$, $\delta = 0.271478197$, we derive that $\max\left(\frac{\ln\frac{1}{1-\delta}}{\delta}\frac{\ln\frac{1}{\alpha}}{1-\alpha}, \frac{\ln\frac{1}{1-\delta}}{\delta}\frac{3}{1-\alpha}, \frac{1}{\delta}\right) <$ 3.6836, which leads to the main result of this section.

Theorem 3.2. *The uniform case of URFLP admits a* 3.6836*-approximation algorithm.*

We use a technique called greedy improvement procedure (see [32] for details) to further improve the approximation factor.

Phase III: Greedy Improvement

- Apply Phase I and Phase II to an instance of the original problem where the facility cost is scaled up by a given factor $\Delta \geq 1$, and output a feasible solution.
- Assume the costs of the current solution are (F, C, P). Pick a facility i with cost f_i such that the ratio
$$(C + P - C_i - P_i - f_i)/f_i$$
is maximized, where C_i and P_i are the corresponding transportation cost and penalty cost if facility i was added to the current solution. If the ratio is positive, open facility i and repeat this step; otherwise, stop.

The greedy improvement procedure can improve the worst case bound of Algorithm TP, as shown next. We omit the proof here as the analysis is very similar to those in [17] and [32].

Lemma 3.5. *For any given (R_f, R_c, R_p)-approximation algorithm for (3–7), there is a $(R_f + \ln(\Delta), 1 + \frac{R_c-1}{\Delta}, 1 + \frac{R_p-1}{\Delta})$-approximation algorithm.*

By choosing $\alpha = 0.42539606, \delta = 0.17430753$, and $\Delta = 2.82899675$, we obtain the following theorem.

Theorem 3.3. *URFLP with uniform failure probabilities admits a* 2.674*-approximation algorithm.*

3.4 Conclusions

In this chapter, we employ various rounding and decomposition techniques to develop a 3.6836-approximation algorithm, then improve this to a 2.674-approximation algorithm using greedy improvement procedures for the uniform uncapacitated reliable facility location problem. To the best of our knowledge, these are the first type of approximation algorithm for the facility location problem with uncertainties in the facility side. Whether these bounds can be improved is an open problem for future research. Another interesting topic is to develop an approximation algorithm for the uniform capacitated reliable facility location problem.

3.5 Proofs

Proof. Theorem 3.1.

(1). Assume we have an optimal solution (x, y, z) for problem (3–6), then we can construct a feasible solution $(\bar{x}, \bar{y}, \theta)$ for problem (3–7) with the same objective value.

Let $\bar{x} = x$ and $\bar{y} = y$. For each $j \in D$ and $k \geq 1$, define $\theta_j^k = \sum_{t=1}^{k} z_j^t$. Because for each j, there is exactly one value of z_j^t would be equal to 1, we can conclude $\theta_j^k \in \{0, 1\}$. And it is straightforward that $(\bar{x}, \bar{y}, \theta)$ is a feasible solution to problem (3–7). Now we check the objective value corresponding to $(\bar{x}, \bar{y}, \theta)$. In fact, we only need to consider

$$\begin{aligned}
\sum_{k=1}^{\infty} p^{k-1}(1-p)\theta_j^k &= \sum_{k=1}^{\infty} p^{k-1}(1-p) \sum_{t=1}^{\min\{k,|F|+1|\}} z_j^t \\
&= \sum_{t=1}^{|F|+1} \sum_{k=t}^{\infty} p^{k-1}(1-p) z_j^t \\
&= \sum_{t=1}^{|F|+1} p^{t-1} z_j^t
\end{aligned}$$

The first equality is due to the fact that $\forall t > |F| + 1, j \in D, z_j^t = 0$.

(2). Now we prove the other direction; i.e., assume we have an optimal solution (x, y, θ) for problem (3–7), then we can construct a feasible solution $(\bar{x}, \bar{y}, z)$ for problem (3–6) with the same objective value.

To that end, we first show it is without losing of generality to assume that for each $j \in D$, if $\theta_j^k = 1$, then $\theta_j^{k+1} = 1$. Otherwise, assume there exist j_0 and k_0 such that $\theta_{j_0}^{k_0} = 1$ but $\theta_{j_0}^{k_0+1} = 0$. Then from the constraints, we know that $x_{ij_0}^{k_0} = 0$ for all i and there exists an i_0 such that $x_{i_0j_0}^{k_0+1} = 1$.

Now we define a new solution $(\tilde{x}, \tilde{y}, \tilde{\theta})$ for problem (3–7) as follows: $\tilde{y} = y$; for $j \neq j_0$, $\tilde{x}_{ij}^k = x_{ij}^k$, $\tilde{\theta}_j^k = \theta_j$ for all $i \neq i_0$ and k; for $k \neq k_0, k_0+1$, $\tilde{x}_{ij_0}^k = x_{ij_0}^k$, $\tilde{\theta}_{j_0}^k = \theta_{j_0}$ for all $i \neq i_0$; $x_{i_0j_0}^{k_0} = 1$, $\theta_{j_0}^{k_0} = 0$, $x_{i_0j_0}^{k_0+1} = 0$ and $\theta_{j_0}^{k_0+1} = 1$. It is easy to verify that $(\tilde{x}, \tilde{y}, \tilde{\theta})$ is a feasible solution for problem (3–7). The difference between the objective value corresponding to (x, y, θ) and the objective value corresponding to $(\tilde{x}, \tilde{y}, \tilde{\theta})$ is

$$\left(d_{j_0}c_{i_0j_0}p^{k_0}(1-p) + d_{j_0}r_{j_0}p^{k_0-1}(1-p)\right) - \left(d_{j_0}c_{i_0j_0}p^{k_0-1}(1-p) + d_{j_0}r_{j_0}p^{k_0}(1-p)\right)$$

which should be less than or equal to zero given that (x, y, θ) is an optimal solution to problem (3–7). It follows that

$$r_{j_0} \leq c_{i_0j_0}.$$

If we define yet another solution $(\hat{x}, \hat{y}, \hat{\theta})$ such that it is the same as (x, y, θ) with the following exceptions:

$$\hat{x}_{i_0j_0}^{k_0} = 0, \hat{\theta}_{j_0}^{k_0} = 1, \hat{x}_{i_0j_0}^{k_0+1} = 0, \hat{\theta}_{j_0}^{k_0+1} = 1.$$

Again, this solution it feasible. Also, the difference between the objective value corresponding to (x, y, θ) and the objective value corresponding to $(\hat{x}, \hat{y}, \hat{\theta})$ is

$$d_{j_0}c_{i_0j_0}p^{k_0}(1-p) - d_{j_0}r_{j_0}p^{k_0}(1-p) \geq 0,$$

which implies that $(\hat{x}, \hat{y}, \hat{\theta})$ is an optimal solution for problem (3–7) and it satisfies the condition that if $\hat{\theta}_{j_0}^{k_0} = 1$ then $\hat{\theta}_{j_0}^{k_0+1} = 1$.

Therefore, we can assume that for each $j \in D$, if $\theta_j^k = 1$, then $\theta_j^{k+1} = 1$. Now we are ready to find a feasible solution for problem (3–6). Let $\bar{x} = x, \bar{y} = y$. And for each j, if $\theta_j^k = 1$ and $\theta_j^{k-1} = 0$, then let $z_j^k = 1$ and let $z_j^{k'} = 0$ for $k' \neq k$; if $\theta_j^1 = 1$, then let $z_j^1 = 1$ and $z_j^{k'} = 0$ for $k' \geq 2$. Then it is clear from the definition that $\theta_j^k = \sum_{t=1}^k z_j^t$. Then we know that $(\bar{x}, \bar{y}, z)$ is feasible for problem (3–6). And from the first part of the proof, we know the objective value corresponding to $(\bar{x}, \bar{y}, z)$ is the same as the one corresponding to (x, y, θ). This completes the proof. $\square$

Proof. Lemma 3.3.

We bound the expected penalty cost for each client $j \in D$, which depends on the value of β that is randomly generated. We consider two cases.

Case 1. $\theta_j^{k_j} \geq \beta$. Then client j is assigned to exactly $m_j = k_j - 1$ facilities. Therefore, a penalty cost will incur if all of these $(k_j - 1)$ facilities failed, which happens with probability p^{k_j-1}. Therefore, the expected total penalty cost for client j is $d_j r_j p^{k_j-1}$. Notice that $\theta_j^k = 1$ for all $k \geq k_j + 1$. Then

$$\begin{aligned} d_j r_j p^{k_j-1} &= \sum_{k=k_j}^{|F|} d_j r_j p^{k-1}(1-p) + \sum_{k=|F|+1}^{\infty} d_j r_j p^{k-1}(1-p) \\ &= d_j r_j p^{k_j-1}(1-p) + \sum_{k=k_j+1}^{\infty} d_j r_j p^{k-1}(1-p)\theta_j^k \end{aligned}$$

Case 2. $\theta_j^{k_j} < \beta$. In this case, client j is assigned to exactly $m_j = k_j$ facilities. A penalty cost will incur if all of these k_j facilities failed, which happens with probability p^{k_j}. Therefore, the expected total penalty cost for client j is $d_j r_j p^{k_j}$, which is equal to

$$\sum_{k=k_j+1}^{\infty} d_j r_j p^{k-1}(1-p) = \sum_{k=k_j+1}^{\infty} d_j r_j p^{k-1}(1-p)\theta_j^k.$$

If $\theta_j^{k_j} \geq \delta$, then Case 1 happens with probability 1. Thus the expected penalty cost of client j is

$$d_j r_j p^{k_j-1}(1-p) + \sum_{k=k_j+1}^{\infty} d_j r_j p^{k-1}(1-p)\theta_j^k \leq \frac{1}{\delta} d_j r_j p^{k_j-1}(1-p)\theta_j^{k_j} + \sum_{k=k_j+1}^{\infty} d_j r_j p^{k-1}(1-p)\theta_j^k.$$

On the other hand, if $\theta_j^{k_j} < \delta$, then Case 1 happens with probability $\frac{\theta_j^{k_j}}{\delta}$, and thus Case 2 happens with probability $1 - \frac{\theta_j^{k_j}}{\delta}$. Therefore, the expected penalty cost of client j is bounded by

$$\begin{aligned}
&\frac{\theta_j^{k_j}}{\delta}\left(d_j r_j p^{k_j-1}(1-p)+\sum_{k=k_j+1}^{\infty} d_j r_j p^{k-1}(1-p)\theta_j^k\right)+(1-\frac{\theta_j^{k_j}}{\delta})\sum_{k=k_j+1}^{\infty} d_j r_j p^{k-1}(1-p)\theta_j^k \\
=\;& \frac{1}{\delta} d_j r_j p^{k_j-1}(1-p)\theta_j^{k_j}+\sum_{k=k_j+1}^{\infty} d_j r_j p^{k-1}(1-p)\theta_j^k.
\end{aligned}$$

This completes the proof. □

Proof. Lemma 3.4.

The costs of interest depend on the value of the randomly generated β. Recall the definition of $(\hat{x},\hat{y})$. For each $j\in D$ and $i\in F$, $\hat{x}_{ij}^k = x_{ij}^k$, if $k\neq m_j$; If $m_j = k_j - 1$, then $\hat{x}_{ij}^{k_j} = 0 \le x_{ij}^{k_j}$; if $m_j = k_j$, then $\hat{x}_{ij}^{k_j} = \frac{x_{ij}^{k_j}}{1-\theta_j^{k_j}}$. The latter is true only if $\theta_j^{k_j} < \beta$. Therefore, in both cases,

$$\hat{x}_{ij}^k \le \frac{1}{1-\beta} x_{ij}^k. \tag{3–13}$$

By the constraint of the linear programming relaxation of (3–7), we know that, for each $i\in F$ and $j\in D$,

$$\sum_{k=1}^{|F|} x_{ij}^k \le y_i.$$

Therefore,

$$\hat{y}_i = \max_{j\in D}\sum_{k=1}^{k_j}\hat{x}_{ij}^k \le \max_{j\in D}\frac{1}{1-\beta}\sum_{k=1}^{k_j} x_{ij}^k \le \frac{1}{1-\beta} y_i. \tag{3–14}$$

To show that

$$\sum_{i\in F}\hat{x}_{ij}^k = 1 \quad \forall j\in D, k\le m_j,$$

we consider the following two cases.

- $m_j = k_j - 1$: In this case, $\forall j\in D, k\le m_j$, $\theta_j^k = 0$ and $\hat{x}_{ij}^k = x_{ij}^k$. From the constraint 3–10, $\sum_{i\in F}\hat{x}_{ij}^k = \sum_{i\in F} x_{ij}^k = 1$.

- $m_j = k_j$: when $k < k_j$, from the proof of the first bullet, we know that $\sum_{i\in F} \hat{x}_{ij}^k = 1$. when $k = m_j$, $\sum_{i\in F} \hat{x}_{ij}^{k_j} = \frac{\sum_{i\in F} x_{ij}^k}{1-\theta_j^{k_j}} = 1$. The last equality holds because of the constraint 3–10, i.e. $\sum_{i\in F} x_{ij}^k + \theta_j^{k_j} = 1$.

  Thus $\hat{x}$ and $\hat{y}$ satisfy the following constraints:

$$\sum_{i\in F} \hat{x}_{ij}^k = 1 \quad \forall j \in D, k \le m_j$$

$$\sum_{k=1}^{m_j} \hat{x}_{ij}^k \le \hat{y}_i \quad \forall i \in F, j \in D$$

$$\hat{x}_{ij}^k \ge 0,$$

$$0 \le \hat{y}_i \le 1$$

.

Therefore, if we apply the algorithm in [17] with a parameter α to round the solution $(\hat{x}, \hat{y})$, then by Lemma 3.1, we can construct a solution $(\bar{x}, \bar{y})$ such that the expected total facility cost and total transportation cost is bounded above by

$$\begin{aligned}
&\frac{\ln\frac{1}{\alpha}}{1-\alpha}\sum_{i\in F} f_i\hat{y}_i + \frac{3}{1-\alpha}\sum_{j\in D}\sum_{k=1}^{m_j}\sum_{i\in F} d_j c_{ij}\hat{x}_{ij}^k p^{k-1}(1-p) \\
\le\ & \frac{1}{1-\beta}\left(\frac{\ln\frac{1}{\alpha}}{1-\alpha}\sum_{i\in F} f_i y_i + \frac{3}{1-\alpha}\sum_{j\in D}\sum_{k=1}^{m_j}\sum_{i\in F} d_j c_{ij} x_{ij}^k p^{k-1}(1-p)\right) \\
\le\ & \frac{1}{1-\beta}\left(\frac{\ln\frac{1}{\alpha}}{1-\alpha}\sum_{i\in F} f_i y_i + \frac{3}{1-\alpha}\sum_{j\in D}\sum_{k=1}^{k_j}\sum_{i\in F} d_j c_{ij} x_{ij}^k p^{k-1}(1-p)\right).
\end{aligned}$$

The first inequality holds because of inequalities 3–13 and 3–14; the second one holds because of $m_j \le k_j$.

Finally notice that β was uniformly distributed in $(0, \delta)$, thus

$$\mathrm{E}[\frac{1}{1-\beta}] = \int_0^\delta \frac{1}{1-\beta}\frac{1}{\delta} d\beta = \frac{1}{\delta}\ln\frac{1}{1-\delta}.$$

This completes the proof of the Lemma. □

CHAPTER 4
SYSTEM RELIABILITY OPTIMIZATION AND MONOTONIC OPTIMIZATION

4.1 Introduction

The performance reliability of a system is of utmost importance in many industrial and military systems. System reliability is a measure of how well a system meets its design objective, and is usually expressed in terms of the reliability (a probability of successful operations) of the subsystems or components. For example, a series system works if and only if every component works. Such a system fails whenever any component fails. The reliability of the series system in Figure 4-1 is

$$R_s = \prod_{i=1}^{n} r_i,$$

where R_s is the system reliability and r_i is the reliability of component i, which is the probability that component i successfully operates during the intended period of time. In this chapter, we assume all components operate independently.

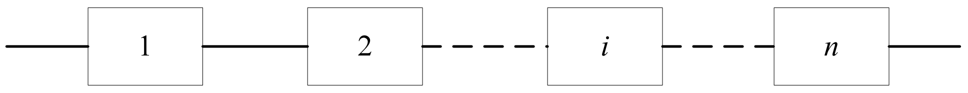


Figure 4-1. Series system

System reliability can be improved in various ways, such as physical enhancement of component reliability, provision of redundant components in parallel, and allocation of interchangeable components.

Unlike in a series system, in a parallel system, not all components are necessary for the system to work successfully. Actually, only one component in such system needs to work properly in order for the whole system to work properly. Including n components when only one is essential is called redundancy. The other $n-1$ components are included to increase the probability that there is at least one working component. Redundancy is a widely used technique in engineering to enhance system reliability.

For instance, in the above series system, each component i can be enhanced to a subsystem i with x_i identical components in parallel. In this chapter, we use *subsystem* and *stage* interchangeably. The reliability of such subsystem i is denoted as R_i. And because only one component is required to make the system perform properly and all components are independent, $R_i = 1 - (1 - r_i)^{x_i}$. Therefore the improved system reliability R_s is the following.

$$R_s(x_1, \ldots, x_n) = \prod_{i=1}^{n} R_i = \prod_{i=1}^{n} [1 - (1 - r_i)^{x_i}],$$

which is increasing in each x_i. The improved system is a parallel-series one as shown in Figure 4-2.

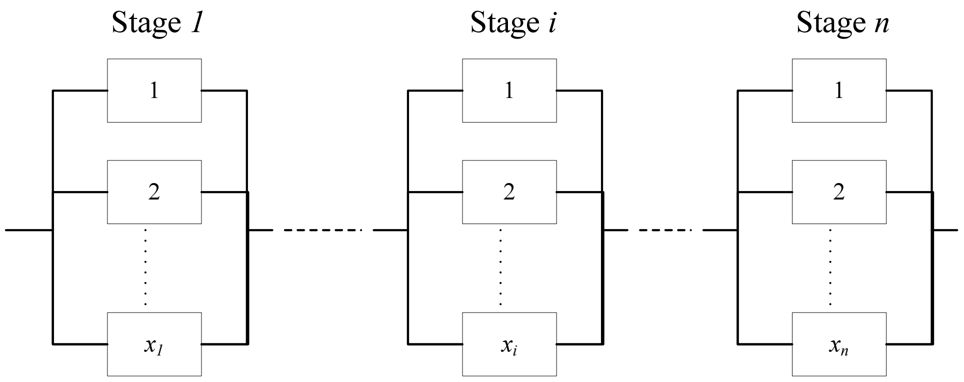


Figure 4-2. Parallel-series system

The system reliability can be maximized by choosing the right combination of x_i under certain resource constraints, denoted by $g_j(\cdot)$, $j = 1, \ldots, m$ in this chapter. This leads to a redundancy allocation optimization problem (RAOP) or a nonlinear integer programming in general.

$$(RAOP): \quad \max \quad R_s = f(x_1, \ldots, x_n) \tag{4–1}$$

$$\text{subject to} \quad g_j(x_1, \ldots, x_n) \le c_j, \quad \forall j = 1, \ldots, m, \tag{4–2}$$

$$0 \le l_i \le x_i \le u_i, \quad \forall i = 1, \ldots, n, \tag{4–3}$$

$$x_i\text{: integer}, \quad \forall i = 1, \ldots, n, \tag{4–4}$$

where $f(\cdot)$ is a general expression of the system reliability.

