12-2017

Efficient Algorithms for Solving Facility Problems with Disruptions

Kaike Zhang
University of Tennessee, kzhang7@vols.utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Kaike Zhang entitled "Efficient Algorithms for Solving Facility Problems with Disruptions." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Industrial Engineering.

Xueping Li, Major Professor

We have read this dissertation and recommend its acceptance:

Bogdan Bichescu, Mingzhou Jin, John E. Kobza

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Efficient Algorithms for Solving Facility Problems with Disruptions

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Kaike Zhang

December 2017

This dissertation is dedicated to my parents, Longxiu Chen and Wen Zhang, for their love, support and encouragement.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Xueping Li, for the precious opportunity to be part of his research group. His excellent guidance, caring and patience provided me with an excellent atmosphere for doing research.

To my dissertation committee members, Dr. John E. Kobza, Dr. Mingzhou Jin and Dr. Bogdan Bichescu. Thank you for generously offered your time and provided your insights on this work. Your inputs in the dissertation are highly appreciated.

My sincere thanks also goes to Dr. Jia Shu, who has guided me research since I was an undergraduate student. I would not start this rewarding journey without his encouragement and support. I also would like to thank Dr. Miao Song and Dr. Yongzhen Li for their constructive suggestions and inputs on this work.

I also would like to thank my fellow graduate students: Cong Guo, Zhaoxia Zhao, Shengchao Zhou, Shima Mohebbi, Mohit Shukla, Phillip Scruggs and Mohammad Ramshani. Thank them for sharing the graduate school life together here and the supportive team environment.

Abstract

This study investigates facility location problems in the presence of facility disruptions. Two types of problems are investigated. Firstly, we study a facility location problem considering random disruptions. Secondly, we study a facility fortification problem considering disruptions caused by random failures and intelligent attacks.

We first study a reliable facility location problem in which facilities are faced with the risk of random disruptions. In the literature, reliable facility location models and solution methods have been proposed under different assumptions of the disruption distribution. In most of these models, the disruption distribution is assumed to be completely known, that is, the disruptions are known to be uncorrelated or to follow a certain distribution. In practice, we may have only limited information about the distribution. In this work, we propose a robust reliable facility location model that considers the worst-case distribution with incomplete information. Because the model imposes fewer distributional assumptions, it includes several important reliable facility location problems as special cases. We propose an effective cutting plane algorithm based on the supermodularity of the problem. For the case in which the distribution is completely known, we develop a heuristic algorithm called multi-start tabu search to solve very large instances.

In the second part of the work, we study an r-interdiction median problem with fortification that simultaneously considers two types of disruption risks: random disruptions that happen probabilistically and disruptions caused by intentional attacks. The problem is to determine the allocation of limited facility fortification resources to an existing network. The problem is modeled as a bi-level programming model that generalizes the r-interdiction median problem with probabilistic fortification. The lower level problem, that is, the interdiction problem, is a challenging high-degree non-linear model. In the literature, only the enumeration method is applied to solve a special case of the problem. By exploring the special structure property of the problem, we propose an exact cutting plane method for the problem. For the fortification problem, an effective logic based Benders decomposition algorithm is proposed.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Due to globalization and the widely applied philosophy of lean production, today's supply chains are more vulnerable to disruptions; severe consequences occur even if only a few critical components fail. Facilities are one of the most critical components in supply chain networks. Facility failure leads to massive negative effects, such as a substantial increase in both service costs and customer dissatisfaction. These failures can be caused by natural disasters, such as hurricanes and earthquakes, and intentional or unintentional human actions, such as labor strikes, fires, malicious cyber-attacks and terrorist strikes.

In response to facility disruption risks, both proactive and reactive mitigation options can be used to improve supply chain reliability. An ideal network design should implement and integrate all of these options to make the network risk resilient and cost-effective. One proactive option is to add redundant facilities in the network design phase such that the system can still performance well even when some facilities are disrupted. Another option is to harden or fortify facilities such that they have a smaller chance of being disrupted. The risk of facility disruption decreases by introducing built-in redundancy, investing to enhance

facility infrastructure, and assuring rapid recovery from disruption. In this dissertation, we focus on optimizing the strategic decision of locating and fortifying facilities, as the effects of disruption can be significantly reduced with optimized decisions at the strategic level.

1.1 Facility Location with Disruption

When designing a supply chain network, one of the most crucial decisions is facility location. The is a problem often faced by both private firms and the public sectors. For example, a courier services company must determine where to locate the local stores and sorting facilities, and city governments must determine locations of schools, hospitals and fire stations. These decisions directly affect the performance of the networks. Operations research models have been developed to aid in these decisions.

The simple facility location problem, or the uncapacitated facility location problem (UFLP), is one of the most well-studied location problems. It can be stated as follows: given a set of potential facility locations and a set of customers with known demand rates, one needs to select a subset of these locations at which to set up facilities and determine customer assignments to minimize total costs. The cost components consist of initial facility setup costs and day-to-day transportation costs. Figure 1.1 shows a location solution for a UFLP instance with the 50 largest cities in the U.S. based on 1990 census data. In the UFLP, the facilities are assumed to be constantly available once they are constructed. However, in reality, facilities may become unavailable due to disruptive events, such as natural disasters, terrorist attacks and labor strikes. Although facility failures rarely happen, they can disrupt the normal operations and impose with high costs. For example, in March of 2000, the fire at

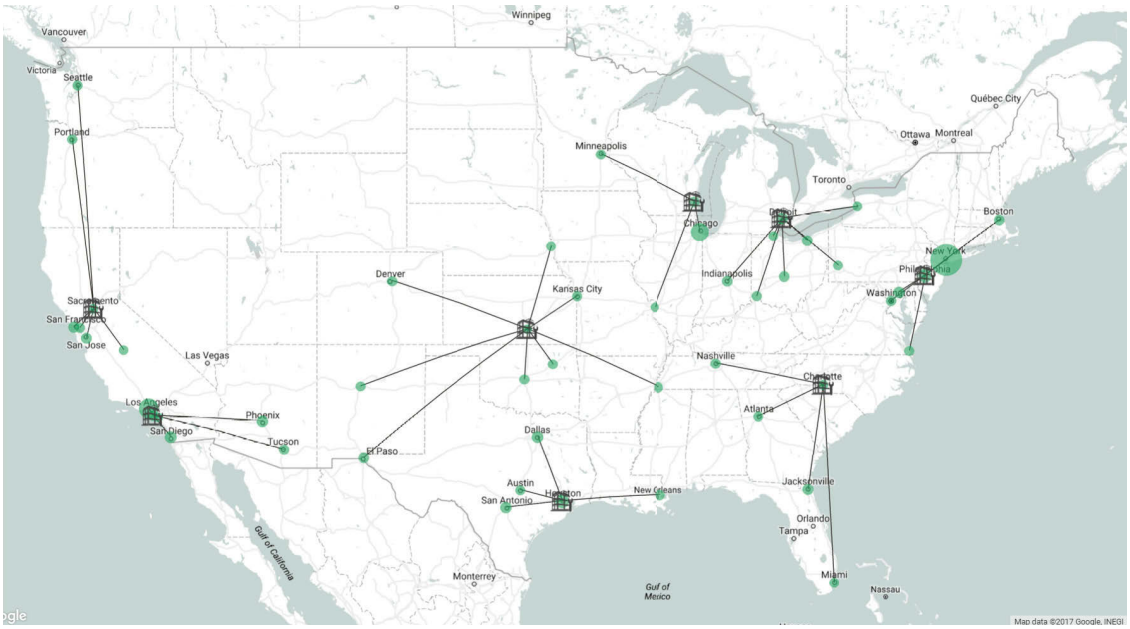

Figure 1.1: An example of the solution to a facility location problem

the Philips microchip factory in New Mexico, U.S. onsignificantly affected its two customers, Nokia and Ericsson. The loss of short term revenue is estimated to be at least $400 million for Ericsson, and the long-term loss was even greater [48]. Other examples, like Hurricane Katrina in 2005 seriously disrupted national oil and gas production in the Gulf of Mexico, amounting to nearly 1.4 million barrels lost per day [14], and the Tohoku-Kanto Earthquake and Tsunami in 2001 caused an estimated $195-305 billion in losses through physical damage alone [64]. Additional examples can be found in the supply chains literature [16, 88, 78]. The risk of disruption is not a new concept in supply chain management, however, interest from practitioners and researchers has increased explosively in recent decades. Snyder et al. [81] give four reasons for this. First, several high-profile events, such as the ones listed above, attracted considerable public attention. Second, the popularity of the just-in-time philosophy results in more vulnerable supply chains. Third, because of globalization, companies tend

to spread their supply chains throughout the world, which increases uncertainty. Lastly, the topic is reaching critical momentum with the maturation of relevant research.

The need to design supply chain networks that effectively balance efficiency and robustness requirements motivates this stream of research. We study a reliable version of UFLP by considering random facility disruptions. In this problem, facility failures have direct effects on transportation costs. When one facility fails, it loses its entire capacity and the customers originally assigned to it may have to be served by other working facilities that are far away, which results in a significant increase in transportation costs. The goal is to design a reliable and resilient distribution network that operates efficiently in both normal and failure scenarios. The objective is to minimize setup costs and expected transportation costs. This problem is referred to as the reliable facility location problem (RFLP). Figure 1.2 shows the solution of the RFLP assuming the disruption probability of a location is proportional to its distance from New Orleans. Compared with the location solution given by figure 1.1, locations near New Orleans are no longer chosen for set-up, and facilities that are far away with lower disruption probabilities are open.

The RFLP is clearly NP-hard because it generalizes the UFLP. Several models have been developed in the last decade and both exact and approximation algorithms have been proposed. However, most RFLP models and algorithms can only deal the problem under certain assumptions about the nature of the facility failures. For example,

1. The RFLP with the most restrictive assumption that all sites fail with equal probability and that facility failures are uncorrelated is studied by Snyder and Daskin [82] and Shen et al. [80].

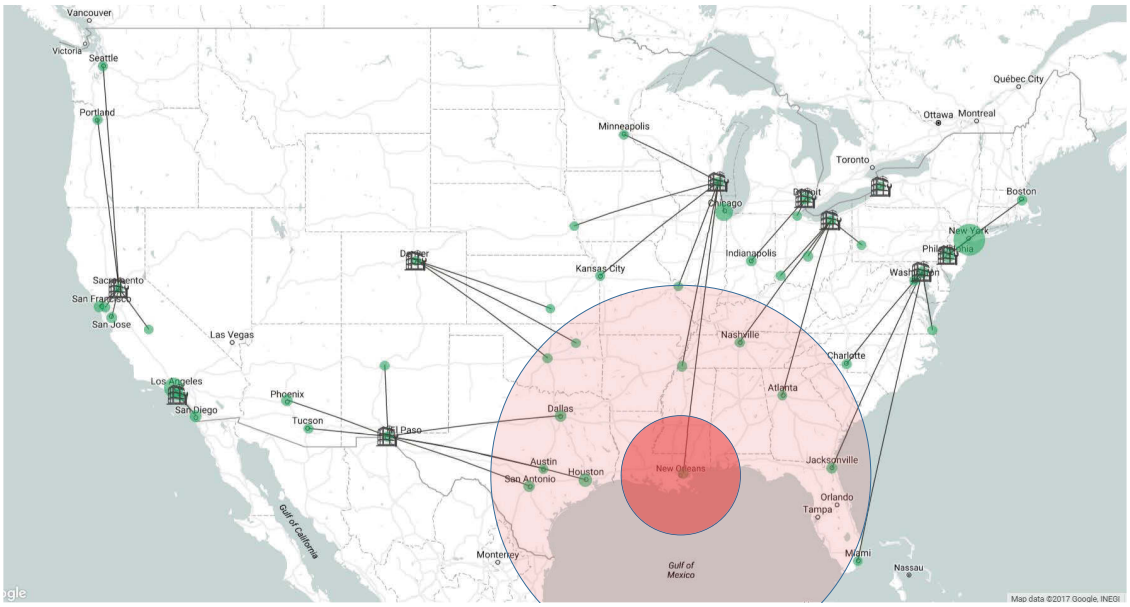

Figure 1.2: An example of the location solution considering disruption risks

2. The problem with heterogeneous failure rates and uncorrelated failures is studied by Cui et al. [24], Shen et al. [80] and Aboolian et al. [1].

3. The problem with heterogeneous failure rates and correlated failures is rarely studied in the literature. Li and Ouyang [51] address this problem with a continuum approximation approach and no exact mathematical programming formulation is provided.

It is still challenging to exactly solve the problem with large scale instances due to its complex nature. Therefore, there is a need to develop more efficient algorithms for the general problem. Moreover, in the works mentioned above, the distribution of the disruptions is assumed to be fully known by the decision maker. However, it is often challenging to obtain full information on the disruptions distribution. In this work, we propose a robust RFLP model that can consider different levels of information visibility. 1. The disruption distribution is completely known; for example, the disruptions are known

to be uncorrelated or to follow a certain distribution. 2. Only partial information about the disruption distribution is known. The proposed model minimizes the setup cost and expected transportation costs under the worst-case distribution. We propose an effective cutting plane algorithm that exactly solves for the problem. As a special case, the cutting plane algorithm solves the RFLP in its general form, that is, with heterogeneous failure rates and correlated failures. We also provide a heuristic algorithm that can solve large instances of RFLP within a reasonable amount of time.

1.2 r-interdiction Median Problem with Fortification

In supply chain networks, identifying and then fortifying most critical nodes is an effective way to hedge against disruption risks in the system. The network fortification problem has gained increasing attention from researchers in the past decades. However, most research considers only one type of disruption risk, such as probabilistic risk that models natural disasters, as in [82, 24, 1], or worst-case risk that models man-made attacks, as in [21, 76, 96]. In reality, disruption risks from different sources exist simultaneously. Therefore, we argue that it is beneficial to have a general model that is able to consider different disruption risks simultaneously.

The model we present in this work considers a generalized r-interdiction median problem with fortification (RIMF). The RIMF is first introduced by Church and Scaparra [21]. In the RIMF, a network of p operating facilities is given. Facilities are assumed to have unlimited capability such that customers are always served by the nearest facility. There exists a malicious attacker who seeks the most critical r facilities to attack such that the effect of the

attack on the network's performance is maximized. The damage is measured by the increase in total weighted distance between customers and their nearest operating facility after the attacks. RIMF solves the problem of allocating protective resources to the most critical q facility in anticipation of the worst-case loss when r facilities are attacked. In the original RIMF, it is assumed that an attack on a protected facility has no effect and an attack on an unprotected facility is successful with certainty.

In reality, a fortified facility may still be disrupted by an attack. Therefore, Zhu et al. [96] propose the r-interdiction median problem with probabilistic fortification (RIMF-p) in which any facility, even a protected one, may be disrupted by a successful attack with some probability. The RIMF-p adds uncertainty in the form of both fortification and interdiction to the RIMF. We study an extension of the RIMF-p with simultaneous risk of probabilistic disruption and intentional attacks. The proposed model includes the RIMF-p as a special case, and the RIMF-p includes the original RIMF and the r-interdiction median (RIM) problem [19] as special cases. We present a bi-level nonlinear mixed-integer programming model for the problem. Due to its high degree of nonlinearity, the model is difficult to solve using standard approaches for bi-level programming models. Even only the lower problem, that is, the attacker's problem, is considered, there is no efficient solution method proposed in the literature. In this dissertation, we present an efficient cutting plane method that exactly solves the attacker's problem. For the upper level problem, that is, the network defenders' problem, we propose solution methods based on the logic-based Benders decomposition framework.

1.3 Document Organization

The remainder of this dissertation is organized as follows. Chapter 2 reviews the literatures on facility location and fortification problems considering disruptions. In Chapter 3, we study the facility location problem with random disruptions. We present the robust reliable facility location model for the problem. By proving the supermodularity of the problem, we propose a cutting plane algorithm for the problem. In addition, a multi-start tabu search algorithm is proposed for solving large instances. The efficiency of the proposed algorithms is demonstrated with extensive computational studies. In Chapter 4, we study the RIMF considering both random disruptions and intentional attacks. We present a bi-level model for the problem. Algorithms are proposed for solving attacker's problem and defender's problem. Computational studies are performed to test the algorithms' performance. Finally, Chapter 5 summaries the contributions of this work and pointing out the future research directions.

Chapter 2

Literature Review

In this chapter, we provide an overview of some basic facility location problems that are relevant to this work in Section 2.1. In Section 2.2, we discuss works related to the facility location problems considering random disruptions. In Section 2.3, we review the related literature on location problems with fortification.

2.1 Facility Location Problems

Facility location, as a critical strategic decision, has been an important research topic in the operations research community for a long time. The systematic study of this problem can be dated to 1909, when Alfred Weber began to study the problem now known as the Weber problem. This problem aims to locate a warehouse such that the total distance to customers are minimized. A vast literature has developed since then. In this section, we review only some of the core location models.

Depending on the decision space, these location problems can be classified into two major categories: continuous location problems, in which a facility can be located at any feasible point in the plane, and discrete location problems, in which facilities are chosen from a set of given candidate locations. The Weber problem and its extensions, such as the multisource Weber problem [59, 25] in which $p > 1$ facilities in the plane has to be determined, belong to the first category.

The most well-known discrete location problems are perhaps the p-median problem and p-center problem introduced by Hakimi [39, 40] in the 1960s. In the p-median problem, one determines the location of p facilities to minimize the weighted average distance between facilities and customers, whereas the p-center problem seeks to minimize the maximal distance. The p-median problem and p-center problem are originally introduced as network location problems in which customers are treated as nodes in a network, and facilities must be placed on the network. However, most of the literature studies these problems from a discrete location perspective because of the following properties: for the p-median problem, Hakimi [40] shows that an optimum can be found by locating facilities among the nodes of the network, and for the p-center problem, Minieka [63] proves that an optimal solution can be found by locating facilities among nodes and a finite number of intersection points on the edge. These two classical problems have received tremendous attention and enormous work has been developed for them and their variants, see survey papers [73, 75].

Motivated by the needs of optimal location in the public sector, such as locations of hospitals and fire stations for which there exist constraints on a maximal distance between a facility and a served customer, Church and Velle [20] introduce the maximal covering location problem. Similar to the p-center problem, the maximal covering location examines

the maximal distance between customers and a facility, but unlike the p-center problem, it includes a fixed covering radius as an input parameter. The goal is to determine facility locations such that the number of customers who can be covered within a given radius is maximized. Farahani et al. [32] summarizes the advances for the covering problem.