Another way to increase the system reliability is to simply use more reliable components, which certainly costs more in terms of various resources. This problem is called a reliability allocation optimization problem ([28]). Suppose there are u_i discrete choices for component reliability at stage i for $i = 1, \ldots, k$ $(\leq n)$ and the choice for component reliability at stage $k+1, \ldots, n$ is on a continuous scale. Let $r_i(1), r_i(2), \ldots, r_i(u_i)$ denote the component reliability choices at stage i for $i = 1, \ldots, k$ $(\leq n)$. Then the continuous/discrete reliability allocation optimization problem can be formulated as follows:

$$\begin{aligned} \max \quad & R_s = f(r_1(x_1), \ldots, r_k(x_k), r_{k+1}, \ldots, r_n) && (4\text{–}5) \\ \text{subject to} \quad & g_j(r_1(x_1), \ldots, r_k(x_k), r_{k+1}, \ldots r_n) \leq c_j, \quad \forall j = 1, \ldots, m, && (4\text{–}6) \\ & x_i \in \{1, 2, \ldots, u_i\} \quad \forall i = 1, \ldots, k, && (4\text{–}7) \\ & 0 \leq r_i^l \leq r_i \leq u_i^u, \quad \forall i = k+1, \ldots, n, && (4\text{–}8) \end{aligned}$$

where r_i^l and u_i^u are the lower bound and upper bound respectively for the component reliability at stage i. If $k = 0$, the above mixed integer nonlinear programming formulation reduces to a pure nonlinear programming problem.

The systems we are interested in are not limited to series and parallel systems. They can be complex (general) systems that are non-series and non-parallel, such as the bridge network in Figure 4-3. The system reliability of such system can be computed by the conditional probability theory. For example, the system reliability of the five-component bridge network in Figure 4-3 can be computed based on whether component 5 is functional or not. We refer readers to Appendix A for the details of the following expression.

$$R_s = (r_1 + r_3 - r_1 r_3)(r_2 + r_4 - r_2 r_4) r_5 + (r_1 r_2 + r_3 r_4 - r_1 r_2 r_3 r_4)(1 - r_5).$$

If each component i is enhanced to a subsystem i with x_i identical components in parallel, the objective function of the redundancy allocation optimization problem is

$$R_s(x_1,\ldots,x_n) = (R_1+R_3-R_1R_3)(R_2+R_4-R_2R_4)R_5+(R_1R_2+R_3R_4-R_1R_2R_3R_4)(1-R_5).$$

Note that $R_i = 1-(1-r_i)^{x_i}$.

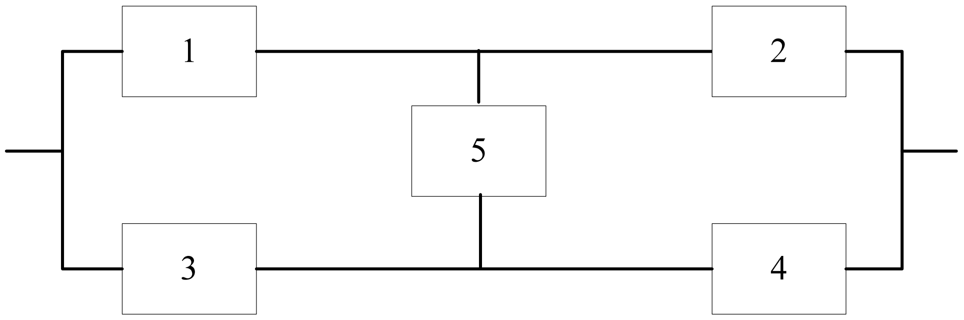


Figure 4-3. Five-component bridge network

Constraints of a system can be the total weight, the total cost, the total volume, and so on. In general, such constraints are in nonlinear forms [28]. As the number of components in each subsystem/stage increases, more connecting equipment is required and thus, the cost and weight may increase exponentially [53].

Next, we present a more general formulation for the system reliability optimization problem (SROP), which includes the redundancy allocation optimization and the reliability allocation optimization problems as special cases. Formally, SROP can be formulated as a nonlinear mixed integer programming problem.

$$(SROP): \quad \max \quad R_s = f(x_1,\ldots,x_q; r_1,\ldots,r_p) \tag{4–9}$$

$$\text{subject to} \quad g_j(x_1,\ldots,x_q; r_1,\ldots,r_p) \le c_j, \quad \forall j=1,\ldots,m, \tag{4–10}$$

$$0 \le l_i \le x_i \le u_i, \quad \forall i=1,\ldots,q, \tag{4–11}$$

$$0 \le r_k^l \le r_k \le r_k^u, \quad \forall k=1,\ldots,p, \tag{4–12}$$

$$x_i\text{: integer}, \quad \forall i=1,\ldots,q, \tag{4–13}$$

where x_i is the total number of parallel components at stage i, $i = 1, \ldots, q$; r_k is the reliability level at stage k, $k = 1, \ldots, p$; q is not necessarily equal to p, because in some stages, one may just have only one option to choose: allocating either redundancy or reliability but not both. If only redundancy allocation is allowed at stage i, then x_i is the only decision variable at stage i; If only reliability allocation is allowed at stage i, then r_i is the only decision variable at stage i; If both redundancy and reliability allocations are allowed at stage i, then both x_i and r_i are the decision variables at stage i. Without loss of the generality, we assume $q + p = n$. $f(\cdot)$ is the function measuring the system reliability that is nondecreasing in each of its variables; $g_j(x, r)$ is the consumption of resource j, $j = 1, \ldots, m$. Naturally, $g_j(x, r)$ is assumed to be nondecreasing in x and r.

The SROP model covers a vast majority of reliability optimization models discussed in the literature. For example, model SROP reduces to model RAOP when $p = 0$, and a continuous version of the reliability allocation optimization problem when $q = 0$. It certainly can also model the general case of the reliability allocation optimization problem by reinterpreting the definition of the variables, since objective functions 4–5 and 4–9 are mathematically equivalent, so are constraints 4–6 and 4–10. In addition, model SROP is obviously a reliability-redundancy allocation model if $p = q$, where at each stage the decisions are which component reliability to choose and how much redundancy as well.

The SROP model has received tremendous research attentions over decades and has been extensively studied and solved using many different mathematical programming techniques and heuristic approaches. Kuo et al. [28], along with [27], provide a detailed introduction to the models and algorithms in the reliability optimization. SROP is often characterized by a nonlinear objective function that is neither convex nor concave over a nonconvex feasible region. Due to the extreme difficulty of such type of problem, the solution methods in the literature are mainly heuristics, meta-heuristics and approximation algorithms. A comprehensive review on these methods can be found in [27].

In this chapter, we are interested in the exact methods for the reliability optimization problems, in particular the branch-and-bound schemes ([37], [28], [51], [29]). The branch-and-bound algorithm is a popular approach for finding global optimal solutions for the nonconvex problems. They differ in (i) methods of selecting a branching variable, (ii) methods of selecting a branching node, (iii) methods of calculating upper and lower bounds. For example, [28] provides a way to transform the integer variables into binary variables that are branched on afterwards. [37] branches on the variables in increasing order of the range, the difference between upper bound and lower bound, then fixes them in that order. [51] and [29] adopt a convexification method to transform the problem into a convex maximization problem during the process of the branch-and-bound. The branching is done on a fractional variable. [50] follows a similar theme to [51] and [29] except that the branch is done on the selected boundary points and it only solves the continuous version of the monotone optimization problem.

The Monotonic Branch-Reduce-Bound (briefly, mBRB) algorithm presented in this chapter follows the development of a specialized algorithm for monotonic optimization proposed by Tuy and his collaborators in a series of papers, [40], [56], [57]. They use a union of hyperrectangles to cover the boundary of the feasible region. The hyperrectangle is called a polyblock in their language so that the algorithm is called a polyblock algorithm. A branch-and-bound implementation of the polyblock algorithm was mentioned in [57] for the continuous version of the monotone optimization problem. The efficiency of this approach is reported on various classes of global optimization problems, such as polynomial fractional programming [59], and discrete nonlinear programming [58].

Compared to the previous branch-and-bound algorithms, the proposed method exploits the monotonicity properties inherent in SROP without requiring any convexity, concavity, differentiability, and separability. Unlike [28], it does not require conversion of the original decision variables into binary ones. The method is efficient and flexible enough to solve pure, mixed-integer, and integer nonlinear programming problems arising from reliability optimization. It appears to be the first of its kind in the literature on reliability optimization.

The remainder of this chapter is organized as follows. The monotonic branch-reduce-bound algorithm is presented in Section 4.2 with its convergence analysis. Several convergence acceleration techniques are also discussed. The algorithm is applied to solve both the reliability allocation and the reliability-redundancy allocation optimization problems in Section 4.3 with a demonstration of its efficiency. The chapter is concluded in Section 4.4.

4.2 A Monotonic Branch-Reduce-Bound Algorithm

To facilitate the presentation of the monotonic branch and bound algorithm, we recast the original SROP, i.e. (4–9), to a more general vector format by denoting $x = (x_1, \ldots, x_q; r_1, \ldots, r_p)$, in other words, $x_{q+i} = r_i$, $\forall i = 1, \ldots, p$.

$$\begin{aligned}
(\mathit{SROP'}):\quad & \max \quad && R_s = f(x) && \text{(4–14)}\\
& \text{subject to} \quad && x \in \mathcal{G} = \{x |\, g_j(x) \le c_j, \quad \forall j = 1, \ldots, m\}, && \text{(4–15)}\\
& && x \in [x^L, x^U], && \text{(4–16)}\\
& && x \in \mathcal{X}_z = \{x |\, x \in \mathcal{R}^n,\ x_i\text{: integer}, \quad \forall i = 1, \ldots, q\}, && \text{(4–17)}
\end{aligned}$$

where $[x^L, x^U]$ denotes a hyperrectangle with lowest boundary x^L and greatest boundary x^U; functions f and g_i are nondecreasing for $j = 1, \ldots, m$. To ensure $\mathcal{G}$ is closed, we assume g_j are semi-continuous for $j = 1, \ldots, m$.

Since no concavity assumption has been made on the objective function, multiple locally optimal solutions may exist. However, from the monotonicity of the functions f and g_i, the following proposition can be easily derived.

Proposition 4.1. *[56] The global maximum of $f(x)$ over $\mathcal{G} \cap \mathcal{X}_z \cap [x^L, x^U]$, if it exists, is attained on its boundary.*

The property of Proposition 4.1 is used by Tuy to develop the original Polyblock Algorithm for monotonic optimization in [56]. The branch-reduce-bound implementation of the polyblock algorithm was mentioned in [57] for the continuous version of the monotone optimization problem. The algorithm recursively partitions the hyperrectangle $[x^L, x^U]$ into a smaller region that still contains the global maximizer. A variant of the algorithm is able to handle the mixed integer version of SROP'.

In the description of Algorithm 1, $\mathcal{S}$ denotes a hyperrectangle partition; $\mathfrak{L}$ is a list of unfathomed hyperrectangles; ϵ is a pre-defined optimality tolerance parameter; x_{best} and f_{best} denote the current best solution and objective value respectively; $UB(\mathcal{S})$ is the upper bound of the objective function over $\mathcal{S}$. Besides initialization, the major steps are described in detail as follows.

```
Initialization
 1: $x^L \leftarrow (\lceil x_1^L \rceil, \dots, \lceil x_q^L \rceil, x_{q+1}^L, \dots, x_n^L)$, $x^U \leftarrow (\lfloor x_1^U \rfloor, \dots, \lfloor x_q^U \rfloor, x_{q+1}^U, \dots, x_n^U)$
 2: if $x^U \in \mathcal{G}$ then
 3:     $x^U$ is the optimal solution, terminate;
 4: else if $x^L \notin \mathcal{G}$ then
 5:     the problem is infeasible, terminate;
 6: else
 7:     set $\mathcal{S} = [x^L, x^U], \epsilon > 0, \mathfrak{L} = \{\mathcal{S}\}$, $x_{best} = x^L$, $f_{best} = f(x^L)$
 8: end if
Select and Branch
 9: if $\mathfrak{L} = \emptyset$ then
10:     output the current $x_{best}$ as the solution, terminate;
11: else
12:     select $\mathcal{S} = [s^L, s^U] \in \mathfrak{L}$ such that $UB(\mathcal{S}) = \max_{S \in \mathfrak{L}}\{UB(S)\}$ and $\mathfrak{L} \leftarrow \mathfrak{L} \backslash \{\mathcal{S}\}$
13:     if $(UB(\mathcal{S}) - f_{best}) \leq \epsilon$ then
14:         output the current $x_{best}$ as the solution, terminate;
15:     else
16:         select $i$ such that $i = \text{argmax}_j\{s_j^U - s_j^L\}$, bisect $\mathcal{S}$ into $\mathcal{S}_1$ and $\mathcal{S}_2$ along the edge $i$.
            (case 1: $i \leq q$; case 2: $i > q$.)
17:     end if
18: end if
Reduce and Bound
19: for $k = 1, 2$ do
20:     reduce $\mathcal{S}_k$ to $\mathcal{S}_k = [x^{L_k}, x^{U_k}]$ according to reduction rules
```

```
21:     compute a suitable upper bound $UB(\mathcal{S}_k)$
22:     **if** $UB(\mathcal{S}_k) \leq f_{best}$ or $x^{L_k} \notin \mathcal{G}$ **then**
23:         continue;
24:     **else if** $x^{U_k} \in \mathcal{G}$ **then**
25:         update $x_{best}$ and $f_{best}$, if improved.
26:     **else**
27:         update $x_{best}$ and $f_{best}$, if improved.
28:         $\mathfrak{L} \leftarrow \mathfrak{L} \cup \{\mathcal{S}_k\}$
29:     **end if**
30: **end for**
31: goto **Select and Branch**.
```

Algorithm 1 Monotonic Branch-Reduce-Bound Algorithm

4.2.1 Select and Branch

The hyperrectangle with the greatest upper bound is selected for branching, as shown in line 12 in Algorithm 1. Then the subsequent branching is carried out along the longest side i of $\mathcal{S}$, with $i = \operatorname{argmax}_j\{s_j^U - s_j^L\}$. If $i \leq q$, then $\mathcal{S} = [s^L, s^U]$ is partitioned to $\mathcal{S}_1$ and $\mathcal{S}_2$ on the discrete variable:

$$\mathcal{S}_1 = \left[s^L, s^U - \left\lfloor \frac{s_i^U - s_i^L}{2} \right\rfloor e^i\right],$$

$$\mathcal{S}_2 = \left[s^L + \left\lfloor \frac{s_i^U - s_i^L}{2} + 1 \right\rfloor e^i, s^U\right],$$

where e^i is the i^{th} unit vector in $\mathcal{R}^n$. If $i > q$, then $\mathcal{S} = [s^L, s^U]$ is partitioned to $\mathcal{S}_1$ and $\mathcal{S}_2$ on the continuous variable:

$$\mathcal{S}_1 = \left[s^L, s^U - \left(\frac{s_i^U - s_i^L}{2}\right) e^i\right],$$

$$\mathcal{S}_2 = \left[s^L + \left(\frac{s_i^U - s_i^L}{2}\right) e^i, s^U\right],$$

where e^i is the i^{th} unit vector in $\mathcal{R}^n$. It is obvious that after partition the least and greatest elements in $\mathcal{S}_1$ and $\mathcal{S}_2$ belong to $\mathcal{X}_z$. A branching process is said to be exhaustive if there exists a sequence of partitions that shrink to a singleton [19]. The longest-edge bisection rule is exhaustive, as indicated in the proof of Theorem 4.1.

Although the longest-edge bisection rule is a popular choice for branching, one can use other rules such as the longest-edge trisection, and the largest increment edge bisection. In the largest incremental edge bisection, the edge is selected on which with the most increment in the objective function. That is $i = \text{argmax}_j\{f(s^L \backslash s_j^L \cup s_j^U)\}$, where $s^L \backslash s_j^L \cup s_j^U$ denotes the value of s^L except that its j^{th} index is replaced by s_j^U.

4.2.2 Reduce and Bound

The next major step is to reduce and bound the generated partitions, $\mathcal{S}_1$ and $\mathcal{S}_2$. The reduction process is to purge the inferior area to reduce the size of the search region so that it helps to obtain a better feasible solution and a tighter upper bound. The bold rectangles on the right hand side of Figure 4-4 shows $\mathcal{S}_1$ and $\mathcal{S}_2$ after reduction.

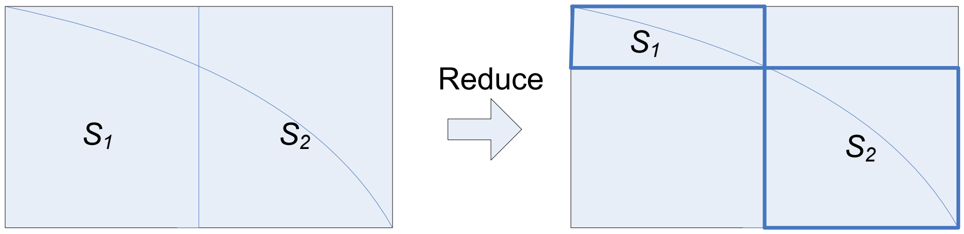


Figure 4-4. Reduce process

In the case, $i > q$, with

$$\mathcal{S}_1 = \left[s^L, s^U - \left(\frac{s_i^U - s_i^L}{2}\right)e^i\right],$$

$$\mathcal{S}_2 = \left[s^L + \left(\frac{s_i^U - s_i^L}{2}\right)e^i, s^U\right],$$

a reduction can be done as follows. Let

$$\alpha_j = \max\left\{\alpha \in [0,1] : s^L + \left(\frac{s_i^U - s_i^L}{2}\right)e^i + \alpha\left((s^U - s^L) - (s^U - s^L) * e^i\right)e^j \in \mathcal{G} \cap \mathcal{X}_z\right\}. \tag{4–18}$$

Geometrically, α_j is the maximum distance from point $s^L + \left(\frac{s_i^U - s_i^L}{2}\right) e^i$ to the boundary of set $\mathcal{G} \cap \mathcal{X}_z$. To achieve the value of α_j is a one dimensional problem can can be easily solved by bisection search when $j > q$ or by a sort algorithm when $j \leq q$. It helps to locate a purge point $P = s^L + \left(\frac{s_i^U - s_i^L}{2}\right) e^i + \alpha_j \left((s^U - s^L) - (s^U - s^L) * e^i\right) e^j$ on the boundary of set $\mathcal{G} \cap \mathcal{X}_z$: the bottom-left area to P can be purged because it is inferior to the point P and the upper-right area to P can be purged too because it is infeasible. More precisely, $\mathcal{S}_1$ and $\mathcal{S}_2$ can be reduced to

$$\mathcal{S}_1 = \left[s^U - \sum_{j=1}^{n} \left(s^U - s^L - \left(\frac{s_i^U - s_i^L}{2} \right) e^i \right) \alpha_j e^j, s^U - \left(\frac{s_i^U - s_i^L}{2} \right) e^i \right],$$

$$\mathcal{S}_2 = \left[s^L + \left(\frac{s_i^U - s_i^L}{2} \right) e^i, s^L + \sum_{j=1}^{n} \left(s^U - s^L - \left(\frac{s_i^U - s_i^L}{2} \right) e^i \right) (1 - \alpha_j) e^j \right].$$

After the reduction, the upper bound of a partition $\mathcal{S} = [s^L, s^U]$ can be calculated in various ways, such as using the original polyblock algorithm in [56]. In this paper, we simply use $f(s^U)$. A partition is discarded if its upper bound is less than the current best solution, or if it does not cover any feasible solution, as shown in line 22 in Algorithm 1.

4.2.3 Convergence Analysis

The selection of greatest upper bound hyperrectangle and the longest-edge bisection branch guarantee the convergence of Algorithm 1. The result is summarized in Theorem 4.1.

Theorem 4.1. *Algorithm 1 either terminates after finite iterations, producing an ϵ-optimal solution or detecting its infeasibility; or it is infinite with a sequence of hyper-rectangles $\mathcal{S}_{p_k} = [L^{p_k}, U^{p_k}]$ such that $\lim_{k\to\infty} L^{p_k} = \lim_{k\to\infty} U^{p_k} = x^*$, where x^* is an optimal solution.*

Proof. It is obvious that if Algorithm 1 terminates within finite iterations, it produces an ϵ-optimal solution or detects its infeasibility. If Algorithm 1 does not terminate within finite iterations, all the discrete variables must have been fixed. In addition, among the infinitely generated partitions, there exists a sequence such that $[L^{p_{k+1}}, U^{p_{k+1}}] \subset [L^{p_k}, U^{p_k}]$.

Due to the longest-edge bisection branch rule,

$$\lim_{k\to\infty} \left(U_j^{p_k} - L_j^{p_k}\right) = 0, \forall j = 1, \ldots, n.$$

It implies that

$$\lim_{k\to\infty} L^{p_k} = \lim_{k\to\infty} U^{p_k} = x^*.$$

We must also show that x^* is an optimal solution. From the selection of greatest upper bound hyperrectangle, we know that the upper bound of the partition in this sequence is no less than any feasible solution. That is, $f(U^{p_k}) \geq f(x), \forall x \in \mathcal{G} \cap \mathcal{X}_z$. Therefore,

$$\lim_{k\to\infty} f(L^{p_k}) = \lim_{k\to\infty} f(U^{p_k}) = f(x^*) \geq f(x), \forall x \in \mathcal{G} \cap \mathcal{X}_z.$$

It means that x^* is an optimal solution. □

From Theorem 4.1, we can add the following condition after line 16 in Algorithm 1 to terminate the algorithm in finite iterations.

```
if $\max_j\{s_j^U - s_j^L\} \leq \epsilon$ then
    output $s_j^L$ as the solution, terminate;
end if
```

4.2.4 Acceleration Techniques

In Algorithm 1, besides the reduction process, there are several other acceleration techniques worth mentioning.

Preprocess. The upper bound of x_i can be tightened by a number that is derived from its lower bound. Let

$$\lambda_i = \max\{\lambda : g_j(\lambda e^i) + g_j(x^L) \leq c_j\}, \tag{4–19}$$

we have all feasible $x_i \leq x_i^L + \lambda_i e^i$. In other words, $x_i \leq \min(x_i^U, x_i^L + \lambda_i e^i)$, which provides a tighter upper bound for x_i.

Selection and partition strategy. Algorithm 1 chooses the partition with the greatest upper bound. Computationally, a delicate rule to periodically select the partition between the greatest upper bound and the least upper bound helps the convergence of the algorithm.

Algorithm 1 employs the longest-edge bisection rule. One can use other rules such as the longest-edge trisection, and the largest incremental edge bisection to speed up the convergence in some cases. In the largest incremental edge bisection, the edge is selected on which with the most increment on the objective function. That is, $i = \text{argmax}_j\{f(s^L \backslash s_j^L \cup s_j^U)\}$, where $s^L \backslash s_j^L \cup s_j^U$ denotes the value of s^L except that its j^{th} index is replaced by s_j^U.

Improved upper bound. An improved upper bound on the hyperrectangle accelerates convergence if it can be calculated efficiently.

4.3 Using Monotonic Branch-Reduce-Bound Algorithm to Solve System Reliability Optimization Problems

In this section, the monotonic branch-reduce-bound algorithm (i.e. Algorithm 1) is applied to solve various problems arising from system reliability optimization. Most of these problems are thoroughly studied in the literature with various solution techniques, mainly heuristics. The limitation of a heuristic is that it may find good solutions, but without guaranteed optimality. In contrast to the heuristic techniques, the presented mBRB Algorithm, as an exact method, can find an ϵ-optimal solution in a finite number of steps. The algorithm that solves all the following instances is implemented in C++ and tested on a Dell Optiplex GX620 computer with a Pentium IV 3.6 GHz processor and 1.0 GB RAM, running under the Windows XP operating system.

4.3.1 Redundancy Allocation Optimization

The following two examples belong to the category of redundancy allocation optimization problems.

Example 1: Five-stage series system. The following problem is to maximize the system reliability of a five-stage series system which are subject to three nonlinear resource constraints. This problem is widely used to demonstrate a number of optimization techniques [28]. The problem was originally presented in [53].

$$\max \quad R_s = \prod_{i=1}^{5} R_i(x_i) \tag{4–20}$$

$$\text{subject to} \quad g_1 = \sum_{i=1}^{5} p_i x_i^2 \le P, \tag{4–21}$$

$$g_2 = \sum_{i=1}^{5} c_i \left[x_i + \exp\left(\frac{x_i}{4}\right)\right] \le C, \tag{4–22}$$

$$g_3 = \sum_{i=1}^{5} w_i x_i \exp\left(\frac{x_i}{4}\right) \le W, \tag{4–23}$$

$$x_i\text{: integer}, \quad \forall i = 1, \ldots, 5, \tag{4–24}$$

where $R_i = 1 - (1 - r_i)^{x_i}$ is the reliability of stage i. Constraint 4–21, g_1, is imposed on the combination of weight and volume: p_i is the product of weight per unit and volume per unit. Component reliability does not usually affect the weight nor the volume, hence g_1 is not a function of r_i ([54]). Constraint 4–22, g_2, is the cost constraint where $c_i x_i$ is the cost of all components at stage i and $c_i \exp\left(\frac{x_i}{4}\right)$ is the additional cost for interconnecting parallel components. Constraint 4–23, g_3, is the weight constraint where $w_i x_i$ is the weight of all components at stage i. The additional factor, $\exp\left(\frac{x_i}{4}\right)$, is added due to the hardware required for interconnecting components ([54]). The weight constraint is not a function of component reliability. The coefficients are given in Table 4-1.

Table 4-1. Coefficients in Example 1

i	r_i	p_i	c_i	w_i	P	C	W
1	0.80	1	7	7			
2	0.85	2	7	8			
3	0.90	3	5	8	110	175	200
4	0.65	4	9	6			
5	0.75	2	4	9			

It only takes the mBRB algorithm 15 milliseconds to obtain the proved optimal solution $(3, 2, 2, 3, 3)$ with the corresponding system reliability at 0.9044673. It is superior to many solution techniques collected in [28].

Example 2: Four-stage series system with 2-out-of-n: G configuration. In this example, stage 1 does not allow for component redundancy, but its reliability is determined by choosing a component from a pool of six different components at that stage. In other words, the reliability levels at stage 1 are discrete. The reliabilities of stages 2 and 4 can only be enhanced by providing redundancy. However, stage 3 has a special configuration called 2-out-of n: G configuration, which works (or is "good") if and only if at least 2 of the n components work (or are good). Based on this definition, the system reliability of stage 3 is shown in the expression of $R_3(x_3)$.

$$\begin{aligned}
\max \quad & R_s = \prod_{i=1}^{4} R_i(x_i) && (4\text{–}25)\\
\text{subject to} \quad & g_1 = 10\exp\left(\frac{0.02}{1-R_1(x_1)}\right) + 10x_2 + 6x_3 + 15x_4 \le 150, && (4\text{–}26)\\
& g_2 = 10\exp\left(\frac{x_1}{2}\right) + 4\exp(x_2) + 2\left[x_3 + \exp\left(\frac{x_3}{4}\right)\right] + 6x_4^2 \le 200, && (4\text{–}27)\\
& g_3 = 40x_1^2 + 6\exp(x_2) + 3x_3\exp\left(\frac{x_3}{4}\right) + 8x_4^3 \le 750, && (4\text{–}28)\\
& x_i\text{: integer}, \quad \forall i = 1, 2, 3, 4, && (4\text{–}29)
\end{aligned}$$

where stage reliabilities are

$$\begin{aligned}
R_1(x_1) &= 0.94, 0.95, 0.96, 0.965, 0.97, 0.975, \quad \text{for } x_1 = 1, 2, \ldots, 6, \text{respectively},\\
R_2(x_2) &= 1 - (1 - 0.75)^{x_2},\\
R_3(x_3) &= \sum_{k=2}^{x_3} \begin{pmatrix} x \\ k \end{pmatrix} (0.90)^k (1 - 0.90)^{x_3 - k},\\
R_4(x_4) &= 1 - (1 - 0.95)^{x_4}.
\end{aligned}$$

It takes the mBRB algorithm 27 iterations to produce and verify the optimal solution $(3, 3, 5, 3)$ with its objective value of 0.9444472 at no CPU time, or more precisely, less than 1 millisecond. The problem was solved by various specialized heuristics, such as [36], [23], and [1], all of which spent more time to solve than the mBRB algorithm and could not verify the optimality of their solutions, although they were able to produce the same solution. As a correction to the literature, it is noted that [1] contains the wrong expression of this example.

4.3.2 Reliability-Redundancy Allocation Optimization

Reliability-redundancy allocation optimization is a mixed integer nonlinear programming problem that is the general form of SROP. In this setting, a design engineer can improve the reliability of a system by increasing the component reliabilities or providing redundancy at various stages. The following two examples are widely used in the reliability literature.

Example 3: Five-stage series system with component reliability choice. This example is a variant of Example 1.

$$
\begin{aligned}
\max \quad & R_s = \prod_{i=1}^{5} R_i(x_i) && (4\text{–}30)\\
\text{subject to} \quad & g_1 = \sum_{i=1}^{5} p_i x_i^2 \le P, && (4\text{–}31)\\
& g_2 = \sum_{i=1}^{5} \alpha_i \left(\frac{-t}{\ln r_i}\right)^{\beta_i} \left[x_i + \exp\left(\frac{x_i}{4}\right)\right] \le C, && (4\text{–}32)\\
& g_3 = \sum_{i=1}^{5} w_i x_i \exp\left(\frac{x_i}{4}\right) \le W, && (4\text{–}33)\\
& 0 \le r_i \le 1, \quad \forall i = 1, \ldots, 5, && (4\text{–}34)\\
& x_i\text{: integer}, \quad \forall i = 1, \ldots, 5, && (4\text{–}35)
\end{aligned}
$$

where the objective function and constraints are the same as those of Example 1 except a more explicitly expression for the unit cost of component i, which is a function of the component reliability r_i. To derive the expression of $c_i(r_i) = \alpha_i \left(\frac{-t}{\ln r_i}\right)^{\beta_i}$, we follow the method in [54] and assume that the unit cost of component i is a decreasing function of the component failure rate λ_i expressed by

$$c_i(\lambda_i) = \alpha_i \lambda_i^{-\beta_i},$$

where α_i and β_i are the inherent characteristics of component i. If component i follows the negative exponential failure law, that is, $r_i = \exp(-\lambda_i t)$, then the component cost is $c_i(r_i) = \alpha_i \left(\frac{-t}{\ln r_i}\right)^{\beta_i}$, where t is the duration for which component i is required to operate. The coefficients of Example 3 are given in Table 4-2.

Table 4-2. Coefficients in Example 3

i	$\alpha_i \times 10^5$	p_i	w_i	β_i	P	C	W	t
1	2.330	1	7	1.5				
2	1.450	2	8	1.5				
3	0.541	3	8	1.5	110	175	200	1000
4	8.050	4	6	1.5				
5	1.950	2	9	1.5				

This problem is considered difficult to solve in the literature. To the best of our knowledge, there is no exact method being applied to solve this problem. We compare the performance of mBRB with the THK heuristic in [55], the GAG heuristic in [15], the KLXZ method in [26], the surrogate-constraints algorithm HNNN in [18], and the genetic algorithm GA in [20]. The comparison results are summarized in Table 4-3, where the CPU time listed in the last column is measured in seconds. The first column lists the names of the methods. The solutions and the obtained system reliability are listed in the second and the third column respectively. The fourth column, $R_s(UB)$, lists the upper bound of the system reliability.

The numbers in brackets after "mBRB" in the first column is the value of ϵ, the pre-defined optimality tolerance. As one can see, the mBRB algorithm produces higher quality solutions with known upper bound comparing to other algorithms, which are able to output some feasible solutions but without upper bound guarantee. With additional

Table 4-3. Performance comparison of Example 3

Method	(x, r)	R_s	$R_s(UB)$	CPU (s)
mBRB (0.01)	(3, 2, 2, 3, 3, 0.77500, 0.87500, 0.89250, 0.71500, 0.79000)	0.930947	0.940913	1.38
mBRB (0.001)	(3, 2, 2, 3, 3, 0.78250, 0.87500, 0.89938, 0.70750, 0.79000)	0.931541[a]	0.939004	4.76
THK	(3, 3, 2, 2, 3, 0.78438, 0.82500, 0.90000, 0.77500, 0.77813)	0.915363	-	-
GAG	(3, 2, 2, 3, 3, 0.80000, 0.86250, 0.90156, 0.70000, 0.80000)	0.930289	-	-
KLXZ	(3, 3, 2, 3, 2, 0.77960, 0.80065, 0.90227, 0.71044, 0.85947)	0.929750	-	-
HNNN	(3, 2, 2, 3, 3, 0.77489, 0.87007, 0.89855, 0.71652, 0.79137)	0.931451	-	-
GA	(3, 2, 2, 3, 3, 0.77943, 0.86848, 0.90267, 0.71404, 0.78689)	0.931578[b]	-	-

[a] Terminated at iteration 2000.

[b] Best solution obtained in multiple trials.

CPU time, the mBRB algorithm can produce a solution of

$$(3, 2, 2, 3, 3, 0.77781, 0.87187, 0.90281, 0.71313, 0.78625)$$

with the system reliability at 0.931669, and its upper bound of 0.933111. This result can serve as a benchmark for this example.

Example 4: Seven-link ARPA network. This problem is to maximize the reliability of the following seven-link ARPA network (see [35], [51]). The derivation of the reliability of this network, i.e. the objective function, is provided in Appendix A.

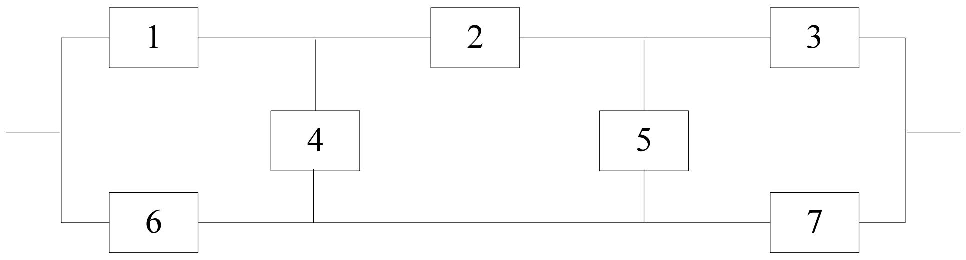


Figure 4-5. Seven-link ARPA network

$$\begin{aligned}
\max \quad & R_s = R_6R_7 + R_1R_2R_3(Q_6+R_6Q_7) + R_1R_4R_7Q_6(Q_2+R_2Q_3) && (4\text{–}36)\\
& \quad +R_3R_5R_6Q_7(Q_1+R_1Q_2) + R_1R_2R_5R_7Q_3Q_4Q_6 \\
& \quad +R_2R_3R_4R_6Q_1Q_5Q_7 + R_1R_3R_4R_5Q_2Q_6Q_7 \\
\text{subject to} \quad & g_1 = x_1x_2 + 0.5x_1\ln(1+x_3) + x_4 + 2x_5 + 0.3\exp\left(\frac{0.02}{1-R_6}\right) && (4\text{–}37)\\
& \quad +0.3\exp\left(\frac{0.01}{1-R_7}\right) \le 27, \\
& g_2 = (x_1+2x_2+1.2x_3)\ln(1+x_1+x_2+2x_3) + 0.4x_4 && (4\text{–}38)\\
& \quad +0.2x_5\exp\left(\frac{0.02}{1-R_6}\right) + 0.5\exp\left(\frac{0.01}{1-R_7}\right) \le 29, \\
& 0.5 \le R_i \le 0.99, \quad \forall i = 6,7, && (4\text{–}39)\\
& x_i\text{: integer}, \quad \forall i = 1,\dots,5, && (4\text{–}40)
\end{aligned}$$

where $R_i = 1-(1-r_i)^{x_i}, \forall i = 1,\dots,5$, $Q_i = 1-R_i, \forall i = 1,\dots,7$, and $r_1 = 0.70, r_2 = 0.90, r_3 = 0.80, r_4 = 0.65, r_5 = 0.70$.