The study of UFLP began at a similar time to the p-median and p-center problems in the 1960s, see [56, 28]. As in the p-median problem, the efficiency of the system is measured by the weighted distance between customers and facilities. However, unlike the p-median problem in which the number of open facilities is given as a parameter, in the UFLP, the number of open facilities becomes an endogenous decision and is a trade-off between fixed location costs and transportation costs. The UFLP has been studied under different names in the literature, usually composed of an adjective (uncapaciated, simple, or optimal) and a substantive (plant, warehouse, facility, or site) followed by the word facility [46]. Extensive work has been devoted to the UFLP and both exact and heuristic algorithms have been proposed, (e.g., [23, 29, 18, 5]).

In all of previously mentioned location problems, customers are assigned to an open facility that minimizes assignment cost. However, in the capacitated facility location problem (CFLP), an importation extension of the UFLP, capacity limit is considered for each candidate location. In the CFLP, a customer can be supplied by multiple facilities (e.g., Geoffrion and Bride [35], Sridharan [84]) unless single sourcing is required, (e.g., [89, 3]).

The models mentioned above are the most basic location models in the location science. The complexity of realistic industrial settings gives rise to more sophisticated models, for example, multi-commodity [71, 13, 79], multi-layer [6, 34, 90], and multi-period [43, 66]

models. Interested readers are referred to recent review papers written by [70] and Melo et al. [60].

2.2 Reliable Facility Location Problems

According to Snyder et al. [81], Drezner [26] is the first to consider facility disruptions in facility location models. The author extends the p-median and the p-center problems by considering random facility disruptions. These extended models are called the unreliable p-median problem and the (p, q)-center problem, respectively. In these two models, facilities are assumed to fail with a known probability. In addition, the probability that a customer is served by his k-nearest facility is known, which greatly simplifies the problem. The author proposes neighborhood-search-type heuristic algorithms for both problems.

Snyder and Daskin [82] introduce the RFLP. They propose two reliable facility location models based on p-median and UFLP. Unlike Drezner [26], instead of assuming that the probability that a customer is served by his k-nearest facility is given, Their models calculate these probabilities endogenously. With the assumption that all sites have identical failure probabilities and that disruptions are uncorrelated, the authors propose a Lagrangian relaxation algorithm to solve the problems.

One obvious shortcoming of Snyder and Daskin [82]'s model is the assumption of identical failure probabilities. Researchers have developed models that relax this assumption. One intuitive approach is to explicitly enumerate all or a sample of scenarios. Snyder and Daskin [83] develop a p-robust model with a minimum-expected-cost solution with the constraint that relative regret is no more than $100p\%$ in each scenario. Shen et al. [80] formulate

the RFLP as a two-stage stochastic programming model and apply the sample average approximation. However, the number of scenarios grows exponentially with the number of facilities and the model becomes intractable for even moderate-sized problems.

A more practical approach is to calculate the expectation of transportation costs as a non-linear term within the model, similar to the approach in Snyder and Daskin [82]. However, it is significantly more difficult to derive such a term with heterogeneous failure probabilities. Cui et al. [24] propose a nonlinear mixed integer formulation that allows site-dependent failures and then present an equivalent linear model of the proposed nonlinear model. A Lagrangian based algorithm is developed to solve the linear model. They also provide a continuum approximation model for the problem. Shen et al. [80] propose another nonlinear integer programming model. They use heuristics to find near optimal solutions. They also develop a 4-approximation algorithm with the identical failure probability assumption. Berman et al. [10] study a reliable p-median problem with site-dependent failures and develop exact algorithms and greedy heuristic algorithms. They also derive the worst-case error bound for the greedy algorithms. O'Hanley et al. [68] provide an alternative linearization method for reliable facility location models with site-dependent failure probabilities. They show that their model is highly compact and requires fewer variables and constraints than the model proposed by Cui et al. [24]. Aboolian et al. [1] develop algorithms that contains local search heuristics and a cutting plane procedure for the problem with independent and heterogeneous failures. They first develop a lower bounding model for the RFLP based on the Cui et al. [24]'s model. The lower bound can be improved by successive cutting planes which cut off the solutions that have been explored by local search. They show that their method outperforms the Lagrangian relaxation algorithm proposed by Cui et al. [24] in both

execution time and solution quality. They also develop efficient heuristics based on local search to solve large problem instances.

Location problems with disruptions have also been studied in other contexts. Berman et al. [11] study a location problem in which a customer does not know whether the facility is working or not until he or she visits it (incomplete information). The objective is to minimize the expected customers' travel costs. Greedy heuristic algorithms are proposed. Berman et al. [12] generalize the problem studied by Berman et al. [11] by considering correlations across failure events and derive closed-form analytical results under restricted settings, such as the 2-facility problem on a unit segment. Zhang et al. [95] study the competitive location problem and formulate it as a bilevel optimization problem. They propose a variable neighborhood decomposition search heuristic algorithm to solve the problem. Considering capacity and disruptions together is rare in literature but not unheard of in the literature. Gade and Pohl [33] extend the CFLP with facility disruptions. The authors use a sample average approximation algorithm to solve the problem approximately. A similar problem is studied by Aydin and Murat [8], who propose a swarm intelligence based sample average approximation (SIBSAA).

In these works, excepting Berman et al. [11] and the scenario based models [83, 80], the disruptions are assumed to be uncorrelated. The study of reliable facility location problems under correlated facility disruptions is still rare in the literature. Li and Ouyang [51] study the RFLP in which facilities are subject to spatially correlated disruptions that occur with site-dependent probabilities. They use the continuum approximation approach to estimate and design the complex system. Lu et al. [55] study a reliable facility location problem in which disruptions are can be correlated according to an uncertain joint distribution. Instead

of minimizing the expected costs under all scenarios, they apply distributionally robust optimization techniques to minimize the expected cost under the worst-case distribution. To the best of our knowledge, no solution method in the literature is able to solve the RFLP exactly with general correlated disruptions.

There are related papers on supply chain network design problems in more complex contexts considering disruption risks. Qi et al. [72] study an integrated supply chain design problem on a two-echelon supply chain. The disruptions may happen at both echelons. They describe the facility's disruption-recovery cycles as a stochastic process with memoryless exponential distributions. Chen et al. [15] propose an integer programming model for a reliable version of the joint inventory-location problem. They assume that all open facilities fail independently with an equal probability. Ahmadi-Javid and Seddighi [2] study a reliable location-routing problem with random disruptions at both facilities and vehicles. Zhang et al. [94] study a three-tiered supply chain network design problem considering disruptions, the risk-pooling effect and economies of scale. The authors develop a Lagrangian-relaxation based algorithm and a heuristic algorithm.

2.3 Location Problem with Fortification

In most models reviewed in Section 2.2, the facility disruption occurs as a probabilistic random event. Snyder et al. [81] classify this kind of risk as an exogenous risk that cannot be affected by the decision-maker's action and is modeled using stochastic processes. The objective is to minimize the expected costs under all failure scenarios. These models take the failure risk into consideration at the supply chain design phase and determine optimal

facility location. The facility fortification option is absent in most of these models except those in [54, 50]. Lim et al. [54] study a reliable facility location problem in which both unreliable and reliable (i.e., those that cannot be disrupted) facilities can be set up. They formulate the problem as a mixed integer programming model and develop a Lagrangian relaxation-based solution algorithm. They prove that when all of the failure probabilities of the unreliable facilities are larger or smaller than certain thresholds, the problem reduces to the classic UFLP. In the model, each customer is assigned to at most an unreliable facility and then a reliable facility as a backup. However, this approach might be unrealistic when an unreliable facility fails but a customer can be served by a closer unreliable facility. Li et al. [50] extend the problem of Lim et al. [54] by considering a limited fortification budget.

A closely related stream of research studies location problems where the disruption risk can be mitigated by fortifying facilities. This stream of research originates from the works of Church et al. [19] and Church and Scaparra [21]. Their model considers the disruption risks caused by intelligent intentional attacks. In these models, the disruptions are modeled explicitly by decision variables rather than random events. Snyder et al. [81] classify this kind of risk as endogenous risk. The objective is to fortify an existing network to minimize network costs anticipating worst-case attacks.

Church et al. [19] propose two interdiction models: the r-interdiction median problem (RIM) and the r-interdiction covering problem (RIC). In RIM, the attacker needs to choose r facilities to interdict among the p existing facilities to maximize the increase in the weighted distance. Church and Scaparra [21] extend the previous study of the RIM [19] by adding a level of fortification decision. This problem is referred to as the r-interdiction median problem with fortification (RIMF). In the RIMF, the defender wants to protect the network

by fortifying q facilities to minimize costs in the worst-case scenario when the attacker destroys r unfortified facilities. Scaparra and Church [77] reformulate the model as a maximal covering problem with precedence constraints,in which the model can provide lower and upper bounds to the problem. The bounds are then used to reduce the size of the original model proposed by Church and Scaparra [21]. Scaparra and Church [76] formulate the RIMF as a bilevel integer programming model and propose a solution method based on a tree search implicit enumeration (IE) procedure. Aksen et al. [4] study the fortification problem with a budget constraint, in which the model determines the optimal number of facilities to fortify instead of using a predetermined number. They assume there is a capacity expansion cost when a customer is reassigned to a non-interdicted facility. Liberatore et al. [53] consider the correlation effects between the facilities and that one attack may affect more than one facility. The facilities being affected may only lose some of their capacity.

Zhu et al. [96] introduce probabilistic factors into the fortification model by assuming that an attack is successful only when all defense units allocated in the facility have failed to intercept it. They propose a model that generalizes the RIMF's bilevel formulation with probabilistic factors. However, they do not provide an efficient algorithm to solve the lower level model, and only brute force enumeration is used to find the solution. Zhang et al. [93] consider random attacks which may be introduced by misplaced attacks or natural disasters. Their model requires explicitly enumerating all attack patterns and the number of patterns grows exponentially; as a result, only the most modest instances can be solved.

Chapter 3

Efficient Solution Methods for Facility Location Problems with Random Disruptions

3.1 Problem Definition and Formulation

Consider the problem of locating facilities from a set of potential locations $J = \{1, ..., |J|\}$ to serve a set of customer demand aggregation points $I = \{1, ..., |I|\}$. Each customer $i \in I$ faces a demand with rate d_i. The fixed setup cost to open facility $j \in J$ is f_j and the unit shipment cost from facility $j \in J$ to customer $i \in I$ is c_{ij}. Furthermore, if the demand of customer $i \in I$ is not served, a unit penalty cost c_{i0} is incurred. We assume that $c_{i0} \geq c_{ij}$ for all $i \in I$ and $j \in J$, i.e., any customer should be served as long as there is an available facility.

Facilities are unreliable with unexpected random failures. The random vector $\tilde{\boldsymbol{\xi}} = (\tilde{\xi}_1, ..., \tilde{\xi}_{|J|})^{\mathrm{T}}$ is adopted to represent the failure status of all facilities, where the random variable $\tilde{\xi}_j \in \{0, 1\}$ is 1 if facility j is online and 0 if it is disrupted. The set of all possible realizations of $\tilde{\boldsymbol{\xi}}$ is denoted by

$$\Xi := \left\{(\xi_1, ..., \xi_{|J|})^{\mathrm{T}} \mid \xi_j \in \{0, 1\}\ \forall j \in J\right\} = \{0, 1\}^{|J|}.$$

For any realization $\boldsymbol{\xi} \in \Xi$, let $p_{\boldsymbol{\xi}}$ be the corresponding probability, i.e., $p_{\boldsymbol{\xi}} := \mathrm{Prob}(\tilde{\boldsymbol{\xi}} = \boldsymbol{\xi})$. A distribution of $\tilde{\boldsymbol{\xi}}$ can then be represented by a vector $\boldsymbol{p}$ of $p_{\boldsymbol{\xi}}$ for all $\boldsymbol{\xi} \in \Xi$. Note that $\boldsymbol{p}$ has $2^{|J|}$ components as the cardinality of Ξ is $2^{|J|}$. Due to the high dimensionality, it is often challenging to determine the distribution $\boldsymbol{p}$ of facility failures. Consequently, we assume that the distribution is partially characterized by n pieces of information. For any $k \in \{1, ..., n\}$, the kth piece of information specifies that the probability of all facilities in set A_k being online and all facilities in set B_k being disrupted is within the interval $[\underline{q}_k, \overline{q}_k]$, i.e., $\mathrm{Prob}(\tilde{\xi}_j = 1\,\forall j \in A_k, \tilde{\xi}_j = 0\,\forall j \in B_k) \in [\underline{q}_k, \overline{q}_k]$. Therefore, the distribution $\boldsymbol{p}$ should be contained in the following set P:

$$P := \left\{\boldsymbol{p} \in [0, 1]^{(2^{|J|})} \,\middle|\, \textstyle\sum_{\boldsymbol{\xi} \in \Xi | \xi_j = 1\,\forall j \in A_k, \xi_j = 0\,\forall j \in B_k} p_{\boldsymbol{\xi}} \in [\underline{q}_k, \overline{q}_k]\ \forall k \in \{1, ..., n\},\ \textstyle\sum_{\boldsymbol{\xi} \in \Xi} p_{\boldsymbol{\xi}} = 1\right\}.$$

To the best of our knowledge, the definition of P generalizes the characterization of the disruption distribution in any existing work. Some special cases are discussed as follows.

- Stochastic model. The case with a completely known distribution $\boldsymbol{p}$ can be viewed as P being a singleton defined by $2^{|J|}$ pieces of information. In this case, the problem is

written as:

$$\min_{S \subseteq J} \sum_{j \in S} f_j + \mathbb{E}[Q(S, \tilde{\boldsymbol{\xi}})]. \tag{3.1}$$

A special case of the problem where the complete information is given by the statement that disruptions are independent is study by [82, 80, 24, 1].

- Marginal distribution model. P can be characterized by $|J|$ pieces of information, each of which specifies the marginal probability for a facility to be online. More specifically, we have $\text{Prob}(\tilde{\xi}_j = 1) = q_j$ for any $j \in J$ and hence

$$P = \left\{ \boldsymbol{p} \in [0,1]^{(2^{|J|})} \,\middle|\, \textstyle\sum_{\boldsymbol{\xi} \in \Xi | \xi_j = 1} p_{\boldsymbol{\xi}} = q_j \ \forall j \in J, \ \sum_{\boldsymbol{\xi} \in \Xi} p_{\boldsymbol{\xi}} = 1 \right\}.$$

Lu et al. [55] consider the same characterization of the disruption probability.

- Moment model. Note that the κth cross moment of the random variables $\tilde{\xi}_j$ where $j \in \{j_1, ..., j_\kappa\} \subseteq J$ is $\mathbb{E}_{\boldsymbol{p}}[\prod_{j=j_1}^{j_\kappa} \tilde{\xi}_j] = \text{Prob}(\tilde{\xi}_j = 1 \forall j \in \{j_1, ..., j_\kappa\})$. Thus, the set P can be used to represent the set of distribution specified by the moments of $\tilde{\boldsymbol{\xi}}$. In particular, suppose that the marginal moment of $\tilde{\xi}_j$ is q_j for any $j \in J$, while the cross moment of $\tilde{\xi}_{j_1}$ and $\tilde{\xi}_{j_2}$ for any $j_1, j_2 \in J$ and $j_1 < j_2$ is $q_{j_1 j_2}$. Then the set P specified by the first two moments can be written as

$$P = \left\{ \boldsymbol{p} \in [0,1]^{(2^{|J|})} \,\middle|\, \begin{array}{ll} \sum_{\boldsymbol{\xi} \in \Xi | \xi_j = 1} p_{\boldsymbol{\xi}} = q_j & \forall j \in J, \\ \sum_{\boldsymbol{\xi} \in \Xi | \xi_{j_1} = \xi_{j_2} = 1} p_{\boldsymbol{\xi}} = q_{j_1 j_2} & \forall j_1, j_2 \in J, j_1 < j_2 \\ \sum_{\boldsymbol{\xi} \in \Xi} p_{\boldsymbol{\xi}} = 1 & \end{array} \right\}. \tag{3.2}$$

Based on these available information, we would choose a set $S \subseteq J$ of facilities to be set up to serve the customers. As all the facilities are uncapacitated, any customer should be served by the closest open facility that is not disrupted. Therefore, given set S of the open facilities and realization $\boldsymbol{\xi}$ of the disruption status, the transportation and penalty cost to serve all customers is

$$Q(S, \boldsymbol{\xi}) = \sum_{i \in I} d_i \min_{j \in \{j \in S | \xi_j = 1\} \cup \{0\}} c_{ij},$$

where the shortage penalty cost c_{i0} satisfying $c_{i0} \geq c_{ij}$ for all $i \in I$ and $j \in J$ is incurred only when all the open facilities are disrupted. Recall that the disruption status $\tilde{\boldsymbol{\xi}}$ is uncertain. If its distribution is known, the expectation of $Q(S, \tilde{\boldsymbol{\xi}})$ should be taken into account when deciding S. However, the distribution of $\tilde{\boldsymbol{\xi}}$ can only be characterized by the set P. Because this is a strategic decision, a decision maker may want to consider the worst case expectation of $Q(S, \tilde{\boldsymbol{\xi}})$ among all distributions in the set P, i.e., $\max_{\boldsymbol{p} \in P} \mathbb{E}_{\boldsymbol{p}}[Q(S, \tilde{\boldsymbol{\xi}})]$. Also note that a fixed cost f_j is charged to set up facility $j \in J$. As a result, an ideal set S of open facilities should minimize both the fixed setup cost and the worst-case expected transportation and penalty cost, which leads to the following robust optimization problem:

$$\mathcal{P}: \quad \min_{S \subseteq J} \sum_{j \in S} f_j + \max_{\boldsymbol{p} \in P} \mathbb{E}_{\boldsymbol{p}}[Q(S, \tilde{\boldsymbol{\xi}})]. \tag{3.3}$$

Obviously, model (3.3) is a NP-hard problem, as it generalizes the UFLP, which is a well-known NP-hard problem. In order to design efficient algorithms for this problem, we next study some properties of model (3.3) in the following subsection.

3.1.1 Model Properties

Supermodularity in discrete optimization is similar to the convexity. Intuitively, it demonstrates "increasing returns" and has many applications. It is well-known that the UFLP is equivalent to minimizing a supermodular function [9]. We next show that this property can be extended to the expected transportation and penalty cost item $\mathbb{E}_{\boldsymbol{p}}[Q(S,\tilde{\boldsymbol{\xi}})]$, which is written as $E(S,\boldsymbol{p})$ for brevity, i.e., $E(S,\boldsymbol{p}) := \mathbb{E}_{\boldsymbol{p}}[Q(S,\tilde{\boldsymbol{\xi}})]$.