This problem was originally solved using convexification method coded in FORTRAN in [51]. The comparison between the mBRB method and the convexification one is done in Table 4-4, where both algorithms obtain the optimal objective value. Admittedly, comparing CPU seconds directly does not reflect the absolute efficiency, since the convexification method was implemented in an older computer system. However, the table clearly shows relative efficiency of the mBRB algorithm: It takes only negligible CPU seconds to get the optimal solution.

Table 4-4. Performance comparison of Example 4

Method	(x, r)	R_s	$R_s(UB)$	CPU (s)
mBRB	(4, 1, 3, 4, 3, 0.9845, 0.9900)	0.99974	0.99974	0.01
Convexification	(4, 1, 3, 4, 3, 0.9845, 0.9899)	0.99974	-	38.63[a]

[a] It is measured on a SUN SPARCstation 5.

4.4 Conclusions

A monotonic branch-reduce-bound algorithm for mixed integer nonlinear programming is presented in this chapter. Its convergence and acceleration techniques are also discussed.

The algorithm is successfully applied to solve a wide range of the system reliability optimization problems. Computational results have been reported to show the superior efficiency of the algorithm over existing ones.

We expect that the monotonic branch-reduce-bound algorithm can be applied to other classes of the problem, such as nonlinear multidimensional knapsack problems, and generalized multiplicative programming problems. In the future research, we also would like to compare the performance of the acceleration techniques mentioned in this chapter, and analyze its worst-case performance theoretically and computationally.

CHAPTER 5
FORTIFYING THE RELIABILITY OF EXISTING FACILITIES AND MONOTONIC OPTIMIZATION

5.1 Introduction

In Chapter 2, we consider the impact of unreliability from the facilities when the system is initially designed. In that case, we are given the option to build facilities from scratch. However, redesigning an entire system is not always an available option given the potentially large expenses involved in closing existing facilities or opening new ones. In many situations, methods for protecting existing infrastructure may be preferable given limited resources available. In this chapter, we address the issues on fortifying the reliability of existing facilities.

A majority of research on reliable supply chain design has been focused on the initial system design, but not on how to improve the existing system. These works have been well documented in [47] and surveyed in Chapter 2 as well. However, reinforcing the components of an existing system may become more valuable and realistic considering the increased potential disruptions and uncertainties. These disruptions and uncertainties may evolve from the new challenges that were never faced when the initial systems were designed. They can be man-made disruptive events or natural disasters, for example, the September 11, 2001 terrorist attack and Hurricane Katrina in 2005.

Only a small strand of literature has been devoted to addressing the fortification of existing facilities, which includes [43], [44], and [49]. These works typically focus on the interdiction-fortification framework based upon the p-median facility location problem. The problems are generally formulated in the form of bilevel programming. These models can help to identify the critical facilities to protect under the events of disruption.

Another related strand of literature is on the network interdiction problems that are mainly developed for military applications, e.g. [12], [33], [62] and [30]. These models study the impact of losing one or more transportation links or network arcs based on the network flow problem. These models can help to identify the critical arcs to protect under the events of disruption.

In this chapter, we follow the theme of Chapter 2, but assume that the facilities have been built and can be reinforced to be more reliable. We also assume that the facilities are uncapacitated. We present two novel models whose objectives are to minimize the expected connection (service) and penalty cost by allocating the limited fortification resources to the open unreliable facilities. In the first model, the fortification efforts are continuous, that is, the failure probability at each facility varies from 0 to 1 at the fortification stage. We call it the Continuous Facility Fortification Model (CFFM). On the contrary, in the second model the fortification efforts are subject to different level of resources. The failure probability at each facility can only be chosen from a set of discrete levels at the fortification stage. Accordingly we call it the Discrete Facility Fortification Model (DFFM). Both models can help to identify the critical facilities to protect and optimally determine how much resources should be allocated to achieve the objective. To some extent, the models mathematically resemble the reliability allocation problem we discussed in Chapter 4.

The remainder of this chapter is organized as follows. In Section 5.2, we present the continuous facility fortification model and reveal its connection to the generalized linear multiplicative programming and the inherent monotonicity. An example is presented in Section 5.2.2 to illustrate the solution structure and properties. In Section 5.3, we present the discrete facility fortification model and apply the monotonic branch-reduce-algorithm to exploit the monotonicity properties inherent in the problem. The efficiency of the algorithm is demonstrated in Section 5.3.2. Section 5.3.2 also contains an analysis of the solution structure and tradeoff between cost deduction and fortification effort. We conclude this chapter in Section 5.4.

5.2 Continuous Facility Fortification Model

We first introduce some common notations that will be used throughout the chapter. Let D denote the set of clients or demand points and F denote the set of facilities. Let f_i be the facility cost to open facility i, d_j be the demand of client j, and c_{ij} be the service cost if j is serviced by facility i. For each client $j \in D$, if it is not served by any open and operational facility, then a penalty cost r_j will be incurred.

In the reliable facility location model setting, each client is assigned to a set of backup facilities, which is differentiated by the levels: in case of a lower level facility fails, the next level facility, if functional, will back it up. $\bar{x}_{ij}^k = 1$ if facility i is the k-th level backup facility of demand node j and $\bar{x}_{ij}^k = 0$ otherwise; $\bar{z}_j^k = 1$ if j has $(k-1)$-th backup facility, but has no k-th backup facility so that j incurs a penalty cost at level k. Contrary to the models in Chapter 2, $\bar{x}_{ij}^k$ and $\bar{z}_j^k$ are not decision variables anymore. They are used to specify the existing network.

To compute the expected failure cost, we follow the logic in Chapter 2. First of all, we need to compute the expected failure cost at level k served by facility i. Each demand node j is served by its level-k facility if all the lower level facilities become non-operational. For any facility l, if it is assigned to a lower level (i.e, less than k) for demand node j, then $\sum_{s=1}^{k-1} \bar{x}_{lj}^s = 1$, otherwise it is zero. So the probability that all lower levels facilities fail is $\prod_{l \in F} p_l^{\sum_{s=1}^{k-1} \bar{x}_{lj}^s}$. And j is served at level-k by i, which has to be operational. The probability is $(1 - p_i)$. Therefore, the expected failure cost at level k served by facility i is $d_j c_{ij} \bar{x}_{ij}^k (1 - p_i) \prod_{l \in F} p_l^{\sum_{s=1}^{k-1} \bar{x}_{lj}^s}$. Similarly, we can calculate the penalty cost at level k, which is $\prod_{l \in F} p_l^{\sum_{s=1}^{k-1} \bar{x}_{lj}^s} d_j r_j \bar{z}_j^k$.

We can now formulate the continuous facility fortification model as follows:

$(CFFM)$

$$\begin{aligned} \text{minimize} \quad & \sum_{j\in D}\sum_{k=1}^{|F|}\sum_{i\in F} d_j c_{ij}\bar{x}_{ij}^k(1-p_i)\prod_{l\in F} p_l^{\sum_{s=1}^{k-1}\bar{x}_{lj}^s} + \sum_{j\in D}\sum_{k=1}^{|F|+1}\prod_{l\in F} p_l^{\sum_{s=1}^{k-1}\bar{x}_{lj}^s} d_j r_j \bar{z}_j^k && (5\text{–}1)\\ \text{subject to} \quad & g_j(p_1,\ldots,p_{|F|}) \le 0, \quad j=1,\ldots,m && (5\text{–}2)\\ & 0 \le p_i \le 1, \quad \forall i \in F. && (5\text{–}3) \end{aligned}$$

The objective function (5–1) is the sum of the expected failure cost and the expected penalty cost. Constraints (5–2) denote various resource restrictions on the fortification levels, $p_i, \forall i \in F$. We assume g_j $(j = 1,\ldots,m)$ are convex functions so that the solution domain is convex. Constraints (5–3) are natural constraints on the failure probability.

5.2.1 Properties of the Continuous Facility Fortification Model

In this section, we show that CFFM is a special case of the following generalized linear multiplicative programming (GLMP) problem (see [41] for details).

$(GLMP)$

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{t}\prod_{i=1}^{k_j}(\mathbf{c}_{ij}^T\mathbf{x} + d_{ij}) && (5\text{–}4)\\ \text{subject to} \quad & \mathbf{x} \in \mathbf{X} && (5\text{–}5) \end{aligned}$$

where $\mathbf{c}_{ij} \in \mathbb{R}^n$, $d_{ij} \in \mathbb{R}^n$, $j = 1,\ldots,t, i = 1,\ldots,p_j$, and $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{X}$ is a nonempty convex set. Note that without the summation sign in the objective function 5–4, GLMP is reduced to linear multiplicative programming, another active topic in global optimization.

We first show that the objective function (5–1) is the sum of linear multiplicative terms with positive coefficients. In other words, the objective function (5–1) can be reduced to the form of

$$\sum_j \alpha_j \prod_{i=1}^{|F|} p_i^{\beta_{ij}} \qquad (5\text{–}6)$$

where α_j are nonnegative and β_{ij} are binary.

Proposition 5.1. *The objective function, 5–1, of the continuous facility fortification model can be reduced to the form of* $\sum_j \alpha_j \prod_{i=1}^{|F|} p_i^{\beta_{ij}}$ *where* α_j *are nonnegative and* β_{ij} *are binary.*

Proof. To prove this proposition, it is sufficient to show that any cost term related to an arbitrary demand node j has the form of $\alpha_j \prod_{i=1}^{|F|} p_i^{\beta_{ij}}$. Considering demand node j, it contributes

$$\sum_{k=1}^{|F|} \sum_{i \in F} d_j c_{ij} \bar{x}_{ij}^k (1 - p_i) \prod_{l \in F} p_l^{\sum_{s=1}^{k-1} \bar{x}_{lj}^s} + \sum_{k=1}^{|F|+1} \prod_{l \in F} p_l^{\sum_{s=1}^{k-1} \bar{x}_{lj}^s} d_j r_j \bar{z}_j^k$$

to the total cost. If no facility is assigned to it, then only the penalty term left, that is $d_j r_j$ which apparently takes the form of $\alpha_j \prod_{i=1}^{|F|} p_i^{\beta_{ij}}$ with $\beta_{ij} = 0$ and $\alpha_j = d_j r_j$. Otherwise, without loss of the generality, we assume facility i' is assigned to it at level k. So the service cost at level k is

$$d_j c_{i'j} (1 - p_{i'}) \prod_{l \in F} p_l^{\sum_{s=1}^{k-1} \bar{x}_{lj}^s}. \tag{5–7}$$

The only negative term is

$$-\, d_j c_{i'j} p_{i'} \prod_{l \in F} p_l^{\sum_{s=1}^{k-1} \bar{x}_{lj}^s}. \tag{5–8}$$

At level $k+1$, two cases can happen:

1. No facility is assigned to demand j at level $k+1$. It is subject to penalty cost

$$\prod_{l \in F} p_l^{\sum_{s=1}^{k-1} \bar{x}_{lj}^s} d_j r_j. \tag{5–9}$$

Because $r_j \geq c_{i'j}$ in this case, adding terms 5–8 and 5–9 leads to

$$d_j (r_j - c_{i'j}) p_{i'} \prod_{l \in F} p_l^{\sum_{s=1}^{k-1} \bar{x}_{lj}^s},$$

which is in the form of $\alpha_j \prod_{i=1}^{|F|} p_i^{\beta_{ij}}$. Note that $\sum_{s=1}^{k-1} \bar{x}_{lj}^s$ is a binary value.

2. Facility i'' is assigned to it. And the corresponding service cost is

$$c_{i''j}(1-p_{i''})p_{i'}\prod_{l\in F}p_l^{\sum_{s=1}^{k-1}\bar{x}_{lj}^s}. \tag{5–10}$$

Adding terms 5–8 and 5–10, we have

$$d_j(c_{i''j}-c_{i'j})p_{i'}\prod_{l\in F}p_l^{\sum_{s=1}^{k-1}\bar{x}_{lj}^s}-d_jc_{i''j}p_{i''}p_{i'}\prod_{l\in F}p_l^{\sum_{s=1}^{k-1}\bar{x}_{lj}^s}. \tag{5–11}$$

From Proposition 2.1, we know that $c_{i''j}-c_{i'j}\geq 0$, so the only negative term left is

$$-d_jc_{i''j}p_{i''}p_{i'}\prod_{l\in F}p_l^{\sum_{s=1}^{k-1}\bar{x}_{lj}^s} \tag{5–12}$$

This term can be absorbed in the next level assignment following a similar analysis at level k. By doing this process recursively, demand j will be eventually subject to penalty cost at certain level higher than $k+1$ and only case 1 can happen. Therefore, all cost terms related to demand node j have the form of $\alpha_j\prod_{i=1}^{|F|}p_i^{\beta_{ij}}$. This completes the proof. □

As a direct result from Proposition 5.1, the following Corollary holds.

Corollary 5.1. *The objective function 5–1 of the continuous facility fortification model is monotonically nondecreasing.*

It is obvious that Function 5–6 is a special case of Function 5–4: If $\beta_{ij}=0$ in function 5–6, then set $\mathbf{c}_{ij}=\mathbf{0}$ and $d_{ij}=\alpha_j$; If $\beta_{ij}=1$, then set $\mathbf{c}_{ij}=\alpha_j\mathbf{e}^i$, $\mathbf{e}^i$ being the i^{th} unit vector in $\mathbb{R}^n$ and $d_{ij}=0$. Therefore, the continuous facility fortification problem is a special form of the generalized linear multiplicative programming. The latter is multiextremal and possesses several local minima [41]. The existing algorithms for GLMP include outer-approximation methods [25], vertex enumeration methods [19], heuristics methods [31], among others. Corollary 5.1 also allows us to apply the monotonic branch-reduce-bound algorithm presented in Chapter 4 when the resource constraints possess monotonicity as well.

5.2.2 An Example of the Continuous Facility Fortification Model

In this section, an example is used to illustrate the solution structure of the continuous facility fortification model and how it helps to identify the critical facilities to protect in the system. In this example, the following constraint is posed:

$$\sum_{i \in F} p_i \geq L \tag{5–13}$$

where L is nonnegative and can be interpreted as the indication of the (un)reliability of the whole system. In other words, with the available fortification resources the system can be maintained at the (un)reliability level of L. If L is positive, not every facility can operate at $p_i = 0$. On the other hand, $L = 0$ means that the system is totally functional: No facility has a possibility to fail. The CFFM model is then formulated as follows,

$$\begin{aligned}
\text{minimize} \quad & \sum_{j \in D} \sum_{k=1}^{|F|} \sum_{i \in F} d_j c_{ij} \bar{x}_{ij}^k (1 - p_i) \prod_{l \in F} p_l^{\sum_{s=1}^{k-1} \bar{x}_{lj}^s} + \sum_{j \in D} \sum_{k=1}^{|F|+1} \prod_{l \in F} p_l^{\sum_{s=1}^{k-1} \bar{x}_{lj}^s} d_j r_j \bar{z}_j^k && \text{(5–14)} \\
\text{subject to} \quad & \sum_{i \in F} p_i \geq L && \text{(5–15)} \\
& 0 \leq p_i \leq 1, \quad \forall i \in F. && \text{(5–16)}
\end{aligned}$$

A network with 20 demand nodes and 5 open facilities is considered in this example. Demands d_j were drawn from $U[0, 1000]$ and rounded to the nearest integer, and x and y coordinates were drawn from $U[0, 1]$. Transportation costs c_{ij} are set to be equal to the Euclidean distance between i and j. The penalty cost r_j were drawn from $U[0, 15]$. The dataset is available in Appendix C. Facilities 2, 5, 15, 18, and 20 are open in this example. We use a vertex enumeration method to solve this problem and obtain the results at various level of L which are shown in Table 5-1 and plotted in Figure 5-1. The first column of Table 5-1 is the (un)reliability of the system. The second to sixth columns report the solution of this example: the profile of the failure probability at each open facility. It is followed by a column showing the total cost. The last column calculates the difference of the total cost between current row and previous row. It indicates the sensitivity of L on the objective value.

Table 5-1 and Figure 5-1 show how the system deteriorates (total cost increases) as the resource to maintain the system reliability decreases. In addition, the last column of Table 5-1 indicates that the curve in Figure 5-1 is piecewise linear. For example, the difference, similarly the slope, in total cost is a constant when L changes from 1.0 to 2.0 in a step of 0.1. This is because of (1) the failure probabilities of all but one facility (20, in this case) remain the same, the whole cost structure does not change; (2) Changing the failure probability in one facility will only affect all the service cost that related to this particular facility.

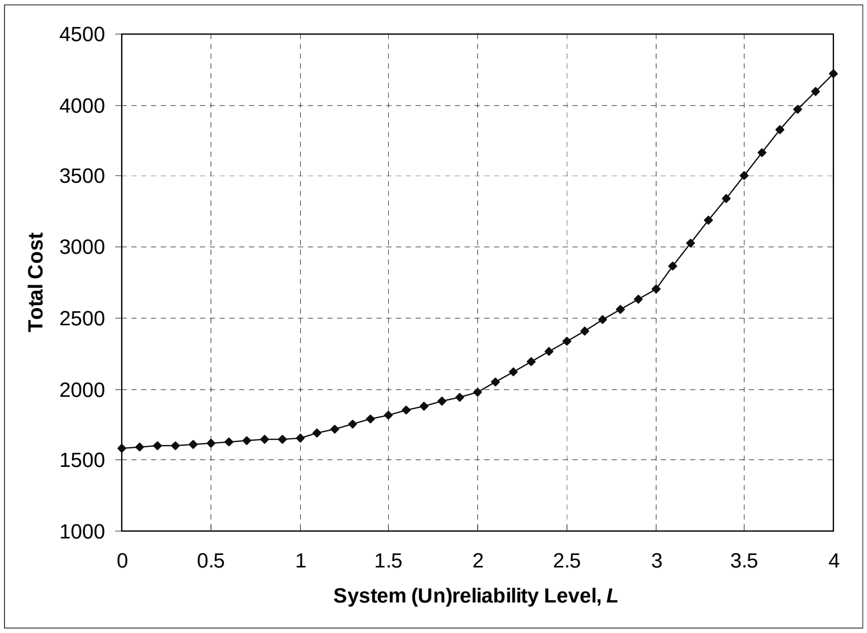


Figure 5-1. Total cost at different system reliability level

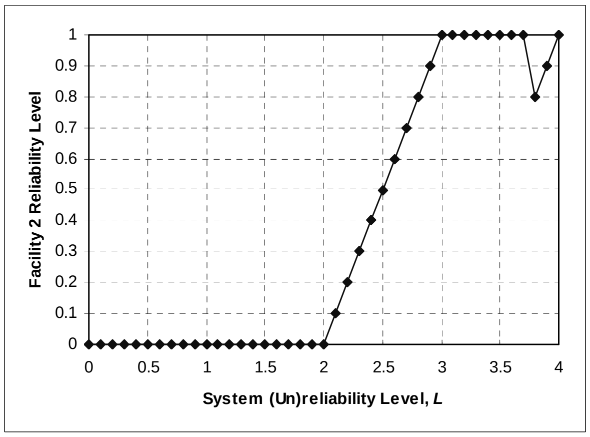
There is a common pattern in Table 5-1: when L increases by 0.1, only one of the facilities changes the failure probability accordingly. For example, when L increases from 1.7 to 1.8, the failure probability of facility 20 jumps from 0.7 to 0.8, all the others remaining the same. The failure probability at an individual facility usually changes by either 0.1 or 0.0 as shown in most cases in Figure 5-2. However, there is one notable

Table 5-1. Solution of a CFFM model

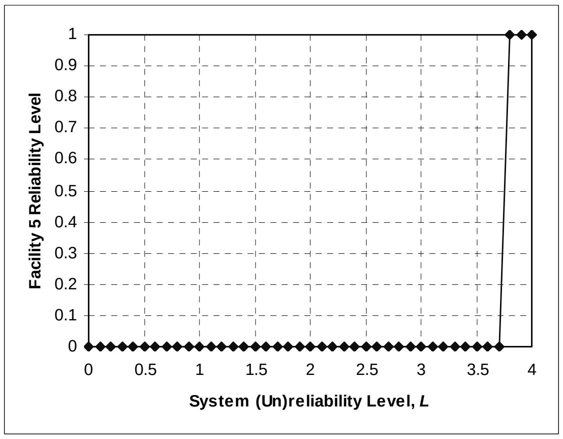
	Failure Probability at						
L	2	5	15	18	20	Total Cost	Difference
0	0	0	0	0	0	1582.93	n/a
0.1	0	0	0.1	0	0	1590.37	7.44
0.2	0	0	0.2	0	0	1597.80	7.43
0.3	0	0	0.3	0	0	1605.24	7.44
0.4	0	0	0.4	0	0	1612.67	7.43
0.5	0	0	0.5	0	0	1620.11	7.44
0.6	0	0	0.6	0	0	1627.55	7.44
0.7	0	0	0.7	0	0	1634.98	7.43
0.8	0	0	0.8	0	0	1642.42	7.44
0.9	0	0	0.9	0	0	1649.86	7.44
1.0	0	0	1	0	0	1657.29	7.43
1.1	0	0	1	0	0.1	1689.40	32.11
1.2	0	0	1	0	0.2	1721.51	32.11
1.3	0	0	1	0	0.3	1753.62	32.11
1.4	0	0	1	0	0.4	1785.73	32.11
1.5	0	0	1	0	0.5	1817.84	32.11
1.6	0	0	1	0	0.6	1849.95	32.11
1.7	0	0	1	0	0.7	1882.06	32.11
1.8	0	0	1	0	0.8	1914.17	32.11
1.9	0	0	1	0	0.9	1946.28	32.11
2.0	0	0	1	0	1	1978.39	32.11
2.1	0.1	0	1	0	1	2050.81	72.42
2.2	0.2	0	1	0	1	2123.23	72.42
2.3	0.3	0	1	0	1	2195.65	72.42
2.4	0.4	0	1	0	1	2268.06	72.41
2.5	0.5	0	1	0	1	2340.48	72.42
2.6	0.6	0	1	0	1	2412.90	72.42
2.7	0.7	0	1	0	1	2485.31	72.41
2.8	0.8	0	1	0	1	2557.73	72.42
2.9	0.9	0	1	0	1	2630.15	72.42
3.0	1	0	1	0	1	2702.56	72.41
3.1	1	0	1	0.1	1	2863.49	160.93
3.2	1	0	1	0.2	1	3024.41	160.92
3.3	1	0	1	0.3	1	3185.34	160.93
3.4	1	0	1	0.4	1	3346.26	160.92
3.5	1	0	1	0.5	1	3507.19	160.93
3.6	1	0	1	0.6	1	3668.11	160.92
3.7	1	0	1	0.7	1	3829.04	160.93
3.8	0.8	1	0	1	1	3972.66	143.62
3.9	0.9	1	0	1	1	4099.09	126.43
4.0	1	1	0	1	1	4225.52	126.43

exception in this pattern. That is when L increases from 3.7 to 3.8, both facility 2 and facility 18 change their failure probability in the optimal solution. It means that the optimal vertex in $L = 3.8$ is dramatically changed from $L = 3.7$. This exception is clearly signaled in Figure 5-2 if we look at the changes from $L = 3.7$ to $L = 3.8$ in sub-figures A to D: a 0.2 dip in A, a 1.0 surge in B, a 1.0 sink in C, and a 0.3 jump in D. This phenomenon is from the multiextremalness of the CFFM problem that we discuss in Section 5.2.1. Using a bisection search, we find out that at $L = 0.749842$, both vertex (1, 0, 1, 0.749842, 1) and vertex (0.749842, 1, 0, 1, 1) produce the same objective value of 3909.2476. It means that at $L = 0.749842$, the optimal vertex transient from (1, 0, 1, 0.749842, 1) to (0.749842, 1, 0, 1, 1). Then from $L = 0.749842$ to $L = 0.8$, the same pattern still holds, facility 2 is the only one that changes its failure probability.

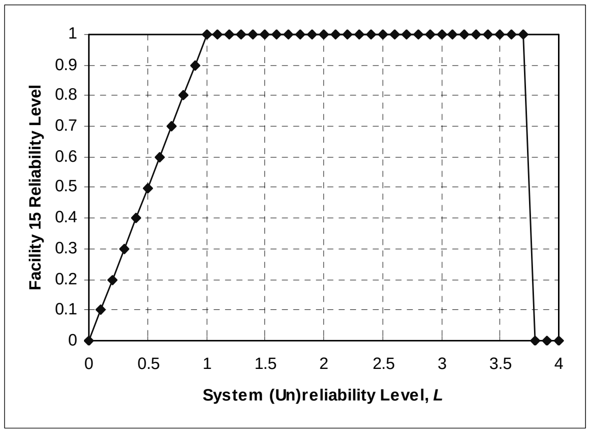
As we point out in the introduction that CFFM model can also help to find the key facility to fortify. In this example, the frequency that a facility is completely open is depicted in Figure 5-3. It shows that facility 5 is very critical in this system, because it is chosen to be completely secured 38 times out of the total 41 cases in this example.



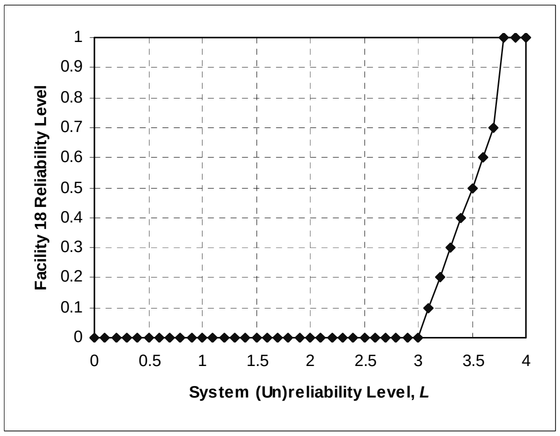
A) Facility 2



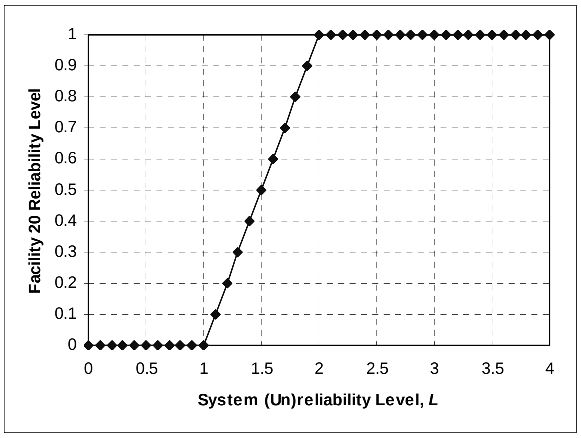
B) Facility 5



C) Facility 15



D) Facility 18



E) Facility 20

Figure 5-2. Failure probability at individual facility vs system reliability Level

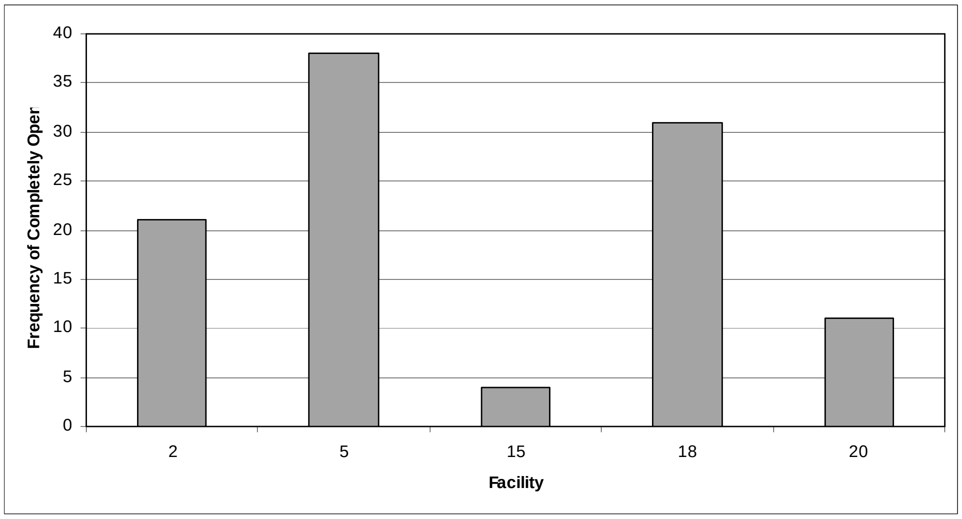


Figure 5-3. Frequency of completely open facility in Table 5-1

5.3 Discrete Facility Fortification Model

The underlying assumptions of the continuous facility fortification model in Section 5.2 are that (1) the limited fortification resource is uniformly distributed across all the facilities; and (2) the distributed resource is dividable. However, in many cases, the fortification resource can only be discretely distributed. That is, the fortification resource at a facility is categorized into different levels, which are functions of available resources. For example, 7000 units of resource may improve the failure probability of a facility to $p = 0.4$, while 8000 units can improve it to $p = 0.3$. But there is no amount of resource that can improve the failure probability to a number between $p = 0.4$ and $p = 0.3$. This inspires us to consider a discrete facility fortification model (DFFM).

In DFFM, let y_i be the fortification level at facility i. Naturally, we assume y_i is a positive integer and only one fortification level is allowed at each facility. The corresponding failure probability and the amount of resource are denoted by functions $P_i(y_i)$ and $V_i(y_i)$ respectively. $P_i(y_i)$, $i \in F$, are nonincreasing functions of y_i, whereas $V_i(y_i)$, $i \in F$, are nondecreasing functions of y_i. This is because more fortification resource would make a facility more reliable, and consequently a smaller failure probability. Let the upper bound of y_i be U_i and the total resource be R.

As in CFFM, the objective of DFFM is to minimize the total expected transport and failure-to-serve cost. Notation $\bar{x}_{ij}^k$ and $\bar{z}_j^k = 1$ keep the same meaning as in CFFM: $\bar{x}_{ij}^k = 1$ if facility i is the k-th level backup facility of demand node j and $\bar{x}_{ij}^k = 0$ otherwise; $\bar{z}_j^k = 1$ if j has $(k-1)$-th backup facility, but has no k-th backup facility so that j incurs a penalty cost at level k. The cost terms are calculated in parallel to CFFM as well, which are shown in the following formulation.

$(DFFM)$

$$\begin{aligned}
\text{minimize} \quad & \sum_{j\in D}\sum_{k=1}^{|F|}\sum_{i\in F} d_j c_{ij}\bar{x}_{ij}^k(1-P_i(y_i))\prod_{l\in F} P_l(y_l)^{\sum_{s=1}^{k-1}\bar{x}_{lj}^s} \\
& + \sum_{j\in D}\sum_{k=1}^{|F|+1}\prod_{l\in F} P_l(y_l)^{\sum_{s=1}^{k-1}\bar{x}_{lj}^s} d_j r_j \bar{z}_j^k && (5\text{–}17) \\
\text{subject to} \quad & \sum_{i\in F} V_i(y_i) \le R, && (5\text{–}18) \\
& 1 \le y_i \le U_i, \quad \forall i \in F && (5\text{–}19) \\
& y_i\text{: integer}, \quad \forall i \in F. && (5\text{–}20)
\end{aligned}$$

The objective function (5–17) is the sum of the expected failure cost and the expected penalty cost. Constraint (5–18) denotes the resource restrictions on the fortification. Constraints (5–19) and (5–20) are integral constraints on the fortification levels.

5.3.1 Properties and Algorithms

In this section, we show that DFFM can be solved via the monotonic branch-reduce-bound algorithm. It is obvious that constraint (5–18) possesses the monotonicity, since $V_i(y_i)$, $i \in F$, are nondecreasing functions of y_i. The objective function of the continuous facility fortification model is shown to be monotonically nondecreasing in Corollary 5.1. A similar result holds in the discrete case, because $P_i(y_i)$, $i \in F$, are nonincreasing functions of y_i. From the fact that the composite of a nondecreasing function and a nonincreasing function is nonincreasing, we have the following corollary, which is parallel to Corollary 5.1.

Corollary 5.2. *The objective function 5–17 of the discrete facility fortification model is monotonically nonincreasing.*

Therefore, DFFM is a monotonic optimization problem which can be solved by the monotonic branch-reduce-bound algorithm presented in Chapter 4. To be consistent with the general format of the monotonic optimization in this thesis, we transform DFFM model to a maximization problem.

(*DFFM-max*)

$$\begin{aligned}
\text{maxmize} \quad & -\sum_{j\in D}\sum_{k=1}^{|F|}\sum_{i\in F} d_j c_{ij}\bar{x}_{ij}^k (1-P_i(y_i))\prod_{l\in F} P_l(y_l)^{\sum_{s=1}^{k-1}\bar{x}_{lj}^s} \\
& \qquad -\sum_{j\in D}\sum_{k=1}^{|F|+1}\prod_{l\in F} P_l(y_l)^{\sum_{s=1}^{k-1}\bar{x}_{lj}^s} d_j r_j \bar{z}_j^k && (5\text{–}21)\\
\text{subject to} \quad & \sum_{i\in F} V_i(y_i) \le R, && (5\text{–}22)\\
& 1 \le y_i \le U_i, \quad \forall i \in F && (5\text{–}23)\\
& y_i\text{: integer}, \quad \forall i \in F. && (5\text{–}24)
\end{aligned}$$

A more general format of DFFM-max can be written as the following by replacing the decision variables y_i with x_i.

$$\begin{aligned}
(MO)\text{:} \quad \max \quad & R_s = f(x) && (5\text{–}25)\\
\text{subject to} \quad & x \in \mathcal{G} = \{x|\ g_i(x) \le c_i, \quad \forall i = 1,\dots,m\}, && (5\text{–}26)\\
& x \in [x^L, x^U], && (5\text{–}27)\\
& x_j\text{: integer}, \quad \forall j = 1,\dots,n, && (5\text{–}28)
\end{aligned}$$

where $[x^L, x^U]$ denotes a hyperrectangle with lowest boundary x^L and greatest boundary x^U, functions f and g_i are nondecreasing for $i = 1,\dots,m$. To ensure $\mathcal{G}$ is closed, we assume g_i are semi-continuous for $i = 1,\dots,m$.

For the completeness of exposition, we also include the description of the monotonic branch-reduce-bound algorithm with slight differences than that in Chapter 4 to accommodate this pure nonlinear integer programming. In the description of Algorithm 2, $\mathcal{S}$ denotes a hyperrectangle partition; $\mathfrak{L}$ is a list of unfathomed hyperrectangles; ϵ is a pre-defined optimality tolerance parameter; x_{best} and f_{best} denote the current best solution and objective value respectively; $UB(\mathcal{S})$ is the upper bound of the objective function over $\mathcal{S}$. Besides initialization, the major steps are described as follows.

```
Initialization
 1: $x^L \leftarrow (\lceil x_1^L \rceil, \ldots, \lceil x_n^L \rceil)$, $x^U \leftarrow (\lfloor x_1^U \rfloor, \ldots, \lfloor x_n^U \rfloor)$
 2: if $x^U \in \mathcal{G}$ then
 3:     $x^U$ is the optimal solution, terminate;
 4: else if $x^L \notin \mathcal{G}$ then
 5:     the problem is infeasible, terminate;
 6: else
 7:     set $\mathcal{S} = [x^L, x^U], \epsilon > 0, \mathfrak{L} = \{\mathcal{S}\}$, $x_{best} = x^L$, $f_{best} = f(x^L)$
 8: end if
Select and Branch
 9: if $\mathfrak{L} = \emptyset$ then
10:     output the current $x_{best}$ as the solution, terminate;
11: else
12:     select $\mathcal{S} = [s^L, s^U] \in \mathfrak{L}$ such that $UB(\mathcal{S}) = \max_{S \in \mathfrak{L}}\{UB(S)\}$ and $\mathfrak{L} \leftarrow \mathfrak{L} \backslash \{\mathcal{S}\}$
13:     if $(UB(\mathcal{S}) - f_{best}) \leq \epsilon$ then
14:         output the current $x_{best}$ as the solution, terminate;
15:     else
16:         select $i$ such that $i = \text{argmax}_j\{s_j^U - s_j^L\}$, bisect $\mathcal{S}$ into $\mathcal{S}_1$ and $\mathcal{S}_2$ along the edge $i$.
17:     end if
18: end if
Reduce and Bound
19: for $k = 1, 2$ do
20:     reduce $\mathcal{S}_k$ to $\mathcal{S}_k = [x^{L_k}, x^{U_k}]$ according to reduction rules
21:     compute a suitable upper bound $UB(\mathcal{S}_k)$
22:     if $UB(\mathcal{S}_k) \leq f_{best}$ or $x^{L_k} \notin \mathcal{G}$ then
23:         continue;
24:     else if $x^{U_k} \in \mathcal{G}$ then
```

```
25:         update $x_{best}$ and $f_{best}$, if improved.
26:     else
27:         update $x_{best}$ and $f_{best}$, if improved.
28:         $\mathfrak{L} \leftarrow \mathfrak{L} \cup \{\mathcal{S}_k\}$
29:     end if
30: end for
31: goto Select and Branch.
```

Algorithm 2 Monotonic Branch-Reduce-Bound Algorithm

5.3.2 Computational Experiments

One advantage of using the monotonic branch-reduce-bound algorithm to solve the discrete facility fortification problem is that no closed form of $P_i(y_i)$, $i \in F$, and $V_i(y_i)$, $i \in F$ are required as long as they are monotonic. In this section, we use the same dataset as the one in section 5.2.2, which is listed in Appendix C. The dataset contains 20 demand nodes and 5 open facilities with the following specification.