Firstly, we recall some well-known results of supermodular set functions [see 65, Chap.III.3.1]. Let U be a finite set and f a real-valued function on the subsets of set U. Denote $\rho_e(S) := f(S \cup \{e\}) - f(S)$ as the incremental value of adding an element e to the set S. We have the following definition and properties of supermodular set functions.

Definition 3.1. *A set function f is supermodular if one of the following statements is satisfied,*

(1) $f(S) + f(T) \leq f(S \cap T) + f(S \cup T)$, *for all* $S, T \subseteq U$;

(2) $\rho_e(S) \leq \rho_e(T)$, *for all* $S \subseteq T \subseteq U$ *and* $e \in U \setminus T$.

Lemma 3.2. *A positive linear combination of supermodular functions is supermodular.*

A set function f is defined as nonincreasing if $f(S) \geq f(T)$, for all $S \subseteq T$. We have the following property for nonincreasing supermodular set functions.

Lemma 3.3. *If f is a nonincreasing supermodular set function, then $f(S) \geq f(T) + \sum_{e \in S \setminus T} \rho_e(T)$, for all $S, T \subseteq U$.*

Based on the definition and lemmas above, we can prove the supermodularity and monotonicity of $E(S,\boldsymbol{p})$.

Proposition 3.4. *Given $\boldsymbol{p}$, $E(S, \boldsymbol{p})$ is supermodular and nonincreasing in S.*

Proof: We firstly prove that given a fixed $\boldsymbol{\xi}$, $Q(S, \boldsymbol{\xi})$ is supermodular and nonincreasing in S. For each $S \subseteq T \subseteq J$ and $e \in J \setminus T$,

$$Q(S \cup \{e\}, \boldsymbol{\xi}) - Q(S, \boldsymbol{\xi}) = \sum_{i \in I} d_i \min_{j \in \{j \in S \cup \{e\} | \xi_j = 1\} \cup \{0\}} c_{ij} - \sum_{i \in I} d_i \min_{j \in \{j \in S | \xi_j = 1\} \cup \{0\}} c_{ij}.$$

If $\xi_e = 0$,

$$Q(S \cup \{e\}, \boldsymbol{\xi}) - Q(S, \boldsymbol{\xi}) = Q(T \cup \{e\}, \boldsymbol{\xi}) - Q(T, \boldsymbol{\xi}) = 0,$$

otherwise, i.e., $\xi_e = 1$,

$$\begin{aligned} Q(S \cup \{e\}, \boldsymbol{\xi}) - Q(S, \boldsymbol{\xi}) &= \textstyle\sum_{i \in I} d_i \min\{0, c_{ie} - \min_{j \in \{j \in S | \xi_j = 1\} \cup \{0\}} c_{ij}\} \\ &\leq \textstyle\sum_{i \in I} d_i \min\{0, c_{ie} - \min_{j \in \{j \in T | \xi_j = 1\} \cup \{0\}} c_{ij}\} \\ &= Q(T \cup \{e\}, \boldsymbol{\xi}) - Q(T, \boldsymbol{\xi}). \end{aligned}$$

Therefore $Q(S, \boldsymbol{\xi})$ is supermodular in S according to Definition 3.1. Furthermore, we have $Q(S \cup \{e\}, \boldsymbol{\xi}) \leq Q(S, \boldsymbol{\xi})$, and thus $Q(S, \boldsymbol{\xi})$ with fixed $\boldsymbol{\xi}$ is nonincreasing and supermodular in S.

We next consider $E(S, \boldsymbol{p})$, i.e., the expected transportation and penalty cost. As the entries of $\boldsymbol{p}$ is finite, $E(S, \boldsymbol{p})$ can be written as $E(S, \boldsymbol{p}) = \sum_{\boldsymbol{\xi} \in \Xi} Q(S, \boldsymbol{\xi}) p_{\boldsymbol{\xi}}$. Note that $p_{\boldsymbol{\xi}}$ is nonnegative. Thus $E(S, \boldsymbol{p})$ is supermodular and nonincreasing in S because of Lemma 3.2 and the fact that, a positive linear combination of nonincreasing functions is nonincreasing. This completes the proof.

Note that $Q(S, \boldsymbol{\xi})$ in our problem is essentially the same as $h(\boldsymbol{x}, \boldsymbol{\xi})$ in [55], in which $\boldsymbol{x}$ denotes the vector of facility location decisions and $\boldsymbol{\xi}$ denotes the disruption scenario. They proved that $h(\boldsymbol{x}, \boldsymbol{\xi})$ is supermodular in $S(\boldsymbol{\xi}) = \{j \in J | \xi_j = 1\}$, which is the set of online facilities depending on the disruption scenario $\boldsymbol{\xi}$. Different from their work, Proposition 3.4 proves that $Q(S, \boldsymbol{\xi})$ is supermodular in S, which is the decision, i.e., the set of facilities decided to open.

We further notice that Proposition 3.4 does not rely on the independence of disruptions. Therefore, the solution methods developed in next section, which are based on Proposition 3.4, can handle the correlated disruptions as well.

3.2 Solution Methods

In this section, we first derive the cutting plane algorithm framework for solving robust model (3.3). Since the stochastic model (3.1) is a special case of the robust model (3.3), the algorithm is readily applicable to solve model (3.1). Then, we propose a multi-start tabu search algorithm for solving model (3.1).

3.2.1 An Exact Cutting Plane Algorithm

By introducing a continuous variable η to represent the worst-case expected transportation and penalty cost, we can write model (3.3) as,

$$\begin{aligned} \mathcal{P}: \quad & \min_{S \subseteq J} \quad \textstyle\sum_{j \in S} f_j + \eta \\ & \text{s.t.} \quad \eta \geq E(S, \boldsymbol{p}), \quad \forall \boldsymbol{p} \in P. \end{aligned} \tag{3.4}$$

Given $\boldsymbol{p} \in P$, for the nonincreasing and supermodular function $E(S, \boldsymbol{p})$ in S, we define $\rho_j(S, \boldsymbol{p}) := E(S \cup \{j\}, \boldsymbol{p}) - E(S, \boldsymbol{p})$, representing the incremental value when involving node j into S, similarly as in Definition 3.1. According to Contreras and Fernández [22], we have the following theorem, which can be derived from Lemma 3.3.

Theorem 3.5. *Given $S \subseteq J$ and $\boldsymbol{p} \in P$, consider a real number $\eta \in \mathbb{R}$ and a set $K := \{\eta \in \mathbb{R} | \eta \geq E(T, \boldsymbol{p}) + \sum_{j \in S \setminus T} \rho_j(T, \boldsymbol{p}), \forall T \subseteq J\}$. Then $\eta \in K$ if and only if $\eta \geq E(S, \boldsymbol{p})$.*

Based on Theorem 3.5, model (3.3) can be written as,

$$\mathcal{P}: \quad \min_{S \subseteq J} \quad \sum_{j \in S} f_j + \eta \tag{3.5a}$$

$$\text{s.t.} \quad \eta \geq E(T, \boldsymbol{p}) + \sum_{j \in S \setminus T} \rho_j(T, \boldsymbol{p}), \quad \forall T \subseteq J, \boldsymbol{p} \in P. \tag{3.5b}$$

Model (3.5) contains an exponential number of constraints, as constraints (3.5b) should be satisfied for all the subsets of J and $\boldsymbol{p}$ in P. It is impractical to directly solve even a problem with moderate size. For example, if $|J| = 100$, the quantity of all the subsets of J is $2^{100} \approx 1.2676506 \times 10^{30}$.

Fortunately, the cutting plane approach can be applied to efficiently solve this problem. Generally speaking, to implement the cutting plane approach, we start from solving the relaxed problem with only a subset of constraints. Then, we identify the violated constraints by solving the separation problem, and solve the relaxed problem again with these violated constraints involved. These steps repeat and continue, until no violated constraint is found.

Given S, the separation problem of constraints (3.5b), denoted as $Z_{sep}(S)$, is obtained by maximizing the right-hand side of constraints (3.5b) over all $T \subseteq J$ and $\boldsymbol{p} \in P$, or

equivalently, by maximizing the right-hand side of the constraints in model (3.4) over all $\boldsymbol{p} \in P$, i.e.,

$$\begin{aligned} Z_{sep}(S) &= \max_{\boldsymbol{p}\in P, T\subseteq J} \left\{ E(T,\boldsymbol{p}) + \textstyle\sum_{j\in S\setminus T} \rho_j(T,\boldsymbol{p}) \right\} \\ &= \max_{\boldsymbol{p}\in P} \max_{T\subseteq J} \left\{ E(T,\boldsymbol{p}) + \textstyle\sum_{j\in S\setminus T} \rho_j(T,\boldsymbol{p}) \right\} \\ &= \max_{\boldsymbol{p}\in P} E(S,\boldsymbol{p}), \end{aligned}$$

where the second line comes from the equivalent transformation, and the third line comes from Lemma 3.3 and Proposition 3.4, or more specifically, as $E(S,\boldsymbol{p})$ is nonincreasing and supermodular in S, we have $E(S,\boldsymbol{p}) \geq E(T,\boldsymbol{p}) + \sum_{j\in S\setminus T} \rho_j(T,\boldsymbol{p})$, for all $T \subseteq J$.

The detailed steps of the cutting plane algorithm is as follows.

Step 1 Consider the relaxation of model (3.5), denoted as $\mathcal{RP}$, with only a subset of constraints (3.5b).

Step 2 Solve $\mathcal{RP}$ and obtain the optimal solution (η^*, S^*).

Step 3 Solve the separation problem $Z_{sep}(S^*) = \max_{\boldsymbol{p}\in P} E(S^*,\boldsymbol{p})$. Obtain the optimal solution $\boldsymbol{p}^*$ and the optimal objective value $Z^*_{sep}(S^*) = E(S^*,\boldsymbol{p}^*)$. If $\eta^* < E(S^*,\boldsymbol{p}^*)$, add constraint $\eta \geq E(S^*,\boldsymbol{p}^*) + \sum_{j\in S\setminus S^*} \rho_j(S^*,\boldsymbol{p}^*)$ into $\mathcal{RP}$, and then go to Step 2; otherwise, i.e., $\eta^* \geq E(S^*,\boldsymbol{p}^*)$, terminate the algorithm and output the current solution (η^*, S^*) as the optimal solution for model (3.5).

Note that in Step 3 above, we claim that $\eta^* \geq E(S^*,\boldsymbol{p}^*)$ indicates the optimality of the current solution (η^*, S^*). This is because

$$\eta^* \geq E(S^*,\boldsymbol{p}^*) = \max_{\boldsymbol{p}\in P} E(S^*,\boldsymbol{p}),$$

where the equality sign comes from the fact that $\boldsymbol{p}^*$ is the optimal solution of the separation problem. Thus we have

$$\eta^* \geq E(S^*, \boldsymbol{p}), \quad \forall \boldsymbol{p} \in P,$$

i.e., all the constraints in model (3.4) are satisfied, or equivalently, no violated constraint can be found for the current solution (η^*, S^*), and therefore (η^*, S^*) is optimal.

For this cutting plane algorithm, we have the following theorem.

Theorem 3.6. *The cutting plane algorithm solves model (3.5) to optimality within finite iterations.*

Proof: An upper bound of the number of iterations of the cutting plane algorithm is the number of J's subsets, which is finite, and thus the algorithm terminates within finite iterations.

Furthermore, as $\mathcal{P}$ is a minimization problem, the objective value of $\mathcal{RP}$ provides a lower bound to model (3.3), and any solution (η^*, S^*) to $\mathcal{RP}$ always satisfies $\eta^* \leq \max_{\boldsymbol{p} \in P} E(S^*, \boldsymbol{p})$. In Step 3, when the algorithm terminates with the solution (η^*, S^*), we have $\eta^* \geq \max_{\boldsymbol{p} \in P} E(S^*, \boldsymbol{p})$. Thus $\eta^* = \max_{\boldsymbol{p} \in P} E(S^*, \boldsymbol{p})$, and an optimal solution is obtained when the algorithm terminates.

Remark: The cutting plane procedure can be implemented using the lazy constraint callback provided by CPLEX. The solver automatically checks if there exists any super-modular constraints that are violated, when an integer solution has been identified. This implementation automatically takes the advantage of a warm start. In our preliminary computational experiment, it speeds up the solution time by about several times when compared to the procedure without lazy constraint callback.

The relaxed problem $\mathcal{RP}$ in Step 2 of the cutting plane algorithm is a mixed integer programming with moderate number of constraints, thus $\mathcal{RP}$ can be solved efficiently by commercial solver like CPLEX. However, because the distribution set P is generally defined, in some cases the separation problem $Z_{sep}(S^*)$ in Step 3 of the cutting plane algorithm could be a linear programming problem with exponential number of decision variables, which is not an easy problem to be solved. In the following subsection 3.2.1, we will discuss the approach to solve this separation problem, and the detailed algorithmic steps to formulate the supermodular cut $\eta \geq E(S^*, \boldsymbol{p}^*) + \sum_{j \in S \setminus S^*} \rho_j(S^*, \boldsymbol{p}^*)$ are presented in subsection 3.2.1.

Solving the Separation Problem

Because the separation problem $Z_{sep}(S)$ contains exponential number of variables, we next apply the column generation approach to solve it efficiently.

With the general definition of P, the separation problem $Z_{sep}(S) = \max_{\boldsymbol{p} \in P} E(S, \boldsymbol{p})$ can be written as,

$$
\begin{aligned}
Z_{sep}(S) = \quad & \max_{p_{\boldsymbol{\xi}} \geq 0} \quad \textstyle\sum_{\boldsymbol{\xi} \in \Xi} Q(S, \boldsymbol{\xi}) p_{\boldsymbol{\xi}} \\
& \text{s.t.} \quad \textstyle\sum_{\boldsymbol{\xi} \in \Xi} p_{\boldsymbol{\xi}} = 1, \\
& \qquad \textstyle\sum_{\boldsymbol{\xi} \in \Xi | \xi_j = 1 \, \forall j \in A_k, \xi_j = 0 \, \forall j \in B_k} p_{\boldsymbol{\xi}} \leq \overline{q}_k, \qquad \forall k = 1, ..., n, \\
& \qquad - \textstyle\sum_{\boldsymbol{\xi} \in \Xi | \xi_j = 1 \, \forall j \in A_k, \xi_j = 0 \, \forall j \in B_k} p_{\boldsymbol{\xi}} \leq -\underline{q}_k, \quad \forall k = 1, ..., n,
\end{aligned}
\tag{3.6}
$$

where the constraints are from the definition of P. Define the dual variables of the constraints in model (3.6) respectively as α, $\overline{\beta}_k$ and $\underline{\beta}_k$ for all $k = 1, ..., n$, and then the dual problem of

model (3.6) is as follows,

$$\begin{aligned} Z^D_{sep}(S) = \min_{\alpha,\overline{\boldsymbol{\beta}},\underline{\boldsymbol{\beta}}} \quad & \alpha + \textstyle\sum_{k=1}^{n}(\overline{q}_k\overline{\beta}_k - \underline{q}_k\underline{\beta}_k) \\ \text{s.t.} \quad & \alpha + \textstyle\sum_{k\in\{1,\ldots,n\}|\xi_j=1\,\forall j\in A_k,\xi_j=0\,\forall j\in B_k}(\overline{\beta}_k - \underline{\beta}_k) \geq Q(S,\boldsymbol{\xi}), \quad \forall \boldsymbol{\xi}\in\Xi, \\ & \overline{\beta}_k \geq 0, \underline{\beta}_k \geq 0, \quad \forall k = 1,\ldots,n, \end{aligned}$$

where $\overline{\boldsymbol{\beta}}$ and $\underline{\boldsymbol{\beta}}$ are respectively the vectors of $\overline{\beta}_k$ and $\underline{\beta}_k$. The corresponding pricing problem (also known as the reduce cost), denoted as $RC(S)$, is as follows,

$$RC(S) = \max_{\boldsymbol{\xi}\in\Xi}\left\{ Q(S,\boldsymbol{\xi}) - \alpha - \sum_{\substack{k\in\{1,\ldots,n\}|\\ \xi_j=1\,\forall j\in A_k,\xi_j=0\,\forall j\in B_k}} (\overline{\beta}_k - \underline{\beta}_k) \right\}.$$

Substitute $Q(S,\boldsymbol{\xi})$ and conduct equivalent reformulation, then $RC(S)$ can be further written as,

$$RC(S) = \max_{\boldsymbol{\xi}\in\Xi}\left\{ \sum_{i\in I} d_i \min_{j\in\{j\in S|\xi_j=1\}\cup\{0\}} c_{ij} - \alpha - \sum_{k=1}^{n}(\overline{\beta}_k - \underline{\beta}_k)\prod_{j\in A_k}\xi_j \prod_{j'\in B_k}(1-\xi_{j'}) \right\}. \quad (3.7)$$

For example, given P in (3.2), which is defined by the first two moments of $\tilde{\boldsymbol{\xi}}$, the corresponding pricing problem to solve the separation problem is

$$RC(S) = \max_{\boldsymbol{\xi}\in\Xi}\left\{ \sum_{i\in I} d_i \min_{j\in\{j\in S|\xi_j=1\}\cup\{0\}} c_{ij} - \alpha - \sum_{j\in J}\beta_j\xi_j - \sum_{j_1,j_2\in J, j_1<j_2}\beta_{j_1j_2}\xi_{j_1}\xi_{j_2} \right\}. \quad (3.8)$$

Here, β_j and $\beta_{j_1j_2}$ are the dual variables corresponding to the first and second moments constraints, i.e., the first two constraints in (3.2), respectively.

For the pricing problem $RC(S)$, we have the following proposition.

Proposition 3.7. *Model* (3.7) *is NP-hard.*

Proof: Consider model (3.8), which is a special case of model (3.7). It is sufficient to show that the weighted MAX CUT problem, a well-known NP-hard problem, can be reduced to model (3.8) in polynomial time.

The weighted MAX CUT problem is formally defined as follows: Given a simple graph $G = (V, E)$ and a weight $w_{ij} \in \mathbb{Z}^+$ for each edge $(i, j) \in E$, partition V into disjoint sets V_1 and V_2 such that the sum of the weights of the edges between V_1 and V_2 is maximized. Let $w_{ij} = 0$ for any $(i, j) \notin E$. It can be formulated as

$$\max_{y_i \in \{-1,1\}} \left\{ \frac{1}{4} \sum_{i,j \in V, i \neq j} w_{ij}(1 - y_i y_j) \right\},$$

where $y_i = 1$ if $i \in V_1$ and $y_i = -1$ if $i \in V_2$.

Consider an instance of model (3.8) where $c_{ij} = c_{i0}$ for all $i \in I$ and $j \in J$. Set $\beta_{j_2 j_1} := \beta_{j_1 j_2}$ for all $j_1, j_2 \in J$ such that $j_1 < j_2$. Model (3.8) can be written as

$$RC(S) = \max_{\xi_j \in \{0,1\}} \left\{ \sum_{i \in I} d_i c_{i0} - \alpha - \sum_{j \in J} \beta_j \xi_j - \frac{1}{2} \sum_{j_1, j_2 \in J, j_1 \neq j_2} \beta_{j_1 j_2} \xi_{j_1} \xi_{j_2} \right\}.$$

Define a new decision variable $x_j := 2\xi_j - 1$, i.e., $\xi_j = (x_j + 1)/2$ for any $j \in J$. We have $x_j \in \{-1, 1\}$ for all $j \in J$ and

$$\begin{aligned} RC(S) &= \max_{x_j \in \{-1,1\}} \left\{ \sum_{i \in I} d_i c_{i0} - \alpha - \sum_{j \in J} \beta_j \cdot \frac{x_j + 1}{2} - \frac{1}{2} \sum_{j_1, j_2 \in J, j_1 \neq j_2} \beta_{j_1 j_2} \cdot \frac{x_{j_1} + 1}{2} \cdot \frac{x_{j_2} + 1}{2} \right\} \\ &= \max_{x_j \in \{-1,1\}} \left\{ \left(\sum_{i \in I} d_i c_{i0} - \alpha - \frac{1}{2} \sum_{j \in J} \beta_j - \frac{1}{8} \sum_{j_1, j_2 \in J, j_1 \neq j_2} \beta_{j_1 j_2} \right) \right. \\ &\qquad \left. - \sum_{j \in J} \left(\frac{1}{2} \beta_j + \frac{1}{4} \sum_{j' \in J, j' \neq j} \beta_{jj'} \right) \cdot x_j - \sum_{j_1, j_2 \in J, j_1 \neq j_2} \frac{1}{8} \beta_{j_1 j_2} x_{j_1} x_{j_2} \right\}. \end{aligned}$$

Suppose that $J = V$ and

$$\begin{cases} \dfrac{1}{8} \beta_{j_1 j_2} = \dfrac{1}{4} w_{j_1 j_2}, & \forall j_1, j_2 \in J, j_1 \neq j_2, \\ \dfrac{1}{2} \beta_j + \dfrac{1}{4} \displaystyle\sum_{j' \in J, j' \neq j} \beta_{jj'} = 0, & \forall j \in J, \\ \displaystyle\sum_{i \in I} d_i c_{i0} - \alpha - \frac{1}{2} \sum_{j \in J} \beta_j - \frac{1}{8} \sum_{j_1, j_2 \in J, j_1 \neq j_2} \beta_{j_1 j_2} = \frac{1}{4} \sum_{j_1, j_2 \in J, j_1 \neq j_2} w_{j_1 j_2}, \end{cases}$$

i.e.,

$$\beta_{j_1 j_2} = 2 w_{j_1 j_2} \ \forall j_1, j_2 \in J, j_1 \neq j_2, \quad \beta_j = - \sum_{j' \in J, j' \neq j} w_{jj'} \ \forall j \in J, \quad \alpha = \sum_{i \in I} d_i c_{i0}.$$

It is straightforward that this specific instance of $RC(S)$ is equivalent the weighted MAX CUT problem.

To implement the column generation approach, the pricing problem need to be solved for many times in order to identify effective columns, or decision variables, thus the efficiency of solving the pricing problem has a significant impact on the efficiency of the whole algorithm. Although the pricing problem (3.7) is proved to be NP-hard, indicating that no

polynomial algorithm exists for it, we can instead reformulate it into an equivalent integer programming with moderate number of decision variables and constraints, which can be solved satisfactorily by commercial solvers like CPLEX. The equivalent reformulation is presented in the following proposition.

Proposition 3.8. *Let* $A_k = \{a_k^1, ..., a_k^{|A_k|}\}$ *and* $B_k = \{b_k^1, ..., b_k^{|B_k|}\}$ *for any* $k \in \{1, ..., n\}$. *Then model* (3.7) *is equivalent to*

$$
\begin{aligned}
RC(S) = \max_{\substack{\boldsymbol{\xi}\in\Xi,\\ \boldsymbol{\pi},\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\nu}}} \quad & \sum_{i\in I} d_i\pi_i - \alpha - \sum_{k\in K^{\ominus}} (\overline{\beta}_k - \underline{\beta}_k)\lambda_k + \sum_{k\in K^{\oplus}} (\overline{\beta}_k - \underline{\beta}_k)\left(-1 + \sum_{m=1}^{|A_k|}\mu_k^m + \sum_{m=1}^{|B_k|}\nu_k^m\right) \\
\text{s.t.} \quad & \pi_i \le c_{ij}\xi_j + c_{i0}(1-\xi_j), && \forall i \in I, j \in S, \\
& \lambda_k \le \xi_j, && \forall k \in K^{\ominus}, j \in A_k, \\
& \lambda_k \le 1 - \xi_j, && \forall k \in K^{\ominus}, j \in B_k, \\
& \mu_k^m \le \xi_j, && \forall k \in K^{\oplus}, m \in \{2, ..., |A_k|\}, j \in \{a_k^1, ..., a_k^{m-1}\}, \\
& \mu_k^m \le 1 - \xi_{a_k^m}, && \forall k \in K^{\oplus}, m \in \{1, ..., |A_k|\}, \\
& \nu_k^m \le 1 - \xi_j, && \forall k \in K^{\oplus}, m \in \{2, ..., |B_k|\}, j \in \{b_k^1, ..., b_k^{m-1}\}, \\
& \nu_k^m \le \xi_j, && \forall k \in K^{\oplus}, m \in \{1, ..., |B_k|\}, j \in A_k \cup \{b_k^m\},
\end{aligned}
\tag{3.9}
$$

where $\boldsymbol{\pi}$, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ *and* $\boldsymbol{\nu}$ *are respectively the vectors of* π_i, λ_k, μ_k^m *and* ν_k^m,

$$
K^{\ominus} := \{k \in \{1, ..., n\} \mid \overline{\beta}_k - \underline{\beta}_k < 0\} \qquad \textit{and} \qquad K^{\oplus} := \{k \in \{1, ..., n\} \mid \overline{\beta}_k - \underline{\beta}_k > 0\}.
$$

Proof: Let π_i represent the term $\min_{j\in\{j\in S|\xi_j=1\}\cup\{0\}} c_{ij}$ for any $i \in I$. We can write model (3.7) as

$$\begin{aligned} RC(S) = \max_{\boldsymbol{\xi}\in\Xi,\boldsymbol{\pi}} \quad & \textstyle\sum_{i\in I} d_i\pi_i - \alpha - \sum_{k=1}^{n}(\overline{\beta}_k - \underline{\beta}_k)\prod_{j\in A_k}\xi_j\prod_{j'\in B_k}(1-\xi_{j'}) \\ \text{s.t.} \quad & \pi_i \le c_{ij}\xi_j + c_{i0}(1-\xi_j), \quad \forall i \in I, j \in S. \end{aligned}$$

Consider the term $-(\overline{\beta}_k - \underline{\beta}_k)\prod_{j\in A_k}\xi_j\prod_{j'\in B_k}(1-\xi_{j'})$ for any $k \in \{1,...,n\}$ in the objective function.

Case 1: Suppose that $\overline{\beta}_k - \underline{\beta}_k < 0$, i.e., $k \in K^{\ominus}$. We can introduce a new decision variable λ_k to represent $\prod_{j\in A_k}\xi_j\prod_{j'\in B_k}(1-\xi_{j'})$. Then the term $-(\overline{\beta}_k - \underline{\beta}_k)\prod_{j\in A_k}\xi_j\prod_{j'\in B_k}(1-\xi_{j'})$ can be replaced with $-(\overline{\beta}_k - \underline{\beta}_k)\lambda_k$ by adding the constraints

$$\lambda_k \le \xi_j, \ \forall j \in A_k \qquad \text{and} \qquad \lambda_k \le 1-\xi_j, \ \forall j \in B_k.$$

Case 2: Suppose that $\overline{\beta}_k - \underline{\beta}_k > 0$, i.e., $k \in K^{\oplus}$. Note that

$$-\prod_{j\in A_k}\xi_j = -\prod_{j=a_k^1}^{a_k^{|A_k|-1}}\xi_j\left(1-(1-\xi_{a_k^{|A_k|}})\right) = -\prod_{j=a_k^1}^{a_k^{|A_k|-1}}\xi_j + \prod_{j=a_k^1}^{a_k^{|A_k|-1}}\xi_j(1-\xi_{a_k^{|A_k|}}) = \cdots$$

$$= -\xi_{a_k^1} + \xi_{a_k^1}(1-\xi_{a_k^2}) + \cdots + \prod_{j=a_k^1}^{a_k^{|A_k|-1}}\xi_j(1-\xi_{a_k^{|A_k|}}) = -1 + (1-\xi_{a_k^1}) + \sum_{m=2}^{|A_k|}\prod_{j=a_k^1}^{a_k^{m-1}}\xi_j(1-\xi_{a_k^m})$$

and

$$-\prod_{j'\in B_k}(1-\xi_{j'}) = -\prod_{j'=b_k^1}^{b_k^{|B_k|-1}}(1-\xi_{j'})(1-\xi_{b_k^{|B_k|}}) = -\prod_{j'=b_k^1}^{b_k^{|B_k|-1}}(1-\xi_{j'}) + \prod_{j'=b_k^1}^{b_k^{|B_k|-1}}(1-\xi_{j'})\xi_{b_k^{|B_k|}} = \cdots$$

$$= -1+\xi_{b_k^1}+(1-\xi_{b_k^1})\xi_{b_k^2}+\cdots+\prod_{j'=b_k^1}^{b_k^{|B_k|-1}}(1-\xi_{j'})\xi_{b_k^{|B_k|}} = -1+\xi_{b_k^1}+\sum_{m=2}^{|B_k|}\prod_{j'=b_k^1}^{b_k^{m-1}}(1-\xi_{j'})\xi_{b_k^m},$$

which yields

$$-\prod_{j\in A_k}\xi_j\prod_{j'\in B_k}(1-\xi_{j'}) = -\prod_{j\in A_k}\xi_j+\prod_{j\in A_k}\xi_j\xi_{b_k^1}+\sum_{m=2}^{|B_k|}\prod_{j\in A_k}\xi_j\prod_{j'=b_k^1}^{b_k^{m-1}}(1-\xi_{j'})\xi_{b_k^m}$$

$$= -1+(1-\xi_{a_k^1})+\sum_{m=2}^{|A_k|}\prod_{j=a_k^1}^{a_k^{m-1}}\xi_j(1-\xi_{a_k^m})+\prod_{j\in A_k}\xi_j\xi_{b_k^1}+\sum_{m=2}^{|B_k|}\prod_{j\in A_k}\xi_j\prod_{j'=b_k^1}^{b_k^{m-1}}(1-\xi_{j'})\xi_{b_k^m}.$$

Define the new decision variables μ_k^1, μ_k^m for any $m = 2, ..., |A_k|$, ν_k^1, and ν_k^m for any $m = 2, ..., |B_k|$, which represent the terms

$$1-\xi_{a_k^1}, \qquad \prod_{j=a_k^1}^{a_k^{m-1}}\xi_j(1-\xi_{a_k^m}), \qquad \prod_{j\in A_k}\xi_j\xi_{b_k^1}, \qquad \text{and} \qquad \prod_{j\in A_k}\xi_j\prod_{j'=b_k^1}^{b_k^{m-1}}(1-\xi_{j'})\xi_{b_k^m},$$

respectively. Under the constraints

$$\begin{aligned}
&\mu_k^m \le \xi_j, && \forall m\in\{2,...,|A_k|\}, j\in\{a_k^1,...,a_k^{m-1}\},\\
&\mu_k^m \le 1-\xi_{a_k^m}, && \forall m\in\{1,...,|A_k|\},\\
&\nu_k^m \le 1-\xi_j, && \forall m\in\{2,...,|B_k|\}, j\in\{b_k^1,...,b_k^{m-1}\},\\
&\nu_k^m \le \xi_j, && \forall m\in\{1,...,|B_k|\}, j\in A_k\cup\{b_k^m\},
\end{aligned}$$

we can replace the term $-(\overline{\beta}_k - \underline{\beta}_k) \prod_{j \in A_k} \xi_j \prod_{j' \in B_k} (1 - \xi_{j'})$ in the objective function with

$$(\overline{\beta}_k - \underline{\beta}_k) \left(-1 + \sum_{m=1}^{|A_k|} \mu_k^m + \sum_{m=1}^{|B_k|} \nu_k^m \right).$$

Proposition 3.8 reformulates the pricing problem $RC(S)$ in (3.7) as the linear integer program in (3.9), which is ready to be solved by any commercial IP solver, e.g., CPLEX. Although model (3.9) has more decision variables and constraints than model (3.7), its numbers of variables and constraints are

$$|J| + |I| + |K^{\ominus}| + \sum_{k \in K^{\oplus}} |A_k| + \sum_{k \in K^{\oplus}} |B_k| \leq |I| + |J| + \sum_{k=1}^{n} (|A_k| + |B_k|) \leq |I| + (n+1)|J|$$

and

$$\begin{aligned}
&|I||S| + \sum_{k \in K^{\ominus}} |A_k| + \sum_{k \in K^{\ominus}} |B_k| + \sum_{k \in K^{\oplus}} \sum_{m=2}^{|A_k|} (m-1) + \sum_{k \in K^{\oplus}} |A_k| + \sum_{k \in K^{\oplus}} \sum_{m=2}^{|B_k|} (m-1) \\
&\quad + \sum_{k \in K^{\oplus}} \sum_{m=1}^{|B_k|} (|A_k| + 1) \\
&\leq |I||J| + \sum_{k=1}^{n} \sum_{m=2}^{|A_k|} (m-1) + \sum_{k=1}^{n} |A_k| + \sum_{k=1}^{n} \sum_{m=2}^{|B_k|} (m-1) + \sum_{k=1}^{n} \sum_{m=1}^{|B_k|} (|A_k| + 1) \\
&= |I||J| + \sum_{k=1}^{n} \frac{1}{2} |A_k|(|A_k| - 1) + \sum_{k=1}^{n} |A_k| + \sum_{k=1}^{n} \frac{1}{2} |B_k|(|B_k| - 1) + \sum_{k=1}^{n} (|A_k| + 1)|B_k| \\
&= |I||J| + \sum_{k=1}^{n} \frac{1}{2} \Big(|A_k|^2 - |A_k| + 2|A_k| + |B_k|^2 - |B_k| + 2|A_k||B_k| + 2|B_k| \Big) \\
&= |I||J| + \sum_{k=1}^{n} \frac{1}{2} \Big(|A_k|^2 + 2|A_k||B_k| + |B_k|^2 + |A_k| + |B_k| \Big) \\
&= |I||J| + \sum_{k=1}^{n} \frac{1}{2} (|A_k| + |B_k| + 1)(|A_k| + |B_k|) \leq |I||J| + \frac{n}{2} |J|(|J| + 1)
\end{aligned}$$

respectively, and hence are still polynomial in the input size, i.e., $|I|$, $|J|$ and n.

As an illustration of Proposition 3.8, we consider P in (3.2), which is defined by the first two moments of $\tilde{\boldsymbol{\xi}}$. The corresponding $RC(S)$ in (3.8) can be reformulated as

$$\begin{aligned}
RC(S) = \max_{\substack{\boldsymbol{\xi}\in\Xi,\\ \boldsymbol{\pi},\boldsymbol{\lambda},\boldsymbol{\mu}}} \; & \sum_{i\in I} d_i\pi_i - \alpha - \sum_{j\in J}\beta_j\xi_j - \sum_{\substack{j_1,j_2\in J, j_1<j_2,\\ \beta_{j_1j_2}<0}} \beta_{j_1j_2}\lambda_{j_1j_2} + \sum_{\substack{j_1,j_2\in J, j_1<j_2,\\ \beta_{j_1j_2}>0}} \beta_{j_1j_2}\left(-\xi_{j_1}+\mu_{j_1j_2}\right) \\
\text{s.t.} \quad & \pi_i \le c_{ij}\xi_j + c_{i0}(1-\xi_j), \quad \forall i\in I, j\in S, \\
& \lambda_{j_1j_2} \le \xi_{j_1}, \qquad \forall j_1,j_2\in J, j_1<j_2, \beta_{j_1j_2}<0, \\
& \lambda_{j_1j_2} \le \xi_{j_2}, \qquad \forall j_1,j_2\in J, j_1<j_2, \beta_{j_1j_2}<0, \\
& \mu_{j_1j_2} \le 1-\xi_{j_2}, \qquad \forall j_1,j_2\in J, j_1<j_2, \beta_{j_1j_2}>0, \\
& \mu_{j_1j_2} \le \xi_{j_1}, \qquad \forall j_1,j_2\in J, j_1<j_2, \beta_{j_1j_2}>0.
\end{aligned} \tag{3.10}$$

Note, for this special case, there exists a simplified formula as follows

$$\begin{aligned}
RC(S) = \max_{\substack{\boldsymbol{\xi}\in\Xi,\\ \boldsymbol{\pi},\boldsymbol{\lambda}}} \; & \sum_{i\in I} d_i\pi_i - \alpha - \sum_{j\in J}\beta_j\xi_j - \sum_{j_1,j_2\in J, j_1<j_2} \beta_{j_1j_2}\lambda_{j_1j_2} \\
\text{s.t.} \quad & \pi_i \le c_{ij}\xi_j + c_{i0}(1-\xi_j), && \forall i\in I, j\in S, \\
& \lambda_{j_1j_2} \le \xi_{j_1}, && \forall j_1,j_2\in J, j_1<j_2, \\
& \lambda_{j_1j_2} \le \xi_{j_2}, && \forall j_1,j_2\in J, j_1<j_2, \\
& \lambda_{j_1j_2} \ge \xi_{j_1}+\xi_{j_2}-1, && \forall j_1,j_2\in J, j_1<j_2.
\end{aligned} \tag{3.11}$$

On Generating Initial Columns

Column generation algorithm starts with a group of initial columns such that the restricted master problem is feasible. For some problems, it is straightforward to find such columns. For example, one can use a set of all singletons as initial columns for a set covering problem. However, for this problem, it is difficult to directly construct a feasible set of columns except

for some special cases, such as the marginal model, see [55]. Actually, if P is not properly given, it may happen that no distribution can satisfy P. Next, we show a method to generate initial columns or prove such columns not exist by using the cross moment model as an example, i.e., P is given by (3.2). This method adopts the same idea of two phase simplex method.