Table 5-2. Input data ($V_i(y_i)$ and $P_i(y_i)$) for the 3-level model

Open Facility i	Level 1		Level 2		Level 3	
	V	P	V	P	V	P
2	0	0.55	79	0.39	688	0.02
5	0	0.85	614	0.45	728	0.28
15	0	0.75	303	0.63	855	0.48
18	0	0.52	135	0.48	409	0.20
20	0	0.30	178	0.24	273	0.22

The monotonic branch-reduce-bound algorithm is implemented in C++. The CPU seconds are reported from a Dell Optiplex GX620 computer with a Pentium IV 3.6 GHz processor and 1.0 GB RAM, running under the Windows XP operating system.

The computational results are reported in Table 5-3 that shows that the fortification level at each open facility given the resource constraints. Table 5-4 shows the results that the fortification level is limited to 2. That is, $y_i \leq 2, \forall i \in F$. The CPU time reported in both tables clearly show the efficiency of the monotonic branch-reduce-bound algorithm for this type of problem. Next, we start to analyze the computational results.

Table 5-3. Solutions of the 3-level model

Constraint	Resource		Fortification Level at					CPU
R	Used	Objective	**2**	**5**	**15**	**18**	**20**	second
25000	2953	2020.69	3	3	3	3	3	0.11
2952	2858	2030.01	3	3	3	3	2	0.89
2857	2401	2057.66	3	3	2	3	3	1.08
2400	2306	2068.99	3	3	2	3	2	1.19
2305	2098	2087.25	3	3	1	3	3	1.48
2097	2003	2100.18	3	3	1	3	2	1.03
2002	1825	2138.98	3	3	1	3	1	1.80
1824	1711	2276.06	3	2	1	3	1	1.94
1710	1689	2392.85	3	3	1	1	3	2.72
1688	1594	2408.41	3	3	1	1	2	2.58
1593	1551	2415.59	3	3	1	2	1	2.72
1550	1416	2455.11	3	3	1	1	1	3.33
1415	1370	2489.01	3	1	1	3	3	3.05
1369	1275	2516.42	3	1	1	3	2	2.88
1274	1097	2598.63	3	1	1	3	1	3.70
1096	1096	2797.01	3	1	1	2	3	3.34
1095	1001	2830.34	3	1	1	2	2	3.64
1000	961	2841.01	3	1	1	1	3	3.95
960	866	2875.19	3	1	1	1	2	3.17
865	823	2930.35	3	1	1	2	1	3.48
822	688	2977.74	3	1	1	1	1	3.48
687	666	3308.47	2	1	1	3	2	2.27
665	488	3503.25	2	1	1	3	1	2.56
487	409	3894.44	1	1	1	3	1	2.56
408	392	4318.51	2	1	1	2	2	2.58
391	352	4331.16	2	1	1	1	3	2.86
351	257	4462.80	2	1	1	1	2	2.09
256	214	4688.43	2	1	1	2	1	2.09
213	79	4857.74	2	1	1	1	1	1.97
78	0	5670.71	1	1	1	1	1	1.64

In Figure 5-4, we plot the objective values in these two different settings (3-level constraint and 2-level constraint) across the resource used. Unlike the result in the continuous facility fortification model, as shown in Figure 5-1, there is no piecewise linear property exhibited in this discrete version. Instead the curves are shown steeper in the earlier stage. That is, the fortification efforts help to reduce a lot of total cost at earlier stage. This indicates that realiability can be drastically improved without large

Table 5-4. Solutions of the 2-level model

Constraint	Resource		Fortification Level at					CPU
R	Used	Objective	**2**	**5**	**15**	**18**	**20**	second
25000	1309	3315.22	2	2	2	2	2	0.11
1308	1174	3402.08	2	2	2	1	2	0.88
1173	1006	3461.27	2	2	1	2	2	1.03
1005	871	3557.78	2	2	1	1	2	1.19
870	828	3666.45	2	2	1	2	1	1.02
827	693	3776.24	2	2	1	1	1	1.19
692	560	4193.94	2	1	2	1	2	1.50
559	392	4318.51	2	1	1	2	2	1.19
391	257	4462.80	2	1	1	1	2	1.34
256	214	4688.43	2	1	1	2	1	1.33
213	79	4857.74	2	1	1	1	1	1.19
78	0	5670.71	1	1	1	1	1	1.06

increases in fortification resource. After that, the whole system seems lacking the room for improvement. In fact, when R is greater than 1500, the decrease in the total cost can not justify the fortification resource. For example, when R increases from 1551 to 2003, the total cost reduced from 2415.59 to 2100.18, a net loss of 136.59.

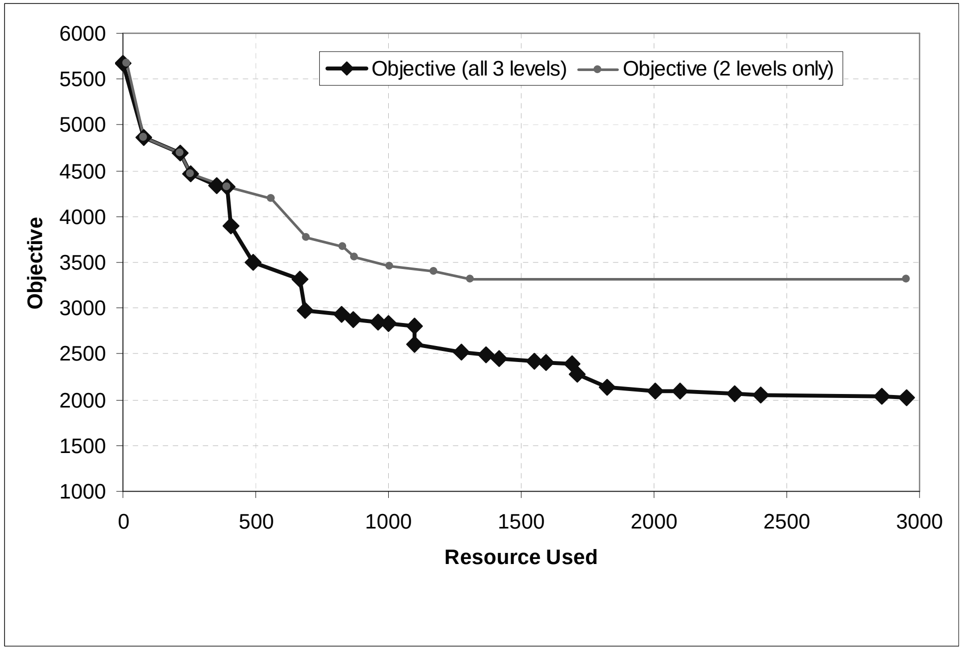


Figure 5-4. Tradeoff between objective and resource used

Comparing these two "objective" lines in Figure 5-4, we also see the benefit to have a 3-level option over a 2-level one. When the available fortification resource is very limited, both share the same objective values as there is no enough resource to fortify the facility to level 3. But as the available fortification resource increases, the 3-level model has more flexibility so that it incurs much lower total cost.

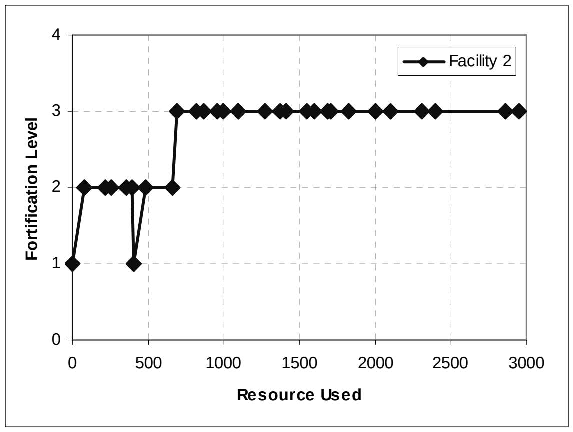
Another phenomenon different from the continuous version is the fortification level at individual facility. In the continuous version, the fortification level is more or less monotonic as shown in Figure 5-2. However, this does not exist in the discrete version as shown in Figure 5-5. As the fortification resource increases, the fortification level at an individual facility does not necessarily increase as a consequence from this combinatorial optimization. Figures D and E in Figure 5-5 clearly show this phenomenon, because facility 18 and 20 do not admit any monotonicity pattern.

5.4 Conclusions

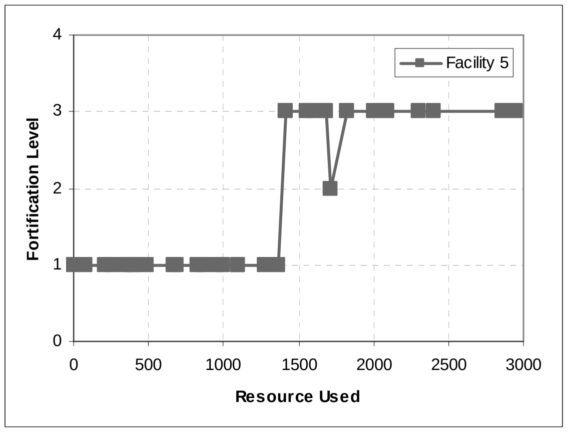
In this chapter, we present two optimization models (a continuous version and a discrete one) on how to choose facilities to fortify and to what extent they should be fortified given limited fortification resources. The objective of both models is to minimize the sum of the expected service cost and fail-to-serve penalty cost.

In the continuous version of the model, the fortification effort is dividable. We show that the model is a special case of the generalized linear multiplicative programming problem. We solve an illustrative example by the vertex numeration method, which is very effective in solving this type of problems. This example also demonstrates the multi-extremeness of the problem: several vertices achieve the minimum. Managerially, the example shows how to identify the key facilities to fortify in a system.

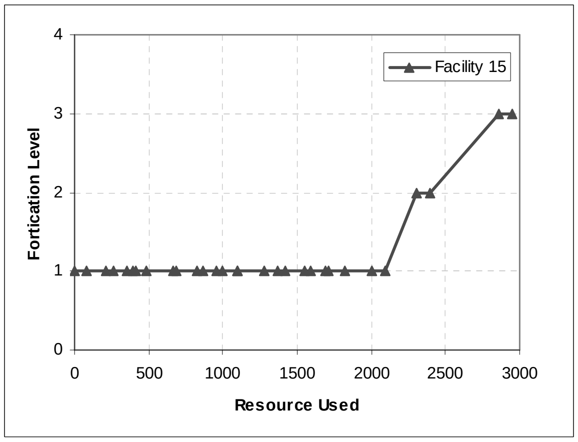
The discrete facility fortification model focuses on choosing the suitable fortification level at each facility when the fortification effort is divided into different levels. The model is shown to be a monotonicity optimization problem since the monotonicity property is inherent in both its objective and constraints. This model is therefore solved by the



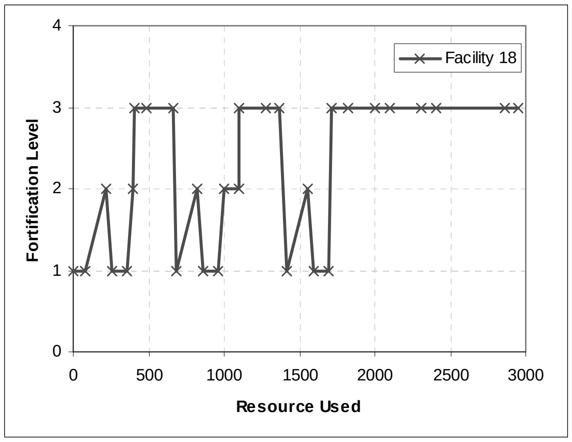
A) Facility 2



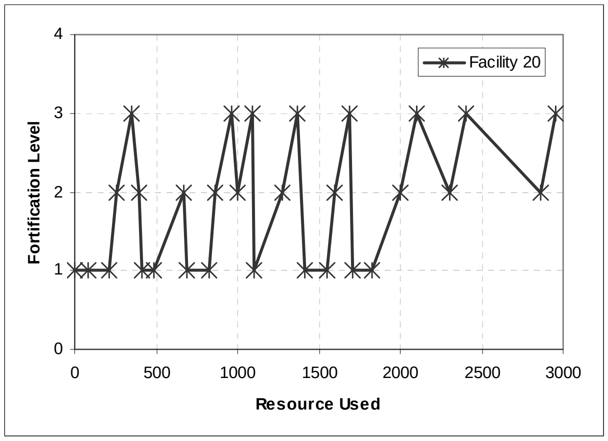
B) Facility 5



C) Facility 15



D) Facility 18



E) Facility 20

Figure 5-5. Fortification level at individual facility by resource used

monotonic branch-reduce-bound algorithm. The computational results of the illustrative example show the efficiency of the algorithm. We also analyze the tradeoff between cost deduction and fortification effort and empirically demonstrated that the tradeoff curve is steeper in the earlier stage, indicating major cost deduction can be achieved without large increases in fortification resource.

The main limitation of the current models is the assumption that the facilities are uncapacitated. Although the assumption itself is very common in the facility location models, it may be unrealistic in practice. In the capacitated case, a 'customer' of the failed facilities can be assigned to the next level backup facilities only if they have sufficient capacity to satisfy the additional demands. This may make the capacitated model very complex. We expect that the monotonic branch-reduce-bound algorithm will still be applicable. We believe that this is a valuable topic worthy of future investigation.

CHAPTER 6
CONCLUDING REMARKS

In this chapter, we summarize the various models and algorithms discussed throughout the dissertation, and point out directions for future research.

We study the impact of uncertainty on the decisions of facility location and demand assignment. The uncertainty is represented by the failure probability in each facility. Several novel models have been presented to offer solutions for both the design of initial supply chain systems and the improvement of the existing systems. We first investigate the uncapacitated reliable facility location model, whose objective is to minimize the total of opening cost, expected service cost, and expected fail-to-serve penalty cost when each facility has a site-specific failure probability. We also study a more general case that each facility has multiple levels of failure probabilities that can be chosen. If the supply chain system already exists, we propose two models for optimally allocating the fortification resource to reduce the expected service and fail-to-serve penalty cost.

The algorithms presented in this dissertation include (1) four heuristics, the sample average approximation heuristic, the greedy adding heuristic, the greedy adding and substitution heuristic, and the genetic algorithm based heuristic; (2) the approximation algorithm with a worst-case bound of 2.674; (3) the monotonic branch-reduce-bound algorithm. An in-depth theoretic treatment is provided for the approximation algorithm. All other algorithms are thoroughly tested in the computational studies. The four heuristics are used to solve the uncapacitated reliable facility location problem. The monotonic branch-reduce-bound algorithm is applied to solve the facility fortification problem as well as the system reliability problem arising from industrial or military applications.

One immediate extension is to study the capacitated version of the current reliable facility location models. Although the capacitated constraints generally pose more challenges on finding the efficient algorithms, we expect that some of the heuristics are still very efficient in finding the optimal or near optimal solutions. For the capacitated facility fortification models, we believe that the monotonic branch-reduce-bound algorithm can still be used to find the global solution.

It would be interesting to see if there exists any approximation algorithm with a constant worst-case bound for the capacitated uniform reliable facility location problem. Based on the research on approximation theory of facility location problem without considering the reliability issue, we expect that some more delicate techniques are required to develop such approximation algorithm.

Another interesting direction is to introduce different risk/reliablity measurements into the current models depending on the needs in reality. By doing so, the objective functions of current models will change accordingly. For example, one may have more interests in the cost of the worst-case scenario instead of the expected cost in the current models.

In summary, the current work we present serves as a useful foundation for further research of more complicated models and delicate algorithms.

APPENDIX A
SYSTEM RELIABILITY COMPUTATION IN CHAPTER 4

We use conditional probability to derive the expressions of reliabilities of the networks in Figure 4-3 and Figure 4-5.