The separation problem $Z_{sep}(S)$ for the cross moment model can be rewritten as,

$$\begin{aligned} Z_{sep}(S) = \max_{p_{\boldsymbol{\xi}} \geq 0} \quad & \textstyle\sum_{\boldsymbol{\xi} \in \Xi} Q(S, \boldsymbol{\xi}) p_{\boldsymbol{\xi}} \\ \text{s.t.} \quad & \textstyle\sum_{\boldsymbol{\xi} \in \Xi} p_{\boldsymbol{\xi}} = 1, \\ & \textstyle\sum_{\boldsymbol{\xi} \in \Xi | \xi_j = 1} p_{\boldsymbol{\xi}} = q_j \qquad \forall j \in J, \\ & \textstyle\sum_{\boldsymbol{\xi} \in \Xi | \xi_{j_1} = \xi_{j_2} = 1} p_{\boldsymbol{\xi}} = q_{j_1 j_2} \quad \forall j_1, j_2 \in J, j_1 < j_2. \end{aligned} \tag{3.12}$$

We construct a first phase master problem, which considers an arbitrary subset of columns as follows

$$\begin{aligned} \mathcal{MP}^{\diamond} = \min_{\substack{p_{\boldsymbol{\xi}} \geq 0, \\ \alpha^{\ddagger} \geq 0, \beta^{\ddagger} \geq 0,}} \quad & \alpha^{\ddagger} + \textstyle\sum_{j \in J | \xi_j = 1} \beta_j^{\ddagger} + \sum_{j_1, j_2 \in J | j_1 < j_2, \xi_{j_1} = \xi_{j_2} = 1} \beta_{j_1 j_2}^{\ddagger} \\ \text{s.t.} \quad & \textstyle\sum_{\boldsymbol{\xi} \in \hat{\Xi}} p_{\boldsymbol{\xi}} + \alpha^{\ddagger} = 1, \\ & \textstyle\sum_{\boldsymbol{\xi} \in \hat{\Xi} | \xi_j = 1} p_{\boldsymbol{\xi}} + \beta_j^{\ddagger} = q_j \qquad \forall j \in J, \\ & \textstyle\sum_{\boldsymbol{\xi} \in \hat{\Xi} | \xi_{j_1} = \xi_{j_2} = 1} p_{\boldsymbol{\xi}} + \beta_{j_1 j_2}^{\ddagger} = q_{j_1 j_2} \qquad \forall j_1, j_2 \in J, j_1 < j_2, \end{aligned} \tag{3.13}$$

where $\alpha^{\ddagger}$, $\beta_j^{\ddagger}$ and $\beta_{j_1 j_2}^{\ddagger}$ are nonnegative artificial variables corresponding to each constraint of the model.

There are two cases when solving $\mathcal{MP}^{\diamond}$:

(i) $\mathcal{MP}^{\diamond} = 0$, and we find a set of initial columns.

(ii) $\mathcal{MP}^{\diamond} > 0$. Then we solve pricing problem W, which can be written as:

$$W = \min_{\boldsymbol{\xi} \in \Xi} \left\{ \alpha^{\dagger} + \sum_{j \in J | \xi_j = 1} \beta_j^{\dagger} + \sum_{j_1, j_2 \in J | j_1 < j_2, \xi_{j_1} = \xi_{j_2} = 1} \beta_{j_1 j_2}^{\dagger} \right\},$$

, where $\alpha^{\dagger}, \beta_j^{\dagger}$ and $\beta_{j_1 j_2}^{\dagger}$ represent the corresponding duals obtained by solving $\mathcal{MP}^{\diamond}$. Similar to (3.11), W can be solved by following integer programming model:

$$\begin{aligned} W = \min_{\boldsymbol{\xi}, \boldsymbol{\lambda}} \quad & \alpha^{\dagger} + \textstyle\sum_{j \in J} \beta_j^{\dagger} \xi_j + \sum_{j_1, j_2 \in J, j_1 < j_2} \beta_{j_1 j_2}^{\dagger} \lambda_{j_1 j_2} \\ & \lambda_{j_1 j_2} \leq \xi_{j_1}, && \forall j_1, j_2 \in J, j_1 < j_2, \\ & \lambda_{j_1 j_2} \leq \xi_{j_2}, && \forall j_1, j_2 \in J, j_1 < j_2, \\ & \lambda_{j_1 j_2} \geq \xi_{j_1} + \xi_{j_2} - 1, && \forall j_1, j_2 \in J, j_1 < j_2, \end{aligned} \tag{3.14}$$

If $W < 0$, we add corresponding column to $\mathcal{MP}^{\diamond}$, and repeat the process. If $W \geq 0$, then the problem $Z_{sep}(S)$ is infeasible for any S.

Formulating the Supermodular Cut

Suppose that S^* and $\boldsymbol{p}^*$ are given. The corresponding cut $\eta \geq E(S^*, \boldsymbol{p}^*) + \sum_{j \in S \setminus S^*} \rho_j(S^*, \boldsymbol{p}^*)$ can be constructed by Algorithm 1, in which variables *cost* and $cost(j)$ respectively denote $E(S^*, \boldsymbol{p}^*)$ and $E(S^* \cup \{j\}, \boldsymbol{p}^*)$. As the number of non-zero items in $\boldsymbol{p}^*$ is $O(n)$, the computational complexity of Algorithm 1 is $O(n|I||J|)$.

For some special cases with further assumptions about the disruption probabilities, such that the conditional disruption probabilities can be calculated in $O(1)$ whereas n is significantly larger than $|J|$, we propose a more efficient Algorithm 2. Several examples fall

Algorithm 1 Construct a supermodular cut corresponding to S^* and $\boldsymbol{p}^*$

Input: A set of facilities S^* and a disruption distribution $\boldsymbol{p}^*$.

Output: A supermodular cut, i.e., $\eta \geq E(S^*, \boldsymbol{p}^*) + \sum_{j \in S \setminus S^*} \rho_j(S^*, \boldsymbol{p}^*)$

$cost \leftarrow 0$, $cost(j) \leftarrow 0$ for all $j \in J \setminus S^*$
for each $\boldsymbol{\xi} \in \Xi$ such that $p^*_{\boldsymbol{\xi}} > 0$ **do**
    $Q \leftarrow 0$, $Q(j) \leftarrow 0$ for all $j \in J \setminus S^*$
    for each $i \in I$ **do**
        $c_{\min} \leftarrow c_{i0}$
        for each $j \in S^*$ such that $\xi_j = 1$ **do**
            $c_{\min} \leftarrow \min\{c_{\min}, c_{ij}\}$
        end for
        $Q \leftarrow Q + d_i c_{\min}$
        for each $j \in J \setminus S^*$ **do**
            if $\xi_j = 1$ **then**
                $Q(j) \leftarrow Q(j) + d_i \min\{c_{\min}, c_{ij}\}$
            else
                $Q(j) \leftarrow Q(j) + d_i c_{\min}$
            end if
        end for
    end for
    $cost \leftarrow cost + p^*_{\boldsymbol{\xi}} Q$, $cost(j) \leftarrow cost(j) + p^*_{\boldsymbol{\xi}} Q(j)$ for all $j \in J \setminus S^*$
end for
return the supermodular cut $\eta \geq cost + \sum_{j \in S \setminus S^*} (cost(j) - cost)$

into this category, including the case in which $\tilde{\xi}_j$ for all $j \in J$ are independently distributed as assumed in [1], and the case in which the correlations of disruptions are induced from shared hazard exposure and follow certain known pattern as assumed in [51].

In Algorithm 2, variable γ denotes the joint disruption probability of facilities $j_1, ..., j_t$, that is, $\text{Prob}(\xi_{j_1} = 0, ..., \xi_{j_t} = 0)$. Step A of Algorithm 2 is to sort a list of at most $|J|$ facilities for $|I|$ times; therefore, Step A runs in $O(|I||J| \log |J|)$. Furthermore, if the conditional probability $\text{Prob}(\xi_{j_t} = 0 | \xi_{j_1} = 0, ..., \xi_{j_{t-1}} = 0)$ can be obtained in $O(1)$, Step B and Step C of Algorithm 2 both run in $O(|J|)$. Thus Lines 5-11 run in $O(|I||J|)$, and Lines 12-22 run in $O(|I||J|^2)$. In summary, Algorithm 2 runs in $O(|I||J| \log |J| + |I||J| + |I||J|^2)$, which is in the order of $O(|I||J|^2)$, and Algorithm 2 is more efficient than Algorithm 1.

3.2.2 A Multi-Start Tabu Search Algorithm

For RFLP, i.e., problem (3.1), although the cutting plane algorithm is able to find the optimal solution of the problem, the computational time grows tremendously with the size of the problem. This motivates us to develop an efficient heuristic algorithm in order to find high quality solutions for large scale problem within an acceptable amount of time.

Tabu search algorithm [36, 37], as an efficient method to solve combinatorial optimization problem, has been shown to be very successful on solving discrete location problems [5, 85, 62, 86, 30, 41]. The general idea of the tabu search algorithm is: Based on a current solution, search its neighbors, and determine the best one among all the neighbors as the candidate solution (and also as the current solution) with the consideration of tabu list and aspiration criterion. Update the tabu list by adding the candidate solution and removing some other

Algorithm 2 Construct a supermodular cut corresponding to S^* and $\boldsymbol{p}^*$

Input: A set of facilities S^* and a disruption distribution $\boldsymbol{p}^*$.

Output: A supermodular cut, i.e., $\eta \geq E(S^*, \boldsymbol{p}^*) + \sum_{j \in S \backslash S^*} \rho_j(S^*, \boldsymbol{p}^*)$

1: **for** each customer $i \in I$ **do** ▷ Step A (Lines 1-3)
2: Sort facilities $j \in S^*$ and the emergency facility $j = 0$ in the nondecreasing order of their distances to customer i. Let $L_i = \{j_1, ..., j_{|S^*|+1}\}$ be the list of indexes of facilities satisfying $c_{ij_1} \leq ... \leq c_{ij_{|S^*|+1}}$.
3: **end for**
4: $cost \leftarrow 0$, $cost(j) \leftarrow 0$ for all $j \in J \backslash S^*$
5: **for** each customer $i \in I$ **do**
6: $\gamma \leftarrow 1$
7: **for** $j_t \in L_i$, i.e.,$t = 1, ..., |S^*| + 1$ **do** ▷ Step B (Lines 7-10)
8: $cost \leftarrow cost + \gamma \times (1 - \text{Prob}(\xi_{j_t} = 0 | \xi_{j_1} = 0, ..., \xi_{j_{t-1}} = 0)) \times d_i c_{ij_t}$
9: $\gamma \leftarrow \gamma \times \text{Prob}(\xi_{j_t} = 0 | \xi_{j_1} = 0, ..., \xi_{j_{t-1}} = 0)$
10: **end for**
11: **end for**
12: **for** each $j \in J \backslash S^*$ **do**
13: **for** each customer $i \in I$ **do**
14: $\gamma \leftarrow 1$
15: Insert j into the sorted list L_i
16: **for** $j_t \in L_i$, i.e.,$t = 1, ..., |S^*| + 2$ **do** ▷ Step C (Lines 16-19)
17: $cost(j) \leftarrow cost(j) + \gamma \times (1 - \text{Prob}(\xi_{j_t} = 0 | \xi_{j_1} = 0, ..., \xi_{j_{t-1}} = 0)) \times d_i c_{ij_t}$
18: $\gamma \leftarrow \gamma \times \text{Prob}(\xi_{j_t} = 0 | \xi_{j_1} = 0, ..., \xi_{j_{t-1}} = 0)$
19: **end for**
20: Remove j from the sorted list L_i
21: **end for**
22: **end for**
23: **return** the supermodular cut $\eta \geq cost + \sum_{j \in S \backslash S^*} (cost(j) - cost)$

ones according to specified rules. If the candidate solution outperforms the currently-known best solution, update the best solution with the candidate solution. Repeat and continue the steps above until the stopping criteria is satisfied.

In this subsection, we design an efficient multi-start tabu search (MSTS) algorithm based on Michel and Van Hentenryck [62] to solve the large scale instances of the RFLP. The details of the MSTS algorithm are explained as follows.

Neighbors For a current solution S, we define the neighbors of S by simply flipping the status of one facility. Let S_j be the neighbor of S obtained by flipping the status of facility j, then we have $\mathbb{I}_{j\in S_j} = 1 - \mathbb{I}_{j\in S}$ and $\mathbb{I}_{k\in S_j} = \mathbb{I}_{k\in S}$ for all $k \neq j$, where $\mathbb{I}_{j\in S} \in \{0, 1\}$ is the indicator of whether facility j belongs to S. The set of all the neighbors of S can be written as $\{S_1, ..., S_J\}$. It is shown through numerical experiments that the MSTS algorithm with this simple definition of neighbors can perform quite satisfactorily.

Tabu list The candidate solution is the best one among the neighbors of the current solution, as long as this best neighbor is not prohibited by the tabu list. The tabu list is built and updated to avoid revisiting to exploited solutions. Let it be the counter of iterations. We use t_j for all $j \in J$ to implement the tabu list, which specifies the iteration number to which the status of facility j cannot be flipped, i.e., given current solution as S, its neighbor S_j cannot be selected as candidate solution if $it \leq t_j$. When the neighbor S_j is selected as candidate solution, t_j is updated as $t_j = it + tLen$, where $tLen$ is an integer keeping track of the length of the tabu list. We follow [62] and use a simple dynamic tabu list length scheme to update $tLen$.

Aspiration criterion We also employ the aspiration criterion to override a solution's tabu state.Recall that the tabu list prohibits some neighbors from being candidate solution, and the candidate solution is selected from the non-prohibited neighbors. The aspiration criterion we adopt is: even one neighbor is prohibited by the tabu list, we could still select it as candidate solution, as long as this solution outperforms the currently-known best solution. This simple criterion is commonly used, and performs well in our algorithm.

Stopping criterion We stop the algorithm when the currently-known best solution has not been improved for $StabilityLimit$ number of iterations. With larger $StabilityLimit$, the algorithm is able to overcome more local optimal solution, however with the tradeoff being longer computational time. $StabilityLimit$ is an important parameter to be tuned in the tabu search algorithm, and we discuss the tuning of this parameter in detail in Appendix-A.

Multi-start technique Unlike most existing tabu search algorithms, which run with only one initial solution, we apply the multi-start technique in this algorithm by executing the tabu search process above for $nStarts$ times with different randomly generated initial solutions. This multi-start technique has been shown to be very successful in combining with meta-heuristic algorithms [57, 74, 61]. $nStarts$ is an important parameter for the MSTS algorithm, and we will introduce the detailed parameter tuning process for it in appendix.

The overview of the MSTS algorithm is shown in Figure 3.1,

and its pseudo-code is shown in Algorithm 3.

In Algorithm 3, we use S to store the current solution, $S_{j\dagger}$ and $Obj^{\dagger}$ to store the candidate solution and its objective value, and S^{*} and Obj^{*} to store the best solution and the best

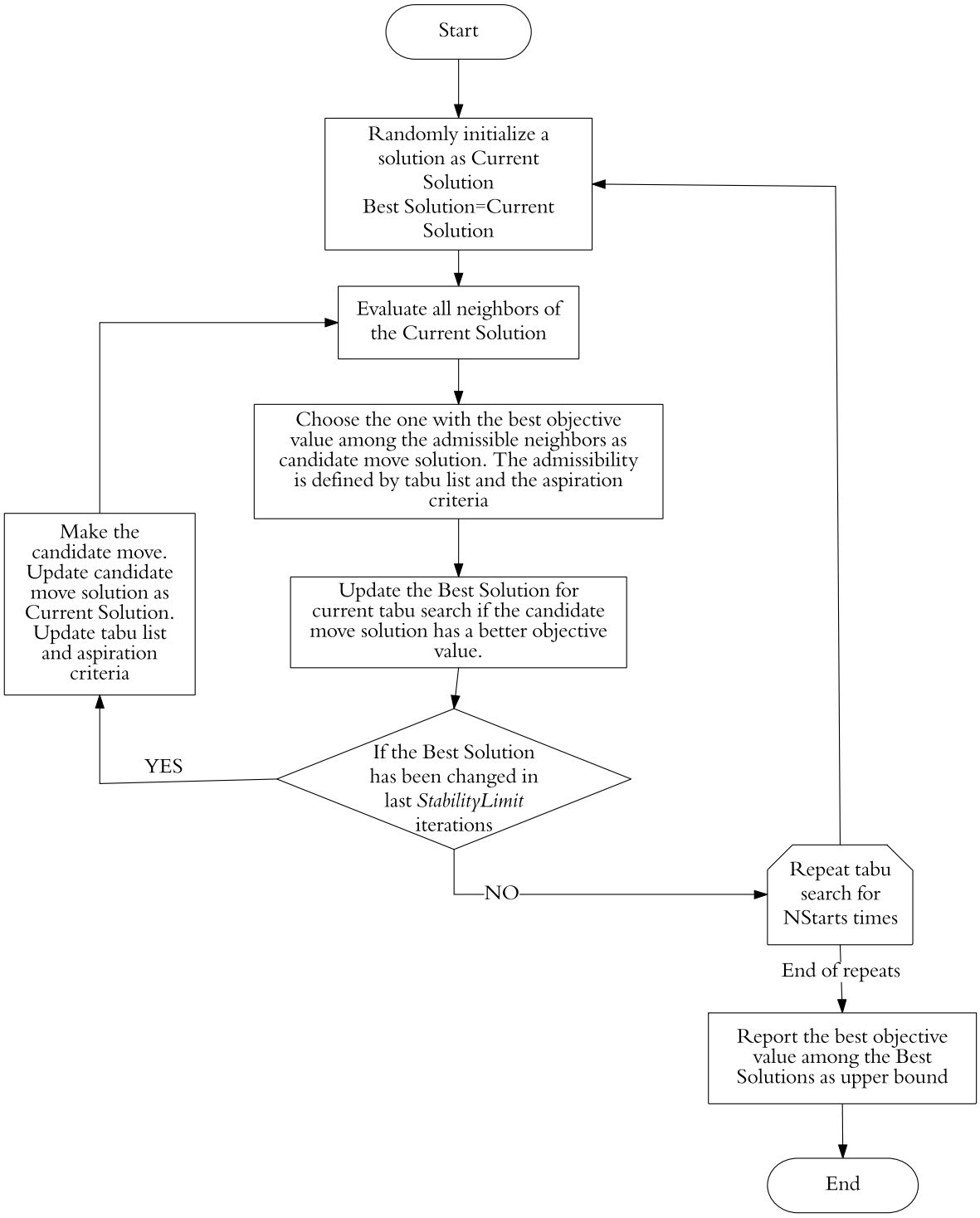

Figure 3.1: Flowchart of the multi-start tabu search

Algorithm 3 MSTS Algorithm

Output: The optimal solution S^* and the optimal objective value

$nS \leftarrow 0$
while $nS < nStarts$ **do**
  Initialize S, $Obj^\dagger \leftarrow Obj(S)$, and $Obj^* \leftarrow Obj(S)$ if $nS = 0$ ▷ Initialization
  $ns \leftarrow 0$, $it \leftarrow 0$, and $t_j \leftarrow 0$ for all $j \in J$
  while $ns < StabilityLimit$ **do**
    $it \leftarrow it + 1$
    for each facility $j \in J$ **do**
      if $it > t_j$ and $Obj(S_j) < Obj^\dagger$ **then** ▷ Tabu list
        $Obj^\dagger \leftarrow Obj(S_j)$
      else if $it \leq t_j$ and $Obj(S_j) < Obj^*$ **then** ▷ Aspiration criterion
        $Obj^\dagger \leftarrow Obj(S_j)$
      end if
    end for
    $j^\dagger \leftarrow random\{j \in J | Obj(S_j) = Obj^\dagger\}$, $S \leftarrow S_{j^\dagger}$ ▷ Update the candidate and the current solutions
    $t_{j^\dagger} \leftarrow it + tLen$ ▷ Update the tabu list
    Dynamic tabu list length scheme
    if $Obj^\dagger < Obj^*$ **then**
      $ns \leftarrow 0$
      $S^* \leftarrow S_{j^\dagger}$, $Obj^* \leftarrow Obj^\dagger$
    else
      $ns \leftarrow ns + 1$
    end if
  end while
  $nS \leftarrow nS + 1$
end while
return S^* as optimal solution and Obj^* as optimal objective value

objective value known so far. $Obj(S)$ is the function to calculate the objective value of model (3.1) for a given S.

3.3 Computational Studies

In this section, we apply the proposed algorithms, i.e., the cutting plane algorithm and the MSTS algorithm, to solve the robust RFLP, and demonstrate their advantages by comparing them with several existing algorithms.

All of the computational experiments are coded with C++, and implemented using ILOG CPLEX Academic Initiative Edition 12.6 (64-bit) in single thread model, on a Dell OptiPlex 9010 with one Intel 3.40 GHz CPU and 4G memory running Unix Operation System unless otherwise noted. For the cutting plane algorithm, the supermodular constraints are implemented as lazy constraints.

3.3.1 Results of Cutting Plane Algorithm

We first test the performance of the cutting plane algorithm for three categories of instances:

(i) We begin with instances with fully known distributions. First, we test instances with independent disruptions, and we compare the cutting plane algorithm with the algorithm proposed by [1], which to our knowledge is the best known algorithm for the problem with independent disruptions. Then, we apply the proposed algorithm to solve two types of instances with correlated disruptions.

(ii) For instances with marginal disruption probabilities, we firstly apply the worst-case distribution proved by [55] to model (3.3), then solve the worst-case problem using the

cutting plane algorithm, and finally compare the results with those of [55], which is obtained using the Benders decomposition algorithm.

(iii) For instances with cross moment of disruption probabilities, as there is no existing comparable algorithm, we solve the problem using the cutting plane algorithm directly without any comparison.

Instances with Independent Disruptions

For the instances with independent disruptions, we compare the cutting plane algorithm with the search-and-cut (SnC) algorithm proposed by Aboolian et al. [1]. Note that when the disruptions are independent and the disruption probability of each facility is given, model (3.3) turns into its special case with set P being a singleton, containing a single known distribution.

The computation is conducted based on the same data sets used in Aboolian et al. [1], which are from 1990 census data. In these data sets, each node represents a potential facility location and an aggregated demand point. We compare the performance of the two algorithms on instances with 50, 75, and 100 nodes, each representing one of the 50, 75, or 100 largest cities in the U.S. The demand, the fixed facility setup cost, and the transportation cost are the same as in Aboolian et al. [1] based on the data sets. As assumed in [1], the facility disruptions occur independently, and the disruption probabilities q_j are given as $q_j = 0.01 + 0.1\alpha e^{-D_j/400}$, where D_j is the great circle distance (in miles) between node j and New Orleans (D_j corresponds to d_j in [1], and we change the notation to avoid confusion). For each number of nodes, α varies from 1.0 to 1.5 in 0.05 increments giving total 33 instances. For the SnC algorithm, we fix the maximum assignment level R to the number of nodes

so that its objective values are comparable with ours. The neighborhood size is set to 3 to achieve the best performance according to their computational results (please refer to Aboolian et al. [1] for details). Both algorithms are solved to a 0.5% optimality gap or a maximum CPU time of 3600 seconds, whichever occurs first.

The computational results are summarized in Table 3.1. The first two columns identify the instance with the number of nodes and the value of α. The middle three columns show the results of the SnC algorithm and the last four columns show the results of the cutting plane algorithm. For each instance, we report the objective value provided by the SnC algorithm in the column titled "Obj" under the name "SnC Algorithm", and report the difference of the objective values between these two algorithms in the column titled "ΔObj" under the name "Cutting Plane Algorithm". In the column titled "Gap (%)" for both algorithms, we report the optimality gap, which is the relative gap between the lower and upper bounds when the corresponding algorithm terminates. For the cutting plane algorithm we proposed, the lower bound is the solution of the relaxed master problem, and the upper bound is the best feasible solution. For the SnC algorithm, the lower and upper bounds are designed by [1]. In short, the lower bound is obtained by solving a specific mixed integer programming problem, and the upper bound is obtained by implementing a neighborhood search starting from the current lower bound.

In addition, we report the CPU time (in seconds) in the column titled "CPU Time", and specially for the cutting plane algorithm we report the number of cuts added in the column titled "#Cuts".

From Table 3.1, we observe that both algorithms find solutions with the same objective values for all instances, i.e., the values of ΔObj are zero for all instances, but the cutting

Table 3.1: Results for cutting plane algorithm - Independent disruptions

Instance		SnC Algorithm			Cutting Plane Algorithm			
Nodes	α	Obj	Gap(%)	CPU Time	#Cuts	ΔObj	Gap(%)	CPU Time
50	1	1,020,180	0.38	47.62	203	0	0.36	1.73
50	1.05	1,021,040	0.50	48.23	210	0	0.00	1.33
50	1.1	1,021,890	0.46	57.49	206	0	0.02	2.18
50	1.15	1,022,750	0.49	69.86	194	0	0.13	1.91
50	1.2	1,023,610	0.47	76.52	214	0	0.28	1.65
50	1.25	1,024,470	0.47	83.27	201	0	0.00	1.72
50	1.3	1,025,330	0.48	96.34	212	0	0.50	1.85
50	1.35	1,026,180	0.48	106.80	241	0	0.00	2.13
50	1.4	1,026,980	0.49	133.95	226	0	0.49	1.68
50	1.45	1,027,780	0.40	160.20	272	0	0.00	2.45
50	1.5	1,028,490	0.49	165.61	231	0	0.33	2.10
75	1	1,148,490	0.41	216.09	389	0	0.00	14.16
75	1.05	1,149,490	0.42	240.59	303	0	0.00	13.39
75	1.1	1,150,490	0.47	251.72	309	0	0.00	11.47
75	1.15	1,151,490	0.46	310.70	318	0	0.00	11.07
75	1.2	1,152,500	0.49	387.15	333	0	0.00	15.28
75	1.25	1,153,500	0.44	473.07	389	0	0.00	14.30
75	1.3	1,154,510	0.43	551.13	407	0	0.00	19.05
75	1.35	1,155,520	0.50	600.89	395	0	0.00	22.85
75	1.4	1,156,520	0.49	731.02	373	0	0.00	23.15
75	1.45	1,157,530	0.50	990.51	392	0	0.47	20.05
75	1.5	1,158,540	0.50	1174.16	363	0	0.00	17.46
100	1	1,252,600	0.58	3678.63	832	0	0.44	207.05
100	1.05	1,253,600	0.73	3606.84	809	0	0.49	167.93
100	1.1	1,254,600	0.86	3604.15	917	0	0.42	244.51
100	1.15	1,255,610	0.98	3619.92	898	0	0.45	244.72
100	1.2	1,256,610	1.11	3657.92	805	0	0.48	204.52
100	1.25	1,257,620	1.25	3645.43	997	0	0.41	304.33
100	1.3	1,258,630	1.39	3625.49	896	0	0.41	207.29
100	1.35	1,259,640	1.51	3669.57	870	0	0.44	233.32
100	1.4	1,260,650	1.63	3687.87	937	0	0.50	242.25
100	1.45	1,261,660	1.77	3625.88	972	0	0.50	325.62
100	1.5	1,262,670	1.90	3600.57	1029	0	0.43	363.55

plane algorithm significantly outperforms the SnC algorithm in computational time. For example, for the instance with 50 nodes and $\alpha = 1$, the cutting plane algorithm solves the instance 27 times faster than the SnC algorithm. For other instances with 50 or 75 nodes which can be solved by the SnC algorithm appropriately (i.e., to an optimality gap of 0.5% within 3600 seconds), the cutting plane algorithm runs about 15-80 times faster than the SnC algorithm. For the instances with 100 nodes, the SnC algorithm fails to reduce the optimality gap to 0.5% within 3600 seconds, while the cutting plane algorithm is still able to solve these instances within around 6 minutes.

As previously discussed in subsection 3.2.1, there are theoretically an exponential number of cuts in the cutting plane algorithm. However, from the column titled "#Cuts" in Table 3.1, we observe that only a very small portion of them are needed, and the number of cuts needed increases moderately with the problem size. It is further observed that the efficiency of the SnC algorithm is sensitive to the value of α, as the CPU time increases acutely with the value of α. For example, for the instances with 50 nodes, as the value of α increases from 1 to 1.5, the CPU time of the SnC algorithm increases from 51.25 seconds to 160.95 seconds. This observation is consistent with those in Aboolian et al. [1]. In contrast, in the cutting plane algorithm, the CPU time remains almost unaffected with the increase of α within this range, and some instances with larger values of α are solved with even less computational effort. This observation indicates that the cutting plane algorithm is algorithmically stable and robust.

We also test the algorithms with randomly generated instances. The instances are randomly generated as follows: the location of facilities and customers are uniformly distributed over $[0, 100] \times [0, 100]$, and the unit transportation cost c_{ij} is assumed to be

proportional to the Euclidean distance in the plane. For each facility $j \in J$, the initial setup cost f_j randomly generated in $[2000, 5000]$ and the failure probability q_j is uniformly generated in $[0, 0.1]$. Demand of each customer $i \in I$ is generated uniformly in $[0, 20]$. We set the unit missed demand cost $d_{i\mathcal{E}} = \phi = 200$ ($d_{i\mathcal{E}} = 200$) for all $i \in I$. In such settings, the missed demand penalty occurs only when all open facilities fail. The computational results are summarized in Tables 3.2. The first two columns identify the instances with the number of facilities ($|J|$) and customers ($|I|$). Instance sizes are ranged from 50 facilities and 50 customers to 100 facilities and 150 customers. For each problem size, we generate 10 random instances and hence, solve at total of 150 instances. We report the average values of the indicators as in Table 3.1 except the CPU time, which we report the minimal, average and maximal CPU time. In addition, we report the number of instances that cannot be solved to a gap of 0.5% in 3600 seconds in the column titled "#Unsolved". We observe that the cutting plane algorithm is significantly faster than the SnC algorithm on these random instances. When the instance size becomes large, the sync algorithm is unable to solve almost all of the instances. For the cutting plane algorithm, it takes an average of 95.9 seconds and a maximum of 317.97 seconds to solve the 10 random instances of 100 facilities and 150 customers and the average optimality gaps it provides are much smaller than those given by the SnC algorithm.

Table 3.2: A comparison of algorithms performance – random instances

		SnC Algorithm						Cutting Plane Algorithm						
$\|J\|$	$\|I\|$	Upper Bound	Gap (%)	CPU Time			#Unsolved	# Cuts	Upper Bound	Gap (%)	CPU Time			#Unsolved
				Min	Ave	Max					Min	Ave	Max	
50	50	25185	0.38	9.29	44.52	203.71	0	76.9	25185	0.06	0.07	0.13	0.23	0
50	75	32750	0.41	23.40	293.01	1064.16	0	100.6	32750	0.05	0.12	0.25	0.44	0
50	100	40894	0.87	567.88	1874.27	3634.02	3	160.8	40894	0.15	0.28	0.76	1.72	0
50	125	48318	1.48	280.18	3269.28	3672.89	8	207.1	48318	0.13	0.31	1.78	3.50	0
50	150	53262	2.34	3622.99	3654.32	3716.95	10	289.7	53262	0.12	1.36	3.68	6.42	0
75	50	24511	0.41	26.05	154.31	546.36	0	96.2	24511	0.08	0.10	0.27	0.86	0
75	75	32901	0.98	189.58	1897.27	3649.46	4	185.9	32901	0.14	0.30	1.13	2.06	0
75	100	39232	1.74	1685.69	3236.77	3646.23	7	292.4	39232	0.21	0.94	4.04	12.56	0
75	125	45924	2.52	3609.52	3651.35	3719.08	10	289.6	45924	0.21	1.00	5.66	15.35	0
75	150	52339	3.57	3600.34	3655.44	3726.47	10	506.0	52338	0.24	3.71	28.69	85.77	0
100	50	23967	0.41	116.71	525.44	839.72	0	115.7	23967	0.10	0.19	0.43	0.76	0
100	75	32433	1.04	533.84	3032.62	3685.62	8	201.0	32433	0.08	0.80	1.86	4.21	0
100	100	38715	2.41	197.73	3016.86	3678.37	8	322.5	38697	0.27	0.46	6.60	20.04	0
100	125	45935	4.18	3604.48	3649.61	3715.39	10	601.3	45935	0.40	3.09	44.69	172.78	0
100	150	51676	4.45	3629.06	3695.84	3744.53	10	785.1	51661	0.34	27.69	95.90	317.97	0

Instances with correlated disruptions

In this subsection, we perform a computational study for the reliable facility location instances with correlated disruptions. We study two examples of correlated disruptions that are used in the related literatures to demonstrate the capability of the proposed algorithm.

Correlation induced from shared hazard exposure

When the disruption mechanism is well understood, such as when correlations are induced from shared hazard exposure, the disruption can be conditioned on the states of the hazard source. This method to characterize the disruption correlation has been studied in recent literature studying reliable facility location problems [51, 55]. We follow their study, assuming that there exists a hazard source at the origin and that hazardous event occurs with a certain probability α. When the hazard occurs, the disaster propagates and disrupts facilities; a facility j fails with a probability $e^{-|D_j|/\theta}$, where $|D_j|$ is the distance of location j to the hazard source and θ is a parameter that characterizes the strength of the disruption propagation effect. The marginal probability of facility disruption at location j is given by $q_j = \alpha e^{-|D_j|/\theta}$. We adopt the same instances used in subsection 3.3.1 except that the facility failure probability is defined by the above conditional probability. Similar to Lu et al. [55], we examine three combinations of the parameters α and θ: $\alpha = 0.1$ and $\theta = 100$ (the probability of hazard is low and the propagation effect is low, which we refer to as the low-risk scenario), $\alpha = 0.2$ and $\theta = 200$ (the probability of hazard is moderate and the propagation effect is moderate, which we refer to as the moderate risk scenario), $\alpha = 0.3$

and $\theta = 800$ (the probability of hazard is high and the propagation effect is high, which we refer it to as the high risk scenario).

Table 3.3: Instances with correlation induced from shared hazard exposure

Nodes	α	θ	Cutting Plane Algorithm – Correlated				Cutting Plane Algorithm – Uncorrelated			
			# Cuts	Upper Bound	Gap (%)	CPU Time	# Cuts	Upper Bound	Gap (%)	CPU Time
50	0.1	200	152	995,407	0.48	0.95	183	995,281	0.17	1.00
50	0.2	400	182	1,025,260	0.00	1.54	243	1,022,640	0.23	1.59
50	0.3	800	667	1,139,280	0.49	24.31	461	1,109,550	0.06	16.20
75	0.1	200	231	1,120,330	0.00	7.02	222	1,120,180	0.00	5.86
75	0.2	400	393	1,158,080	0.00	20.73	447	1,154,140	0.00	22.47
75	0.3	800	2039	1,295,240	0.31	1718.38	1659	1,260,560	0.40	631.11
100	0.1	200	631	1,224,000	0.31	87.34	596	1,223,790	0.40	105.25
100	0.2	400	784	1,262,520	0.45	273.24	1018	1,257,740	0.29	217.01
100	0.3	800	2617	1,380,400	3.05	3600.00	2561	1,344,720	1.35	3600.01

Table 3.3 highlights the computational results. The aggregated column titled "Cutting Plane Algorithm – Correlated" reports the results of the proposed algorithm for instances with correlated disruptions, and the column titled "Cutting Plane Algorithm – Uncorrelated" reports the results for corresponding instances with same marginal failure probabilities but uncorrelated failures. From the result, we observe that the cutting plane algorithm is able to solve most instances. For the same instance, the objective value, the solution time, and the number of cuts added all increase with the risk parameters. Compared with instances in which the disruptions are independent, the objective is larger when the disruptions are correlated, and the difference increases from the low risk scenario to the high risk scenario.

Correlations specified by the beta-geometric distribution

In this subsection, we consider facility status as a series of coin flip experiments, with "heads" and "tails" corresponding to "working" and "failure". If the coin flip experiments are independent and the probability of getting "head" is a known constant p, then X, the number of trials needed to get the first "head" conforms to the (shifted) geometric distribution with probability mass function $P(X = k) = (1-p)^{k-1}p$ and is exactly the case of uncorrelated disruptions, where $P(X = k)$ is the probability that a customer is served by its k-th closest facility in the context of our problem. If the probability of getting "head" is not a constant but follows a beta distribution $Beta(\alpha, \beta)$, then the trials are positively correlated and X conforms to the beta-geometric distribution. The beta-geometric distribution and the closely related beta-binomial distribution that models the number of "heads" in N trials have been used in various fields to model correlation failures, such as in biometrics (Griffiths [38], Weinberg and Gladen [91]), computer science (Nicola and Goyal [67]) and marketing (Fader and Hardie [31]). Although Li and Ouyang [51] use the beta-binomial distribution to model the correlation and derive the conditional probability, we find beta-geometric distribution approaches the studied problem more directly and the resulting formula is essentially the same. We denote the beta-geometric distribution $BG(\alpha, \beta)$, which has two parameters of the beta distribution. It has the probability mass function:

$$P(X = t) = \frac{B(\alpha + 1, \beta + t - 1)}{B(\alpha, \beta)}$$

, where $B(\cdot,\cdot)$ is the beta function. To avoid dealing with the beta function, one can compute $P(T=t)$ using the following forward-recursion formula:

$$P(T=t) = \begin{cases} \frac{\alpha}{\alpha+\beta} & t=1 \\ \frac{\beta+t-2}{\alpha+\beta+t-1} P(T=t-1) & t=2,3,... \end{cases}$$

Let r_t denote the conditional probability that the t-th closest facility fails given that all the closer facilities fail. We have:

$$r_t = \frac{\beta+t-1}{\alpha+\beta+t-1}, t=1,2,3,...$$

The probability that the closest facility fails is $r_1 = \frac{\beta}{\alpha+\beta}$ and the formula for r_t shows that the disruptions are positively correlated because $r_{t_1} \leq r_{t_2}$ for all $t_1 \leq t_2$, and the larger the value $\alpha+\beta$ the less significant the correlation. We refer readers to Fader and Hardie [31] for step-by-step details of the derivations.