Reliability of the five-component bridge network. Reliability of the network in Figure 4-3 can be written based on whether component 5 is functional or not.

$$R_s = Pr(\text{system works} \mid \text{component 5 works})r_5 + Pr(\text{system works} \mid \text{component 5 fails})(1 - r_5). \tag{A–1}$$

When component 5 works, the original network in Figure 4-3 is reduced to Figure A-1(A), which is a parallel-series system with a reliability of

$$Pr(\text{system works} \mid \text{component 5 works}) = (r_1 + r_3 - r_1 r_3)(r_2 + r_4 - r_2 r_4). \tag{A–2}$$

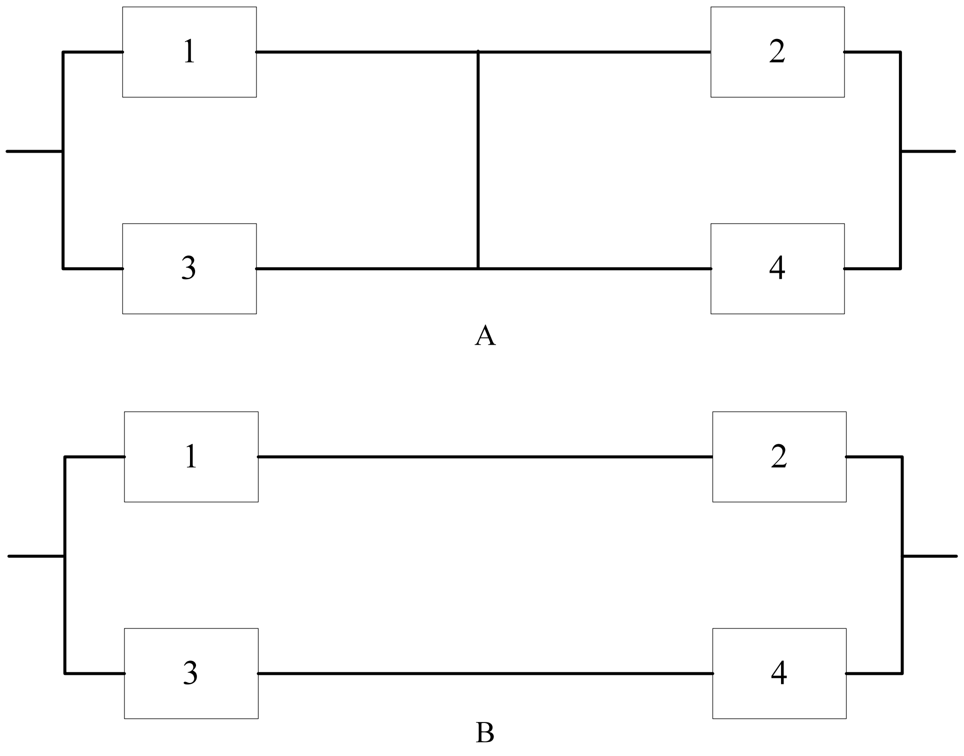


Figure A-1. Configurations based on state of component 5 in Figure 4-3: A) Component 5 works; B) Component 5 fails

Similarly, when component 5 fails, the original network in Figure 4-3 is reduced to Figure A-1(B), which is a series-parallel system with a reliability of

$$Pr(\text{system works} \mid \text{component 5 fails}) = (r_1r_2 + r_3r_4 - r_1r_2r_3r_4). \tag{A–3}$$

Substitution of equations A–2 and A–3 into equation A–1 yields the reliability of the five-component bridge network depicted in Figure 4-3:

$$R_s = (r_1 + r_3 - r_1r_3)(r_2 + r_4 - r_2r_4)r_5 + (r_1r_2 + r_3r_4 - r_1r_2r_3r_4)(1 - r_5) \tag{A–4}$$

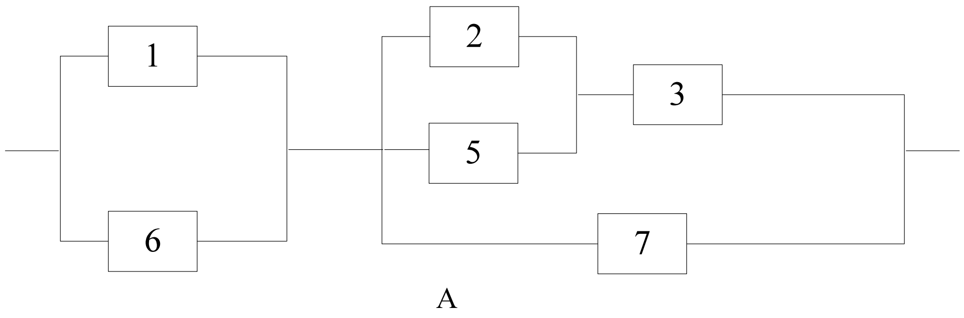
Reliability of the seven-link ARPA network. Following the notation in Example 4, we assume that each block from blocks 1 to 5 in Figure 4-5 represents a subsystem. Blocks 6 and 7 are individual components. Recall that $R_i = 1 - (1 - r_i)^{x_i}, \forall i = 1, \ldots, 5$, $Q_i = 1 - R_i, \forall i = 1, \ldots, 7$. Reliability of the network in Figure 4-5 can be written based on whether subsystem 4 is functional or not.

$$R_s = Pr(\text{system works} \mid \text{subsystem 4 works})R_4 + Pr(\text{system works} \mid \text{subsystem 4 fails})(1 - R_4). \tag{A–5}$$

When subsystem 4 works, the original network in Figure 4-5 is reduced to Figure A-2(A), whose reliability can be obtained by applying parallel and series reductions:

$$Pr(\text{system works} \mid \text{subsystem 4 works}) = (1 - Q_1Q_6)\{1 - Q_7[1 - R_3(1 - Q_2Q_5)]\} \tag{A–6}$$

When subsystem 4 fails, the original network in Figure 4-5 is reduced to Figure A-2(B). A series reduction on subsystems 1 and 2 produces a super-component, which helps to map the topology in Figure A-2(B) to the five-component bridge network in Figure 4-3. After the mapping, we can directly use the result of equation A–4 by replacing r_1 with R_1R_2, r_2



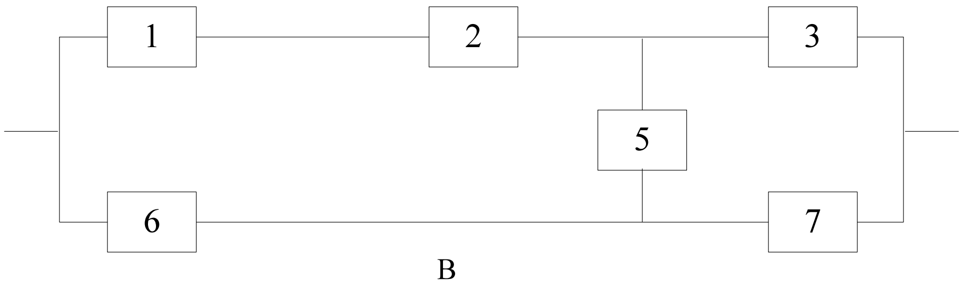


Figure A-2. Configurations based on state of subsystem 4 in Figure 4-5: A) Subsystem 4 works; B) Subsystem 4 fails

with R_3, r_3 with R_6, r_4 with R_7, r_5 with R_5:

$$Pr(\text{system works} \mid \text{subsystem 4 fails}) =$$
$$(R_1R_2 + R_6 - R_1R_2R_6)(R_3 + R_7 - R_3R_7)R_5 + (R_1R_2R_3 + R_6R_7 - R_1R_2R_3R_6R_7)(1 - R_5) \tag{A–7}$$

Substitution of equations A–6 and A–7 into equation A–5 yields the reliability of the five-component bridge network depicted in Figure 4-5:

$$\begin{aligned} R_s &= (1 - Q_1Q_6)\{1 - Q_7[1 - R_3(1 - Q_2Q_5)]\}R_4 + (R_1R_2 + R_6 - R_1R_2R_6)(R_3 + R_7 - R_3R_7)R_5 \\ &\quad + (R_1R_2R_3 + R_6R_7 - R_1R_2R_3R_6R_7)(1 - R_5)(1 - R_4). \end{aligned} \tag{A–8}$$

Equation A–8 can be reformulated as the following one:

$$\begin{aligned} R_s = R_6R_7 &+ R_1R_2R_3(Q_6 + R_6Q_7) + R_1R_4R_7Q_6(Q_2 + R_2Q_3) \\ &+ R_3R_5R_6Q_7(Q_1 + R_1Q_2) + R_1R_2R_5R_7Q_3Q_4Q_6 \\ &+ R_2R_3R_4R_6Q_1Q_5Q_7 + R_1R_3R_4R_5Q_2Q_6Q_7. \end{aligned} \tag{A–9}$$

APPENDIX B
DATASET USED IN CHAPTER 2

The meaning of each column in Table B-1 is provided as follows: $\#i$ denotes the facility name; (x, y) is the coordinates, d_i is the demand; f_i is the fixed cost; r_i is the penalty cost; and p_i is the failure probability.

Table B-1. Dataset of URFLP-SFP

$\#i$	x	y	d_i	f_i	r_i	p_i	$\#i$	x	y	d_i	f_i	r_i	p_i
1	0.82	0.18	957	938	5.32	0.81	51	0.63	0.04	486	971	3.14	0.63
2	0.54	0.7	202	642	1.9	0.39	52	0.53	0.32	548	1023	0.31	0.94
3	0.91	0.72	186	1230	3.11	0.42	53	0.89	0.99	870	754	0.66	0.76
4	0.15	0.31	635	1008	1.83	0.36	54	0.02	0.19	335	734	0.22	0.97
5	0.74	0.16	737	1279	1.34	0.28	55	0.51	0.32	446	1249	3.93	0.91
6	0.58	0.92	953	1431	2.3	0.83	56	0.53	0.06	198	1371	3.77	0.31
7	0.6	0.09	450	1187	7.96	0.98	57	0.81	0.86	212	1489	9.92	0.58
8	0.37	0.19	188	1044	3.42	1	58	0.53	0.36	903	863	4.06	0.29
9	0.7	0.52	206	1466	9.05	0.86	59	0.89	0.58	594	521	1.64	0.15
10	0.22	0.4	995	989	4.56	0.55	60	0.87	0.56	250	865	5.11	0.21
11	0.5	0.45	429	948	9.87	0.58	61	0.91	0.16	472	1464	2.64	0.43
12	0.3	0.52	528	585	0.53	0.79	62	0.32	0.15	244	730	6.05	0.09
13	0.95	0.2	570	923	3.41	0.46	63	0.37	0.37	353	1034	4.27	0.78
14	0.65	0.07	938	758	8.98	0.15	64	0.38	0.73	183	585	6.18	0.23
15	0.53	0.11	726	552	3.53	0.48	65	0.96	0.34	749	782	9.13	0.6
16	0.95	0.95	533	1471	1.64	0.7	66	0.15	0.76	200	985	6.63	0.42
17	0.15	0.13	565	930	1.36	0.98	67	0.15	0.48	321	662	7.02	0.1
18	0.31	0.4	322	1103	5.1	0.2	68	0.99	0	650	904	8.73	0.16
19	0.98	0.73	326	586	1.22	0.49	69	0.47	0.28	946	1242	3.92	0.52
20	0.59	0.04	663	812	6.95	0.3	70	0.84	0.16	143	513	8.16	0.79
21	0.46	0.21	952	850	9.57	0.37	71	0.71	0.9	565	1117	1.25	0.84

continued from previous page

$\#i$	x	y	d_i	f_i	r_i	p_i	$\#i$	x	y	d_i	f_i	r_i	p_i
22	0.77	0.44	919	561	1.49	0.38	72	0.46	0.86	11	928	4.44	0.26
23	0.87	0.79	292	750	6.95	0.6	73	0.09	0.74	374	1028	2.65	0.33
24	0.69	0.15	48	956	2.29	0.71	74	0.71	0.78	284	522	1.45	0.35
25	0.24	0.28	581	1456	2.9	0.3	75	0.27	0.04	598	609	9.63	0.22
26	0.84	0.73	659	764	5.05	0.99	76	0.25	0.07	720	541	3.72	0.25
27	0.49	0.24	986	1005	8.81	0.23	77	0.57	0.18	457	1493	2.38	0.43
28	0.56	0.77	486	1121	3	0.81	78	0.96	0.49	213	736	7.09	0.49
29	0.38	0.05	915	1065	6.94	0.4	79	0.83	0.21	550	1452	8.3	0.43
30	0.43	0.22	282	1494	0.88	0.68	80	0.72	0.49	418	758	2.55	0.57
31	0.61	0.73	310	605	2.66	0.39	81	0.69	0.5	863	1147	1.53	0.93
32	0.48	0.88	980	1304	3.32	0.86	82	0.22	0.89	368	976	3.5	0.52
33	0.81	0.75	134	1073	0.66	0.28	83	0.37	0.88	282	1180	7.15	0.95
34	0.13	0.71	20	1107	3.08	0.62	84	0.36	0.82	811	1045	9.07	0.3
35	0.41	0.34	151	651	2.64	0.01	85	0.11	0.1	866	1021	2.46	0.37
36	0.72	0.14	615	800	3.84	0.35	86	0.77	0.69	895	1165	3.19	0.44
37	0.3	0.28	369	1479	1.74	0.22	87	0.16	0.09	959	1250	5.52	0.66
38	0.02	1	875	1278	1.14	0	88	0.75	0.63	375	1406	5.28	0.06
39	0.62	0.9	73	1289	2.76	0.42	89	0.16	0.41	711	515	7.34	0.46
40	0.8	0.06	776	510	3.36	0.69	90	0.01	0.21	208	923	3.99	0.87
41	0.1	0.98	342	536	5.61	0	91	0.51	0.76	954	1378	4.06	0.31
42	0.15	0.13	929	1022	9.77	0.73	92	0.98	0.32	843	733	7.77	0.33
43	0.48	0.44	445	594	4.32	0.83	93	0.55	0.39	905	545	0.08	0.74
44	0.83	0.22	684	1466	9.61	0.28	94	0.36	0.63	729	1047	8.47	0.84
45	0.82	0.39	643	677	1.45	0.71	95	0.18	0.75	382	1365	6.23	0.38
46	0.67	0.53	771	1334	7.46	0.74	96	0.09	0.46	91	513	0.53	0.16
47	0.03	0.73	181	1445	1.03	0.35	97	0.18	0.67	991	1338	6.62	0.92
48	0.26	0.39	926	691	7.6	0.26	98	0.1	0.38	644	1341	3.13	0.11
49	0.59	0.56	733	1001	0.15	0.08	99	0.25	0.66	539	1256	0.5	0.41
50	0.22	0.66	326	1244	5.93	0.98	100	0.68	0.49	294	1168	7.27	0.66

The meaning of each column in Table B-2 is provided as follows: $\#i$ denotes the facility name; (x, y) is the coordinates, d_i is the demand; r_i is the penalty cost; and f_i, p_i $(i = 1, 2, 3)$ are the investment level and its corresponding failure probability.

Table B-2. Dataset of URFLP-MFP: 3-level

$\#i$	x	y	d_i	r_i	f_1	p_1	f_2	p_2	f_3	p_3
1	0.82	0.18	957	5.32	938	0.81	954	0.44	1260	0.15
2	0.54	0.7	202	1.9	642	0.55	721	0.39	1330	0.02
3	0.91	0.72	186	3.11	1125	0.63	1230	0.42	1355	0.19
4	0.15	0.31	635	1.83	772	1	1008	0.9	1440	0.36
5	0.74	0.16	737	1.34	665	0.85	1279	0.45	1393	0.28
6	0.58	0.92	953	2.3	890	0.9	1034	0.83	1431	0.61
7	0.6	0.09	450	7.96	620	0.98	1187	0.48	1394	0.06
8	0.37	0.19	188	3.42	503	1	703	0.75	1044	0.49
9	0.7	0.52	206	9.05	1231	0.86	1278	0.65	1466	0.24
10	0.22	0.4	995	4.56	989	1	1037	0.55	1455	0.54
11	0.5	0.45	429	9.87	948	0.87	1082	0.6	1422	0.58
12	0.3	0.52	528	0.53	551	0.84	585	0.79	1303	0.39
13	0.95	0.2	570	3.41	682	0.46	923	0.32	999	0.04
14	0.65	0.07	938	8.98	758	0.78	987	0.5	1497	0.15
15	0.53	0.11	726	3.53	552	0.75	855	0.63	1407	0.48
16	0.95	0.95	533	1.64	791	0.7	1471	0.66	1488	0.47
17	0.15	0.13	565	1.36	930	0.98	958	0.8	1227	0.35
18	0.31	0.4	322	5.1	694	0.52	829	0.48	1103	0.2
19	0.98	0.73	326	1.22	586	0.95	975	0.49	1127	0.15
20	0.59	0.04	663	6.95	634	0.3	812	0.24	907	0.22
21	0.46	0.21	952	9.57	850	0.37	932	0.18	980	0.14
22	0.77	0.44	919	1.49	561	0.92	742	0.76	1290	0.38
23	0.87	0.79	292	6.95	750	0.6	1213	0.59	1311	0.24
24	0.69	0.15	48	2.29	684	0.91	772	0.79	956	0.71
25	0.24	0.28	581	2.9	1040	0.62	1153	0.3	1456	0.23

continued from previous page

$\#i$	x	y	d_i	r_i	f_1	p_1	f_2	p_2	f_3	p_3
26	0.84	0.73	659	5.05	764	0.99	941	0.2	1035	0.04
27	0.49	0.24	986	8.81	910	0.94	1005	0.25	1058	0.23
28	0.56	0.77	486	3	928	0.81	1079	0.72	1121	0.35
29	0.38	0.05	915	6.94	1040	0.81	1065	0.4	1108	0.16
30	0.43	0.22	282	0.88	771	0.76	1028	0.68	1494	0.04
31	0.61	0.73	310	2.66	605	0.81	748	0.41	1381	0.39
32	0.48	0.88	980	3.32	1106	0.86	1304	0.21	1400	0.06
33	0.81	0.75	134	0.66	1073	0.88	1153	0.34	1315	0.28
34	0.13	0.71	20	3.08	905	0.79	1107	0.62	1260	0.62
35	0.41	0.34	151	2.64	651	0.87	652	0.25	1066	0.01
36	0.72	0.14	615	3.84	800	0.98	1298	0.57	1423	0.35
37	0.3	0.28	369	1.74	723	0.7	1351	0.51	1479	0.22
38	0.02	1	875	1.14	890	0.81	1029	0.04	1278	0
39	0.62	0.9	73	2.76	1104	0.88	1289	0.42	1485	0.37
40	0.8	0.06	776	3.36	510	0.91	981	0.69	1341	0.59
41	0.1	0.98	342	5.61	536	0.63	1157	0.06	1351	0
42	0.15	0.13	929	9.77	1022	0.73	1064	0.67	1157	0.33
43	0.48	0.44	445	4.32	594	0.83	1289	0.76	1386	0.43
44	0.83	0.22	684	9.61	1130	0.68	1192	0.28	1466	0.09
45	0.82	0.39	643	1.45	518	0.71	677	0.38	1072	0.28
46	0.67	0.53	771	7.46	892	0.74	1334	0.72	1483	0.45
47	0.03	0.73	181	1.03	1055	0.97	1156	0.63	1445	0.35
48	0.26	0.39	926	7.6	691	0.56	865	0.26	1312	0.23
49	0.59	0.56	733	0.15	1001	0.72	1058	0.61	1450	0.08
50	0.22	0.66	326	5.93	699	0.98	1244	0.42	1472	0.04
51	0.63	0.04	486	3.14	971	0.68	1154	0.63	1493	0.49
52	0.53	0.32	548	0.31	853	0.99	1023	0.94	1207	0.28
53	0.89	0.99	870	0.66	754	0.84	787	0.76	1271	0.61
54	0.02	0.19	335	0.22	611	0.97	734	0.83	1214	0.51
55	0.51	0.32	446	3.93	1217	0.91	1249	0.88	1367	0.05
56	0.53	0.06	198	3.77	742	0.69	1025	0.49	1371	0.31