Table 3.4: Instances with correlation specified by beta-geometric distribution

Nodes	$\frac{\beta}{\alpha+\beta}$	$\alpha+\beta$	Cutting Plane Algorithm – Correlated				Cutting Plane Algorithm – Uncorrelated			
			# Cuts	Upper Bound	Gap (%)	CPU Time	# Cuts	Upper Bound	Gap (%)	CPU Time
50	0.01	10	170	1,004,760	0.42	1.22	246	1,003,270	0.00	1.14
50	0.01	20	226	1,004,000	0.00	1.22	246	1,003,270	0.00	1.14
50	0.01	50	214	1,005,120	0.29	1.14	246	1,003,270	0.00	1.14
50	0.05	10	372	1,064,520	0.26	5.74	308	1,056,460	0.30	2.83
50	0.05	20	338	1,059,770	0.00	5.98	308	1,056,460	0.30	2.83
50	0.05	50	307	1,057,760	0.00	5.22	308	1,056,460	0.30	2.83
50	0.1	10	874	1,137,710	0.41	31.64	663	1,126,010	0.49	17.55
50	0.1	20	667	1,131,750	0.41	23.72	663	1,126,010	0.49	17.54
50	0.1	50	606	1,127,900	0.42	16.14	663	1,126,010	0.49	17.55
75	0.01	10	339	1,129,830	0.00	9.62	302	1,128,720	0.00	8.45
75	0.01	20	308	1,129,230	0.00	8.24	302	1,128,720	0.00	8.46
75	0.01	50	291	1,128,920	0.00	7.35	302	1,128,720	0.00	8.45
75	0.05	10	553	1,193,330	0.49	74.94	617	1,187,050	0.00	42.50
75	0.05	20	744	1,189,910	0.50	69.70	617	1,187,050	0.00	42.45
75	0.05	50	542	1,188,140	0.11	46.13	617	1,187,050	0.00	42.47
75	0.1	10	1972	1,277,540	0.49	1058.79	1439	1,263,000	0.39	448.38
75	0.1	20	1693	1,269,740	0.43	596.47	1439	1,263,000	0.39	449.82
75	0.1	50	1489	1,266,790	0.50	527.93	1439	1,263,000	0.39	448.77
100	0.01	10	562	1,233,750	0.00	137.64	577	1,232,780	0.35	136.29
100	0.01	20	702	1,233,230	0.41	98.34	577	1,232,780	0.35	136.15
100	0.01	50	577	1,232,960	0.44	106.82	577	1,232,780	0.35	136.46
100	0.05	10	1540	1,289,910	0.45	1960.06	1484	1,284,260	0.30	1157.08
100	0.05	20	1303	1,287,280	0.49	1103.06	1484	1,284,260	0.30	1162.74
100	0.05	50	990	1,285,280	0.47	601.64	1484	1,284,260	0.30	1159.95
100	0.1	10	2886	1,361,450	3.47	3600.48	2335	1,348,630	2.29	3600.00
100	0.1	20	2488	1,356,680	2.32	3600.00	2339	1,348,630	2.27	3600.00
100	0.1	50	2451	1,347,850	1.81	3600.00	2335	1,348,630	2.29	3600.00

Computational results for reliable supply chain network design problems in which the correlation is specified by the beta-geometric distribution are shown in Table 3.4. The instances are the same as those used in the previous section, except that the facility failure probability is defined by the beta distribution. We vary the value of $\frac{\beta}{\alpha+\beta}$ and $\alpha+\beta$ to study the effect of the marginal disruption rate and the degree of correlation. Each column in Table 3.4 has the same meaning as in the previous subsection. From Table 3.4, we observe that our algorithm can solve most instances very efficiently. Generally speaking, for the same instance, the solution time, the objective value and the number of cuts added increase with the marginal disruption rate and the degree of correlation. Compared with the uncorrelated counterpart instances, the instances with correlated disruptions have larger objective value and the difference increases with the degree of the correlation. These observations are consistent with the results in Table 3.3. There are two discrepancy observations, that is, the third row has an upper bound larger than the second row for the correlated case, and in the last row, the correlated case has an upper bound smaller than the uncorrelated case. We believe this is caused by the existence of an optimality gap.

Instances with Marginal Disruption Probabilities

For instances with marginal disruption probabilities, Lu et al. [55] prove the worst-case distribution for this problem. Note that when applying the worst-case distribution they proved, model (3.3) turns into its special case with known distribution in set P. Therefore, we solve the worst-case problem using the cutting plane algorithm, and compare our results with those of [55], which is obtained by solving a stochastic programming problem using a Benders decomposition algorithm.

The same assumptions from [55] about marginal disruption probabilities are adopted. A disastrous event occurs at a certain source with probability β (which corresponds to α in [55], and we change the notation to avoid confusion). When the disaster occurs, it propagates and causes disruptions to facilities. The marginal disruption probability of facility j, denoted as q_j, decreases exponentially with its distance from the source, denoted as D_j. We assume that q_j is given by $q_j = \beta e^{-D_j/\theta}$, where the parameter θ characterizes the strength of disruption propagation effect. For demand, the fixed setup cost, and transportation cost, we use the same data sets as in subsection 3.3.1, i.e., the data sets in [1]. For both algorithms, we solve to a 0.5% optimality gap.

The results are shown in Table 3.5. Each instance is characterized by the combination of the number of nodes, the source disaster probability β, and the disruption propagation factor θ. As in [55], we assume that the number of nodes is chosen from value the set $\{50, 75, 100\}$, $\beta \in \{0.1, 0.2, 0.3\}$, and $\theta \in \{200, 400, 800\}$, giving us 27 total instances in total. For each instance, we compare the objective value, the optimality gap, and the CPU time in seconds for these two algorithms. From the results, we observe that the cutting plane algorithm out performs the Benders decomposition algorithm in terms of computational time, and in many cases the cutting plane algorithm is able to find better solutions.

Instances with Cross Moment of Disruption Probabilities

For instances with cross moment of disruption probabilities, there is no existing comparable algorithm, so we solve the problem with the cutting plane algorithm directly to show that this algorithm can provide satisfactory solution in an efficient way.

Table 3.5: Results for cutting plane algorithm - Marginal disruption probabilities

Instance			Lu et al. [55]			Cutting Plane Algorithm			
Nodes	β	θ	Obj	Gap(%)	CPU Time	#Cuts	ΔObj	Gap(%)	CPU Time
50	0.1	200	998,603	0.42	20.08	132	0	0.01	2.21
		400	1,071,085	0.47	23.60	174	-545	0.01	2.56
		800	1,426,616	0.48	28.77	141	-3,458	0.00	2.32
50	0.2	200	1,006,565	0.42	22.13	131	0	0.32	2.27
		400	1,140,029	0.43	28.32	168	-883	0.37	3.34
		800	1,843,037	0.50	19.10	146	-3,611	0.39	2.01
50	0.3	200	1,012,901	0.42	21.68	158	0	0.01	2.39
		400	1,209,215	0.49	33.43	187	-3,941	0.38	2.62
		800	2,258,727	0.47	17.46	114	-3,033	0.01	1.85
75	0.1	200	1,123,501	0.02	118.61	234	0	0.01	19.02
		400	1,201,468	0.03	146.94	258	2,404	0.29	19.61
		800	1,604,318	0.01	148.79	191	0	0.01	12.72
75	0.2	200	1,135,638	0.48	110.74	231	-3,275	0.01	16.74
		400	1,281,522	0.28	124.93	301	-1,993	0.01	24.35
		800	2,090,157	0.47	102.43	216	-4,928	0.41	17.49
75	0.3	200	1,141,226	0.49	110.96	208	0	0.01	15.58
		400	1,357,590	0.27	121.64	226	0	0.01	17.97
		800	2,568,298	0.37	111.07	272	-5,337	0.46	21.45
100	0.1	200	1,231,258	0.49	727.66	467	-3,583	0.42	146.22
		400	1,299,906	0.49	485.45	416	-149	0.01	136.31
		800	1,738,672	0.49	634.66	446	-2,879	0.30	171.57
100	0.2	200	1,237,231	0.49	688.72	596	-303	0.37	161.96
		400	1,380,660	0.48	461.10	493	-1,128	0.41	157.31
		800	2,240,389	0.39	384.49	355	0	0.49	95.16
100	0.3	200	1,245,840	0.23	943.04	514	0	0.01	163.52
		400	1,460,349	0.48	440.13	402	-1,192	0.46	105.30
		800	2,743,006	0.49	311.32	322	-3,644	0.35	93.48

The cross moment matrix of disruption probabilities is denoted as matrix $\boldsymbol{Q}_{|J|\times|J|}$, where the entry q_{jk} corresponds to the joint disruption probability of facility j and facility k. Specially, the diagonal entry, i.e., q_{jj}, is the marginal disruption probability of facility j. We generate random matrix Q in the following way: first, we assume a independent distribution with marginal probability the same as in subsection 3.3.1, i.e., $q_j = \beta e^{-D_j/\theta}$; we then generate a sample with size of $5 \times |J|$ from the independent distribution and let the matrix Q be a cross moment matrix that describes the sample distribution. For demand, the fixed setup cost, and the transportation cost, we use the same data sets as in subsection 3.3.1, i.e., the data sets in [1]. The results are shown in Table 3.6. The instances considered, i.e., the combination of the number of nodes and the values of β and θ, are the same as in Table 3.5.

From the results, we observe that the cutting plane algorithm is able to solve samll instances efficiently, e.g., 10 nodes or 20 nodes. We also notice that when the value of θ increases from 200 to 800, the solution time increases significantly. This increase factor also increases with instance size. Therefore, the instances with 50 nodes and $\theta = 400$ or 800 become extremely difficult for the proposed algorithm to solve.

3.3.2 Results of Multi-start Tabu Search

In this subsection, we study the performance of the MSTS algorithm for solving RFLP instances with independent disruptions.

According to our discussion in subsection 3.2.2, the number of start $nStarts$ and the stopping criterion $StabilityLimit$ significantly affect the efficiency of the MSTS algorithm. For all the computations below, we set $nStarts = 20$ and $StabilityLimit = 20$, according

Table 3.6: Results for cutting plane algorithm - Cross moment of disruption probabilities

Instance			Cutting Plane Algorithm			
Nodes	β	θ	#Cuts	Obj	Gap(%)	CPU Time
10	0.1	200	14	539,356	0.00	0.61
		400	13	539,356	0.00	0.78
		800	21	597,117	0.00	4.46
	0.2	200	13	545,225	0.00	0.62
		400	18	554,397	0.00	1.85
		800	19	615,285	0.00	6.48
	0.3	200	18	542,659	0.11	1.50
		400	15	581,538	0.00	0.55
		800	23	652,007	0.00	6.00
20	0.1	200	42	770,300	0.01	6.55
		400	40	781,371	0.00	13.49
		800	53	807,559	0.00	171.16
	0.2	200	45	776,661	0.00	7.86
		400	55	794,413	0.00	39.92
		800	76	852,476	0.01	830.31
	0.3	200	40	773,959	0.00	5.95
		400	40	816,136	0.00	119.79
		800	74	887,295	0.16	3518.79
50	0.1	200	136	997,204	0.01	1527.70
		400	50	1,102,230	287.51	3600.25
		800	1	4,507,876	6678.76	3603.36
	0.2	200	114	999,456	0.49	869.38
		400	2	2,526,397	1128.19	3693.15
		800	1	40,624,200	60989.00	3604.67
	0.3	200	161	1,003,734	0.00	2346.48
		400	2	2,466,342	921.68	3618.49
		800	1	8,048,309	12002.70	3614.29

to the parameter tuning process in the appendix. We also study the acceleration of the MSTS algorithm when applying the parallelization techniques. Details of the acceleration are available in appendix-A.

We test the proposed multi-start tabu search algorithm's performance with the instances used in Aboolian et al. [1]. Table 3.7 summarizes the results. The results from the proposed exact algorithm is also listed for comparison. For the MSTS, because it is a randomized algorithm, we perform 10 runs for each instance to reduce the occasionality. We report the best, worst and average objective values obtained by the heuristic algorithm among 10 runs. Table 3.7 shows that the MSTS finds a very high quality solution much faster than the exact algorithm. For many instances, it finds the optimal solution in all 10 runs.

The algorithm is also tested with very large random instances. The random instances are generated in the same way as in subsection 3.3.1. The size of the instances ranges from 200 potential facilities and 200 customers to 600 potential facilities and 800 customers. The instances are also solved with the cutting plane algorithm with a time limit of 7200 seconds. Table 3.8 reports the results. The cutting plane algorithm fails to solve all of these instances to optimality within the time limit. The upper bounds are the best objective values that have been found when the algorithm terminates. For the MSTS, we run the algorithm 10 times and report the minimal, average, and maximal values of the upper bound and the CPU time used. As can be seen from Table 3.8, the MSTS is able to solve extremely large instances within a reasonable amount of time. With respect to the solution quality, the MSTS find solutions with equal or better quality than the cutting plane algorithm even in the worst case among the 10 runs (except the worst case for 400 facilities and 400 customers).

Table 3.7: Performance of the multi-start tabu search algorithm on benchmark instances

Nodes	α	Cutting Plane Algorithm		Multi-start Tabu search				
		Upper Bound	CPU Time	Upper Bound			Iterations	CPU Time
				Best	Ave	Worst		
50	1	1,020,180	1.73	1,020,180	1,020,180	1,020,180	1250.2	0.4331
50	1.1	1,021,890	2.18	1,021,890	1,021,890	1,021,890	1215.7	0.4194
50	1.2	1,023,610	1.65	1,023,610	1,023,610	1,023,610	1245.8	0.4333
50	1.3	1,025,330	1.85	1,025,330	1,025,330	1,025,330	1285.1	0.4419
50	1.4	1,026,980	1.68	1,026,980	1,026,980	1,026,980	1109.6	0.3854
50	1.5	1,028,490	2.10	1,028,490	1,028,500	1,028,600	1093.4	0.3767
75	1	1,148,490	14.16	1,148,490	1,148,490	1,148,490	1391.9	1.2463
75	1.1	1,150,490	11.47	1,150,490	1,150,490	1,150,490	1410.9	1.3171
75	1.2	1,152,500	15.28	1,152,500	1,152,500	1,152,500	1409.9	1.2999
75	1.3	1,154,510	19.05	1,154,510	1,154,510	1,154,510	1405.7	1.2871
75	1.4	1,156,520	23.15	1,156,520	1,156,520	1,156,520	1402.8	1.2916
75	1.5	1,158,540	17.46	1,158,540	1,158,540	1,158,540	1367.4	1.2461
100	1	1,252,600	207.05	1,252,600	1,252,600	1,252,600	1532.5	2.9813
100	1.1	1,254,600	244.51	1,254,600	1,254,600	1,254,600	1510.3	2.8418
100	1.2	1,256,610	204.52	1,256,610	1,256,700	1,257,500	1447.1	2.6782
100	1.3	1,258,630	207.29	1,258,630	1,258,700	1,259,310	1471.7	2.7327
100	1.4	1,260,650	242.25	1,260,650	1,260,650	1,260,650	1508.8	2.8135
100	1.5	1,262,670	363.55	1,262,670	1,262,670	1,262,670	1501.8	2.8676

Table 3.8: Performance of the multi-start tabu search algorithm on large size random instances

$\|J\|$	$\|I\|$	Cutting Plane Algorithm		Multi-start tabu search					
				Upper Bound			CPU Time		
		Upper Bound	CPU Time	Best	Ave	Worst	Min	Ave	Max
200	200	58137	7200.87	58137	58137	58137	16.91	22.87	29.79
200	400	102114	7200.29	100932	100956	100998	36.28	49.33	65.02
200	600	131082	7200.03	128270	128463	128967	56.69	74.92	99.46
200	800	160749	7200.00	158295	158567	158809	67.19	99.11	127.66
400	200	61837	7200.01	61213	61386	61514	126.03	250.60	367.22
400	400	95272	7200.01	94599	94805	95321	280.80	410.00	545.15
400	600	128505	7200.02	123501	123655	124069	394.94	784.99	1149.21
400	800	156446	7200.00	151960	152294	152520	463.24	794.04	1495.11
600	200	63292	7200.02	61078	61131	61188	414.44	736.44	1171.49
600	400	101333	7200.02	97661	97902	98189	930.21	1640.13	2399.03
600	600	129135	7200.03	123920	124227	124495	1755.01	2492.36	3521.72
600	800	157959	7200.00	151738	152309	152783	1787.61	3527.27	4990.80

Acceleration of MSTS Algorithm with Parallelization

Because each start of the tabu search is independent of the others, the algorithm can be easily paralleled by letting each start run in parallel. The parallelization of the algorithm is implemented using OpenMP. This set of computations is done using the Newton high performance compute (HPC) cluster (https://newton.utk.edu/). We limit our parallelization to 48 processors because this is the current limit on the cluster. In Figure 3.2 we present speedup of the proposed MSTS with respect to the number of processors. The algorithm is applied to instances with 50, 75, and 100 nodes with $\alpha = 1$ and the number of starts is set to 500. From Figure 3.2, we observe that it achieves an almost linear speedup.