continued from previous page

$\#i$	x	y	d_i	r_i	f_1	p_1	f_2	p_2	f_3	p_3
57	0.81	0.86	212	9.92	1271	0.58	1446	0.4	1489	0.38
58	0.53	0.36	903	4.06	863	0.75	1078	0.29	1354	0.05
59	0.89	0.58	594	1.64	521	0.64	1071	0.44	1100	0.15
60	0.87	0.56	250	5.11	865	0.95	1273	0.6	1348	0.21
61	0.91	0.16	472	2.64	974	0.92	1220	0.43	1464	0.22
62	0.32	0.15	244	6.05	689	0.88	730	0.58	810	0.09
63	0.37	0.37	353	4.27	745	0.89	821	0.78	1034	0.46
64	0.38	0.73	183	6.18	585	0.79	1026	0.78	1259	0.23
65	0.96	0.34	749	9.13	782	0.84	1219	0.6	1392	0.13
66	0.15	0.76	200	6.63	561	0.93	985	0.42	1068	0.22
67	0.15	0.48	321	7.02	569	0.68	662	0.5	713	0.1
68	0.99	0	650	8.73	880	0.79	904	0.26	1424	0.16
69	0.47	0.28	946	3.92	1190	0.64	1242	0.58	1332	0.52
70	0.84	0.16	143	8.16	513	0.79	993	0.48	1355	0.14
71	0.71	0.9	565	1.25	601	1	945	0.84	1117	0.13
72	0.46	0.86	11	4.44	904	0.53	928	0.32	982	0.26
73	0.09	0.74	374	2.65	691	0.48	1028	0.33	1291	0.32
74	0.71	0.78	284	1.45	522	0.94	818	0.68	1246	0.35
75	0.27	0.04	598	9.63	609	0.85	771	0.22	1026	0.11
76	0.25	0.07	720	3.72	541	0.25	704	0.14	1012	0.09
77	0.57	0.18	457	2.38	986	0.75	1307	0.43	1493	0.19
78	0.96	0.49	213	7.09	666	0.77	736	0.66	861	0.49
79	0.83	0.21	550	8.3	564	0.74	796	0.43	1452	0.15
80	0.72	0.49	418	2.55	588	0.78	619	0.57	758	0.39
81	0.69	0.5	863	1.53	1147	0.93	1299	0.77	1310	0.71
82	0.22	0.89	368	3.5	510	0.65	976	0.52	1070	0.39
83	0.37	0.88	282	7.15	598	0.95	871	0.73	1180	0.07
84	0.36	0.82	811	9.07	674	0.58	1045	0.4	1166	0.3
85	0.11	0.1	866	2.46	568	0.63	1021	0.37	1192	0.24
86	0.77	0.69	895	3.19	665	0.93	1015	0.44	1165	0.25
87	0.16	0.09	959	5.52	980	0.92	1250	0.68	1412	0.66

continued from previous page

$\#i$	x	y	d_i	r_i	f_1	p_1	f_2	p_2	f_3	p_3
88	0.75	0.63	375	5.28	522	0.59	1206	0.06	1406	0.04
89	0.16	0.41	711	7.34	515	0.63	707	0.48	1415	0.46
90	0.01	0.21	208	3.99	923	0.87	1049	0.84	1281	0.32
91	0.51	0.76	954	4.06	739	0.79	1186	0.31	1378	0.28
92	0.98	0.32	843	7.77	733	0.52	834	0.33	1335	0.18
93	0.55	0.39	905	0.08	545	0.74	869	0.61	1178	0.61
94	0.36	0.63	729	8.47	1047	0.84	1098	0.66	1460	0.6
95	0.18	0.75	382	6.23	538	0.62	613	0.59	1365	0.38
96	0.09	0.46	91	0.53	513	0.67	521	0.26	1448	0.16
97	0.18	0.67	991	6.62	723	0.92	1119	0.91	1338	0.41
98	0.1	0.38	644	3.13	628	0.54	822	0.46	1341	0.11
99	0.25	0.66	539	0.5	848	0.49	942	0.48	1256	0.41
100	0.68	0.49	294	7.27	1069	0.99	1108	0.88	1168	0.66

APPENDIX C
DATASET USED IN CHAPTER 5

The meaning of each column in Table C-1 is provided as follows: $\#i$ denotes the facility name; (x, y) is the coordinates, d_i is the demand; f_i is the fixed cost; r_i is the penalty cost.

Table C-1. Dataset of DFFM

$\#i$	x	y	d_i	r_i	$\#i$	x	y	d_i	r_i
Open Facilities:					2, 5, 15, 18, 20				
1	0.82	0.18	957	5.32	51	0.63	0.04	486	3.14
2	0.54	0.7	202	1.9	52	0.53	0.32	548	0.31
3	0.91	0.72	186	3.11	53	0.89	0.99	870	0.66
4	0.15	0.31	635	1.83	54	0.02	0.19	335	0.22
5	0.74	0.16	737	1.34	55	0.51	0.32	446	3.93
6	0.58	0.92	953	2.3	56	0.53	0.06	198	3.77
7	0.6	0.09	450	7.96	57	0.81	0.86	212	9.92
8	0.37	0.19	188	3.42	58	0.53	0.36	903	4.06
9	0.7	0.52	206	9.05	59	0.89	0.58	594	1.64
10	0.22	0.4	995	4.56	60	0.87	0.56	250	5.11
11	0.5	0.45	429	9.87	61	0.91	0.16	472	2.64
12	0.3	0.52	528	0.53	62	0.32	0.15	244	6.05
13	0.95	0.2	570	3.41	63	0.37	0.37	353	4.27
14	0.65	0.07	938	8.98	64	0.38	0.73	183	6.18
15	0.53	0.11	726	3.53	65	0.96	0.34	749	9.13
16	0.95	0.95	533	1.64	66	0.15	0.76	200	6.63
17	0.15	0.13	565	1.36	67	0.15	0.48	321	7.02
18	0.31	0.4	322	5.1	68	0.99	0	650	8.73
19	0.98	0.73	326	1.22	69	0.47	0.28	946	3.92
20	0.59	0.04	663	6.95	70	0.84	0.16	143	8.16
21	0.46	0.21	952	9.57	71	0.71	0.9	565	1.25
22	0.77	0.44	919	1.49	72	0.46	0.86	11	4.44

continued from previous page

$\#i$	x	y	d_i	r_i	$\#i$	x	y	d_i	r_i
23	0.87	0.79	292	6.95	73	0.09	0.74	374	2.65
24	0.69	0.15	48	2.29	74	0.71	0.78	284	1.45
25	0.24	0.28	581	2.9	75	0.27	0.04	598	9.63
26	0.84	0.73	659	5.05	76	0.25	0.07	720	3.72
27	0.49	0.24	986	8.81	77	0.57	0.18	457	2.38
28	0.56	0.77	486	3	78	0.96	0.49	213	7.09
29	0.38	0.05	915	6.94	79	0.83	0.21	550	8.3
30	0.43	0.22	282	0.88	80	0.72	0.49	418	2.55
31	0.61	0.73	310	2.66	81	0.69	0.5	863	1.53
32	0.48	0.88	980	3.32	82	0.22	0.89	368	3.5
33	0.81	0.75	134	0.66	83	0.37	0.88	282	7.15
34	0.13	0.71	20	3.08	84	0.36	0.82	811	9.07
35	0.41	0.34	151	2.64	85	0.11	0.1	866	2.46
36	0.72	0.14	615	3.84	86	0.77	0.69	895	3.19
37	0.3	0.28	369	1.74	87	0.16	0.09	959	5.52
38	0.02	1	875	1.14	88	0.75	0.63	375	5.28
39	0.62	0.9	73	2.76	89	0.16	0.41	711	7.34
40	0.8	0.06	776	3.36	90	0.01	0.21	208	3.99
41	0.1	0.98	342	5.61	91	0.51	0.76	954	4.06
42	0.15	0.13	929	9.77	92	0.98	0.32	843	7.77
43	0.48	0.44	445	4.32	93	0.55	0.39	905	0.08
44	0.83	0.22	684	9.61	94	0.36	0.63	729	8.47
45	0.82	0.39	643	1.45	95	0.18	0.75	382	6.23
46	0.67	0.53	771	7.46	96	0.09	0.46	91	0.53
47	0.03	0.73	181	1.03	97	0.18	0.67	991	6.62
48	0.26	0.39	926	7.6	98	0.1	0.38	644	3.13
49	0.59	0.56	733	0.15	99	0.25	0.66	539	0.5
50	0.22	0.66	326	5.93	100	0.68	0.49	294	7.27

REFERENCES

[1] M. Agarwal, R. Gupta, Penalty function approach in heuristic algorithms for constrainted redundancy reliability optimization, IEEE Transactions on Reliability 54(3) (2005) 549–558.

[2] O. Alp, E. Erkut, Z. Drezner, An efficient genetic algorithm for the p-median problem, Annals of Operations Research 122 (2003) 21–42.

[3] F. Bellanti, F. Della Croce, R. Tadei, A greedy-based neighborhood search approach to a nurse rostering problem, European Journal of Operational Research 153(1) (2004) 28–40.

[4] O. Berman, Z. Drezner, G. O. Wesolosky, Locating service facilites whose reliability is distance dependent, Computers & Operations Research 30 (2003) 1683–1695.

[5] O. Berman, D. Krass, M. B. C. Menezes, Facility reliability isuues in network p-median problems: strategic centralization and co-location effects, Operations ResearchTo appear.

[6] A. F. Bumb, J. van Ommeren, An approximation algorithm for a facility location problem with stochastic demands and inventory, Operations Research Letters 34(3) (2006) 257–263.

[7] G. Chen, M. Daskin, Z. J. Shen, S. Uryasev, A new model for stochastic facility locations, Naval Research Logistics 53(7) (2006) 617–626.

[8] M. S. Daskin, Network and Discrete Location: Models, Algorithms, and Applications, Wiley, New York, 1995.

[9] M. S. Daskin, K. Hogfan, C. ReVelle, α-reliable p-minimax regres: a new model for strategic facility locaton modeling, Location Science 5(4) (1997) 227–246.

[10] P. M. França, H. P. L. Luna, Soving stochastic transportation-location problems by generalized benders decomposition, Transportatoin Science 16(2) (1982) 113–126.

[11] M. Gen, R. Cheng, Genetic Algorithms and Engineering Design, John Wiley and Sons, New York, 1997.

[12] P. M. Ghare, D. C. Montgomery, W. C. Tuner, Optimal interdiction policy for a flow network, Naval Research Logistics Quarterly 18(1) (1971) 37–45.

[13] D. Ghosh, Neighborhood search heuristics for the uncapacitated facility location problem, European Journal of Operational Research 150 (2003) 150–162.

[14] D. Goldberg (ed.), Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[15] K. Gopal, K. K. Aggarwal, J. S. Gupta, A new method for solving reliability optimization problem, IEEE Transactions on Reliability R-29 (1980) 36–38.

[16] S. Guha, Approximation algorithms for facility location problems, phD thesis, Stanford University (2000).

[17] S. Guha, A. Meyerson, K. Munagala, A constant factor approximation algorithm for the fault-tolerant facility location problem, Journal of Algorithms 48 (2003) 429–440.

[18] M. Hikita, Y. Nakagawa, K. Nakashima, H. Narihisa, Reliability optimization of systems by a surrogate-constraints algorithm, IEEE Transactions on Reliability R-41(3) (1992) 473–480.

[19] R. Horst, P. M. Pardalos, N. V. Thoai, Intorduction to Global Optimization, Kluwer Academic Publishers, 2001.

[20] Y. C. Hsieh, T. C. Chen, B. D. L., Genetic algorithms for reliability design problems, Microelectronics Reliability 38(10) (1998) 1599–1605.

[21] K. Jain, M. Mahdian, E. Markakis, A. Saberi, V. Vazirani, An improved approximation algorithm for uncapacitated facility location problem with penalties, Journal of the ACM 50 (2003) 795–824.

[22] J. H. Jaramillo, J. Bhadury, R. Batta, On the use of genetic algorithms to solve location problems, Computers & Operations Research 29 (2002) 761–779.

[23] J. H. Kim, B. J. Yum, A heuristic method for solving redundancy optimization problems in complex systems, IEEE Transactions on Reliability 42(4) (1993) 572–578.

[24] A. J. Kleywegt, A. Shapriro, T. Homem-de mello, The sample average approximation method for stochastic discrete optimization, SIAM Journal of Optimization 12(2) (2001) 479–502.

[25] H. Konno, Y. Yajima, T. Matsui, An outer approximatin method for minimizing the product of several convex functions on a convex set, Journal of Global Optimization 3(3) (1993) 325–335.

[26] W. Kuo, H. Lin, Z. Xu, W. Zhang, Reliability optimization with the lagrange multiplier and branch-and-bound technique, IEEE Transactions on Reliability R-36 (1987) 624–630.

[27] W. Kuo, V. R. Prasad, An annotated overview of system-reliability optimization, IEEE Transactions on Reliability 49(2) (2000) 176–187.

[28] W. Kuo, V. R. Prasad, F. A. Tillman, C. L. Hwang, Optimal reliability design: fundamentals and applications, Cambridge University Press, 2001.

[29] D. Li, X. L. Sun, K. Mchinnon, An exact method for reliability optimization in complex systems, Annals of Operations Research 133 (2005) 129–148.

[30] C. Lim, J. C. Smith, Algorithms for discrete and continuous multicommodity flow network interdiction problems, IIE Transactions 39(1) (2007) 15–26.

[31] X. J. Liu, T. Umegaki, Y. Yamamoto, Heuristic methods for linear multiplicative programs, Journal of Global Optimization 15(4) (1999) 433–447.

[32] M. Mahdian, Y. Ye, J. Zhang, Approximation algorithms for metric facility location problems, SIAM Journal on Computing 36 (2006) 411–432.

[33] A. McMasters, M. Thomas, Optimal interdiction of a suppy network, Naval Research Logistics Quarterly 17 (1970) 261–268.

[34] R. Mirchandani, R. Francis (eds.), Discrete Location Theory, Wiley, New York, 1990.

[35] K. B. Misra, U. Sharma, An efficient algorithm to solve integer-porgramming problems arising in system-reliability design, IEEE Transactions on Reliability 40(1) (1991) 81–91.

[36] Y. Nakagawa, Nakashima, A heuristic method for determining optimal reliability allocations, IEEE Transactions on Reliability R-26 (1977) 156–161.

[37] Y. Nakagawa, K. Nakashima, Y. Hottori, Optimal reliability allocation by branch-and-bound technique, IEEE Transactions on Reliability R-27 (1978) 31–38.

[38] S. H. Owen, M. S. Daskin, Strategic facility location: A review, European Journal of Operational Research 111 (1998) 423–447.

[39] C. R. Reeves (ed.), Modern Heuristic Techniques for Combinatorial Problems, Orient Longman, Hyderabad, 1993.

[40] A. Rubinov, H. Tuy, H. Mays, An algorithm for monotonic global optimization problems, Optimization 49 (2001) 205–221.

[41] H. S. Ryoo, N. Sahinidis, Global optimization of multiplicative programs, Journal of Global Optimization 26 (2003) 387–418.

[42] T. Santoso, S. Ahmed, M. Goetshalckx, A. Shapriro, A stochastic programming approach for supply chain network design under uncertainty, European Journal of Operational Research 167 (2005) 96–115.

[43] P. M. Scaparra, R. L. Church, An optimal approach for the interdiction median problem with fortification, working paper, Kent business School (2005).

[44] P. M. Scaparra, R. L. Church, A bilevel mixed integer program for critical infrastructure protection planning, Computers and Operations ResearchTo appear.

[45] Z. J. Shen, A profit maximizing supply chain network design model, Operations Research Letters 34 (2005) 673–682.

[46] D. Shmoys, E. Tardos, K. Aasdal, Approximation algortihms for facility locatoin problems, in: Proceedings of 29th ACM STOC, 1997.

[47] L. V. Snyder, Facility location under uncertainty: a review, IIE Transactions 38 (2006) 537–554.

[48] L. V. Snyder, M. S. Daskin, Reliability models for facility location: The expected failure cost case, Transportation Science 39 (2005) 400–416.

[49] L. V. Snyder, P. M. Scaparra, M. S. Daskin, R. L. Church, Planning for disruptions in supply chain networks, in: TutORials in Operations Research, INFORMS, 2006.

[50] X. L. Sun, J. L. Li, A branch-and-bound based method for solving monotone optimization problems, Journal of Global Optimization 35 (2006) 367–385.

[51] X. L. Sun, K. Mchinnon, D. Li, A convexification method for a class of global optimization problems with applications to reliability optimization, Journal of Global Optimization 21 (2001) 185–199.

[52] C. Swamy, D. Shmoys, Approximation algorithms for 2-stage stochastic optimization problems, ACM SIGACT News 37(1) (2006) 33–46.

[53] F. A. Tillman, C. L. Hwang, L. T. Fan, S. A. Balale, Systems reliability subject to multiple nonlinear constraints, IEEE Transactions on Reliability R(17) (1968) 153–157.

[54] F. A. Tillman, C. L. Hwang, W. Kuo, Determining component reliability and redundancy for optimum system reliability, IEEE Transactions on Reliability 26(3)) (1977) 162–165.

[55] F. A. Tillman, C. L. Hwang, W. Kuo, Determining componetn reliability and redundancyfor optimum system reliability, IEEE Transactions on Reliability R-26 (1977) 162–165.

[56] H. Tuy, Monotonic optimization: problems and sulution approaches, SIAM Journal of Optimization 11(2) (2000) 464–494.

[57] H. Tuy, F. Al-Khayyal, Monotonic optimization revisited, working paper, http://www.math.ac.vn/library/download/e-print/03/pdf/htuy23.pdf. Accessed in Feburary, 2007. (2004).

[58] H. Tuy, M. Minoux, N. T. Hoai-Phuong, Discrete monotonic optimization with application to a discrete location problem, SIAM Journal on Optimization 17(1) (2006) 78–97.

[59] H. Tuy, P. T. Thach, H. Konna, Optimization of polynomial fractional functions, Journal of Global Optimization 29 (2004) 19–44.

[60] D. Tuzun, L. I. Burke, A two-phase tabu search approach to the location routing problem, European Journal of Operational Research 116 (1999) 87–99.

[61] B. Verweij, S. Ahmed, A. J. Kleywegt, G. Nemhauser, A. Shapriro, The sample average approximation method applied to stochastic routing problems: a computational study, Computational Optimization and Applications 24 (2003) 289–333.

[62] R. K. Wood, Deterministic network interdiction, Mathematical and Computer Modelling 17(2) (1993) 1–18.

[63] T. Zeng, J. E. Ward, The stochastic location-assignment problem on a tree, Annals of Operations Research 136 (2005) 81–97.

BIOGRAPHICAL SKETCH

Lezhou Zhan was born in Zhejiang, China, in the year of the Horse. He is also known as Roger whose pronunciation is similar to Lezhou in his hometown dialect. Prior to college, he graduated from Yueqing Middle School in 1997. He received his Bachelor of Science in Applied Mathematics and his Bachelor of Science in Business Administration and Engineering Management from Chongqing University in 2001. Before he transferred to the University of Florida in the fall of 2002, he studied scientific computation at the Hong Kong University of Science and Technology in a Master of Philosophy program. He served as Secretary-General of FACSS, a Chinese student association at UF, from 2003 to 2004. He earned his Master of Science and Doctor of Philosophy in Industrial and Systems Engineering from the University of Florida in May, 2004 and August, 2007 respectively. His current research interests include reliable supply chain design, auction mechanism design, operations research models in airline applications, and system reliability optimization. His work has been presented in various conferences, book chapters, and journals, including *Proceedings of the 2005 Winter Simulation Conference*, *Proceedings of the 2005 IIE Research Conference* and *Production and Operations Management.* He is a member of INFORMS, SIAM, and IIE.