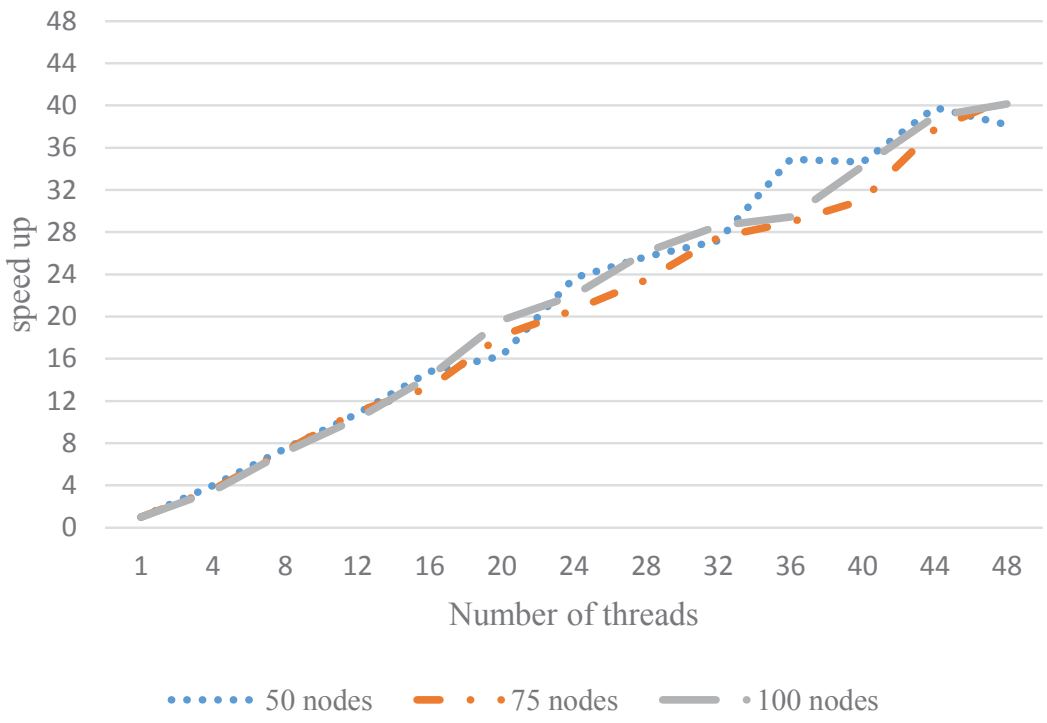

Figure 3.2: Parallel multi-start tabu search

Chapter 4

Efficient Solution Methods for a General r-interdiction Median Problem with Fortification

4.1 Problem Definition and Formulation

Consider an existing supply network with a set of operating facilities $J = \{1, ..., p\}$ and a set of customer demand aggregation points $I = \{1, ..., |I|\}$. Each customer $i \in I$ has demand given by d_i. The unit shipment cost from facility $j \in J$ to customer $i \in I$ is c_{ij}. The efficiency of the system is measured by the total weighted distance between customers and the facilities. The capacity of a facility is assumed to be unlimited. Thus, a customer is always served by its nearest working facility. Furthermore, we assume there is an emergency facility 0 to model the costs of lost sales. If the demand of customer $i \in I$ is not served, a unit penalty cost c_{i0} is incurred. Without loss of generality, we assume that $c_{i0} \geq c_{ij}$ for all

$i \in I$ and $j \in J$; that is, any customer will be served if there is at least one available facility. Let J^+ denote the set open facilities including the emergency facility, i.e., $J^+ = \{0, 1, ..., p\}$

.

Facilities face two types of disruption risks simultaneously: probabilistic, exogenous risks and worst-case, endogenous risks. The two types of risks are independent from each other. A facility fails if either of these two disruption risks takes effect. For exogenous risks, facility j fails with probability q_j. Facility disruptions caused by exogenous risks are independent across facilities. Endogenous risks are modeled by an attacker with r interdict resources. Each unit of these resources can be used to attack a facility; however, two or more units cannot be used on the same facility. If a facility is not fortified, an attack succeeds for sure. However, if the facility is fortified, the probability of successful attack is w. The total number of fortified locations is bounded above by h due to limited defensive resources. The problem is to determine h locations to fortify to minimize expected transportation costs anticipating worst-case attacks.

The problem can be viewed as a leader-follower game, as illustrated by Figure 4.1: the leader, that is, the network planner or defender, aims to minimize costs by making fortification decisions, then the follower, that is, the attacker, tries to interdict the network to maximize its damage. The following bi-level model is formulated by adding the probabilistic disruption factor to the RIMF-p model proposed by Zhu et al. [96].

Defender's decision variables:

$$z_j = \begin{cases} 1, & \text{if facility } j \text{ is fortified;} \\ 0, & \text{otherwise.} \end{cases}$$

Attacker's decision variables:

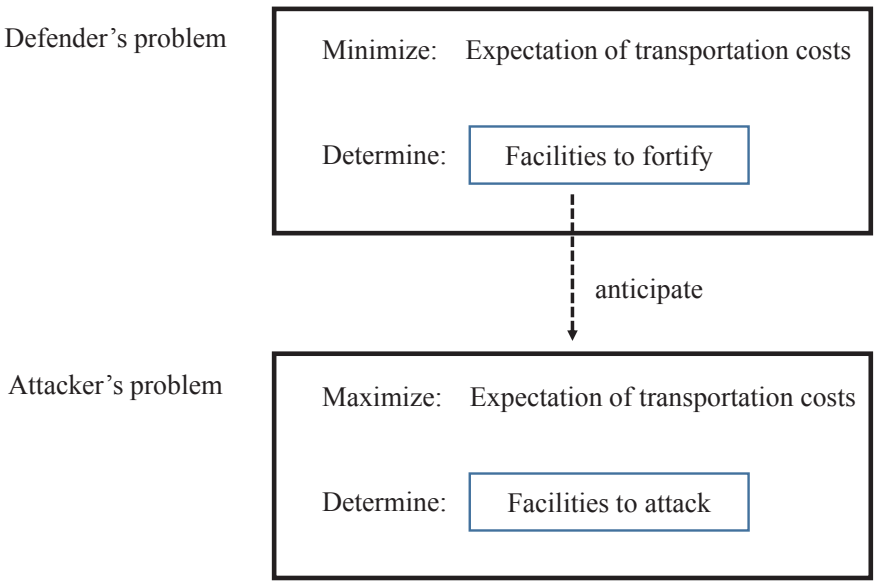

Figure 4.1: Bi-level problem

$$s_j = \begin{cases} 1, & \text{if facility is attacked;} \\ 0, & \text{otherwise.} \end{cases}$$

The fortification problem (FP) is written as:

$$\min F(z) \tag{4.1}$$

$$\begin{aligned} \text{s.t.} \quad & \textstyle\sum_{j \in J} z_j \leq h && \forall j \in J \\ & z_j \in \{0,1\}, && \forall j \in J \end{aligned} \tag{4.2}$$

$F(z)$ is the expected transportation costs under worst-case attacks for a fortification decision z and is computed by solving the attacker's problem.

A facility j's failure probability p_j is an expression relating to s_j:

$$p_j = 1 - (1 - q_j)(1 - s_j w^{z_j}) \tag{4.3}$$

When $z_j = 0$, i.e., the facility is not fortified, $w^{z_j} = 1$, and the facility is disrupted for sure when it is attacked, i.e., $p_j = 1$, as long as $s_j = 1$. Otherwise, the facility fails when either the probabilistic disruption, with probability q_j, takes place or an intentional attack succeeds, with probability w.

Then we can compute the probability that customer i is assigned to its v-th closest facility as:

$$\beta_{ii_v} = \begin{cases} 1 - p_{i_v}, & v = 1; \\ (1 - p_{i_v}) \times \prod_{l=1}^{v-1} p_{i_l}, & 2 \leq v \leq |J^+|; \end{cases} \tag{4.4}$$

The expected transpiration costs can be expressed as:

$$\sum_{i \in I} \sum_{v=1}^{|J^+|} c_{ii_v} d_i \beta_{ii_v} = \sum_{i \in I} c_{ii_1} d_i (1 - p_{i_1}) + \sum_{i \in I} \sum_{v=2}^{|J^+|} c_{ii_v} d_i (1 - p_{i_v}) \times \prod_{l=1}^{v-1} p_{i_v} \tag{4.5}$$

Thus, $F(z)$ is computed by solving the attacker's problem (AP):

$$F(z) = \max \sum_{i \in I} c_{ii_1} d_i (1 - p_{i_1}) + \sum_{i \in I} \sum_{v=2}^{|J^+|} c_{ii_v} d_i (1 - p_{i_v}) \times \prod_{l=1}^{v-1} p_{i_v} \tag{4.6}$$

$$\text{s.t.} \quad \sum_{j \in J} s_j \leq r \qquad \forall j \in J^+, \tag{4.7}$$

$$p_j = 1 - (1 - q_j)(1 - s_j w^{z_j}) \qquad \forall j \in J^+, \tag{4.8}$$

$$s_j \in \{0, 1\}, \qquad \forall j \in J^+.$$

In the bi-level model, the defender makes an upper-level decision, and the objective is to minimize expected transportation costs under worst-case attacks, which is given by solving the attacker's problem. Constraints (4.2) and (4.7) model the limited resources of the defender and attacker, respectively. The attacker's objective is to maximize the expected transportation costs after launching attacks.

The difficulty of the attacker's problem arises from the high degree of nonlinearity in its objective function and its constraints (4.8). Even for a special case such as the one studied by Zhu et al. [96] in which only disruptions caused by intentional attacks are considered, there is no solution method proposed other than enumeration. For the fortification problem, only a greedy heuristic method is proposed. An additional special case is one in which $w = 0$; that is, an attack fails for sure if a fortified facility is attacked. The problem is studied by Church et al. [19], Church and Scaparra [21], Scaparra and Church [76, 77], and Liberatore et al. [52]. In such a case, the attacker's problem can be formulated as a mixed integer programming model, and therefore a commercial solver such as CPLEX can be used to solve the problem. For the fortification problem with this attacker's problem as the lower level problem, there exist relatively efficient solution methods in the literature (e.g., [76, 77]).

4.2 Model Properties

In this section, we discuss the structure properties of the model from a set function point of view, which provides the theoretical cornerstone for the solution methods developed in the next section. Some preliminary knowledge of supermodular function is reviewed in Section 3.1.1, in Chapter 3. Same notations are used in this section.

4.2.1 The Attacker's Problem

Consider a known fortification decision, let $\Theta(S)$ be a function defined on a subset S of J^+ and $\Theta(S)$ is defined as the expected transportation costs when facilities in S are attacked. Next, we show Θ is supermodular.

Proposition 4.1. *Θ is nondecreasing.*

Proof: It is sufficient to show Θ is nonincreasing if $\Theta(S\cup\{e\}) \geq \Theta(S)$ for any $S \in J^+, e \notin S$. Let $\Gamma_i(S)$ be a set function corresponding to the expected transportation for customer i, thus $\Theta(S) = \sum_{i\in I} \Gamma_i(S)$, where $\Gamma_i(S) = \sum_{l=1}^{|J^+|} \beta_{ii_l} c_{ii_l}$. By lemma 3.2, it is sufficient to show $\Gamma_i(S)$ is nondecreasing for any i.

Without loss of generality, assume facility e is the k-th closest facility of the customer, and let p_e be the probability facility e fails when e is not attacked and p'_e be the probability when e is attacked. Obviously, $p'_e \geq p_e$, the equality holds when $w = 0$ and the facility is fortified. We have:

$$\Gamma_i(S \cup \{e\}) = \sum_{l=1}^{k-1} \beta_{ii_l} c_{ii_l} + \frac{1-p'_k}{1-p_k} \beta_{ii_k} c_{ii_k} + \frac{p'_k}{p_k} \sum_{l=k+1}^{|J^+|} \beta_{ii_l} c_{ii_l}.$$

Thus,

$$\begin{aligned}\rho_e^{\Gamma_i}(S) = \quad \Gamma_i(S \cup \{e\}) - \Gamma_i(S) &= \tfrac{p_k - p'_k}{1-p_k}\beta_{ii_k}c_{ii_k} + \tfrac{p'_k - p_k}{p_k}\textstyle\sum_{l=k+1}^{|J^+|}\beta_{ii_l}c_{ii_l} \\ &\geq c_{ii_k}\left(\tfrac{p_k - p'_k}{1-p_k}\beta_{ii_k} + \tfrac{p'_k - p_k}{p_k}\textstyle\sum_{l=k+1}^{|J^+|}\beta_{ii_l}\right) \\ &= c_{ii_k}\left(\tfrac{p_k - p'_k}{1-p_k}\beta_{ii_k} + \tfrac{p'_k - p_k}{p_k} \times \tfrac{p_k}{1-p_k} \times \beta_{ii_k}\right) \\ &= 0.\end{aligned}$$

The second line holds because by the definition of c_l, we have $c_{ii_1} \leq c_{ii_2} ... \leq c_{ii_{|J^+|}}$. Because there is an emergency facility, thus $p_{ii_{|J^+|}} = 0$ and $\beta_{ii_{|J^+|}} = \prod_{l=1}^{|J^+|-1} p_{i_l}$. With simple chaining process, we have:

$$\sum_{l=k+1}^{|J^+|} \beta_{ii_l} = \frac{p_k}{1-p_k}\beta_k. \tag{4.9}$$

This completes the proof.

Proposition 4.2. Θ *is supermodular.*

Proof: Consider another set T, $T = S \cup \{\sigma\}, \sigma \neq e, \sigma, e \notin S$. σ is the u-th closest facility. similarly, we have:

$$\Gamma_i(T) = \sum_{l=1}^{u-1} \beta_{ii_l}c_{ii_l} + \frac{1-p'_u}{1-p_u}\beta_{ii_u}c_{ii_u} + \frac{p'_u}{p_u}\sum_{l=u+1}^{|J^+|} \beta_{ii_l}c_{ii_l}.$$

Case 1, $u < k$:

$$\Gamma_i(T \cup \{e\}) = \sum_{l=1}^{u-1} \beta_{ii_l}c_{ii_l} + \frac{1-p'_u}{1-p_u}\beta_{ii_u}c_{ii_u} + \frac{p'_u}{p_u}\sum_{l=u+1}^{k-1} \beta_{ii_l}c_{ii_l} + \frac{p'_u}{p_u} \times \frac{p'_k}{p_k}\sum_{l=k+1}^{|J^+|} \beta_{ii_l}c_{ii_l}.$$

$$\rho_e^{\Gamma_i}(T) = \Gamma_i(T \cup \{e\}) - \Gamma_i(T) = \frac{p'_u}{p_u}(\frac{p_k - p'_k}{1 - p_k}\beta_{ii_k}c_{ii_k} + \frac{p'_k - p_k}{p_k}\sum_{l=k+1}^{|J^+|}\beta_{ii_l}c_{ii_l}).$$

Recall,

$$\rho_e^{\Gamma_i}(S) = \Gamma_i(S \cup \{e\}) - \Gamma_i(S) = \frac{p_k - p'_k}{1 - p_k}\beta_{ii_k}c_{ii_k} + \frac{p'_k - p_k}{p_k}\sum_{l=k+1}^{|J^+|}\beta_{ii_l}c_{ii_l}.$$

.

We have:

$$\rho_e^{\Gamma_i}(T) = \frac{p'_u}{p_u}\rho_e^{\Gamma_i}(S) \geq \rho_e^{\Gamma_i}(S).$$

.

Case 2, $u > k$:

$$\Gamma_i(T \cup \{e\}) = \sum_{l=1}^{k-1}\beta_{ii_l}c_{ii_l} + \frac{1 - p'_k}{1 - p_k}\beta_{ii_k}c_{ii_k} + \frac{p'_k}{p_k}\sum_{l=k+1}^{u-1}\beta_{ii_l}c_{ii_l} + \frac{p'_k}{p_k} \times \frac{1 - p'_u}{1 - p_u}\beta_{ii_u}c_{ii_u} + \frac{p'_k}{p_k} \times \frac{p'_u}{p_u} \times \sum_{l=u+1}^{|J^+|}\beta_{ii_l}c_{ii_l}.$$

$$\begin{aligned}\rho_e^{\Gamma_i}(T) &= \Gamma_i(T \cup \{e\}) - \Gamma_i(T) = \\ &= \tfrac{p_k - p'_k}{1 - p_k}\beta_{ii_k}c_{ii_k} + \tfrac{p'_k - p_k}{p_k}\textstyle\sum_{l=k+1}^{u-1}\beta_{ii_l}c_{ii_l} + \tfrac{p'_k - p_k}{p_k} \times \tfrac{1 - p'_u}{1 - p_u}\beta_{ii_u}c_{ii_u} + \tfrac{p'_k - p_k}{p_k} \times \tfrac{p_u}{p_u}\textstyle\sum_{l=u+1}^{|J^+|}\beta_{ii_l}c_{ii_l}.\end{aligned}$$

Thus,

$$\begin{aligned}
\rho_e^{\Gamma_i}(T) - \rho_e^{\Gamma_i}(S) &= \tfrac{p'_k - p_k}{p_k}\left(\tfrac{p_u - p'_u}{1-p_u}\beta_{ii_u}c_{ii_u} + \tfrac{p'_u - p_u}{p_u}\textstyle\sum_{l=u+1}^{|J^+|}\beta_{ii_l}c_{ii_l}\right) \\
&\geq \tfrac{p'_k - p_k}{p_k}\left(\tfrac{p_u - p'_u}{1-p_u}\beta_{ii_u}c_{ii_u} + c_{ii_u} \times \tfrac{p'_u - p_u}{p_u}\textstyle\sum_{l=u+1}^{|J^+|}\beta_{ii_l}\right) \\
&= \tfrac{p'_k - p_k}{p_k} \times c_{ii_u}\left(\tfrac{p_u - p'_u}{1-p_u}\beta_{ii_u} + \tfrac{p'_u - p_u}{p_u} \times \tfrac{p_u}{p'_u - p_u}\right) \\
&= 0,
\end{aligned}$$

In summary, $\rho_e^{\Gamma_i}(T) - \rho_e^{\Gamma_i}(S) \geq 0$ when T has exactly one more element than S.

Consider any sets $S \subseteq T \subset J^+$ and $e \in J^+ \setminus T$, let $V = T \setminus S = \{v_1, v_2, ..., v_n\}$.

We have

$$\begin{aligned}
\rho_e^{\Gamma_i}(S \cup \{v_1\}) - \rho_e^{\Gamma_i}(S) &\geq 0 \\
\rho_e^{\Gamma_i}(S \cup \{v_1, v_2\}) - \rho_e^{\Gamma_i}(S \cup \{v_1\}) &\geq 0 \\
&\vdots \\
\rho_e^{\Gamma_i}(T) - \rho_e^{\Gamma_i}((S \cup \{v_1, ..., v_n\}) &\geq 0.
\end{aligned}$$

Sum the inequalities, we have $\rho_e^{\Gamma_i}(T) \geq \rho_e^{\Gamma_i}(S)$. By second condition in Definition 3.1, we completes the proof.

4.2.2 The Fortification Problem

Define $\Psi(Z)$ as the corresponding set function for $F(z)$. Let $\mathbb{I}(Z)$ be the attacked facilities in an optimal solution of the attackers' problem when Z is the fortification set.

Proposition 4.3. *Ψ is nonincreasing.*

It is clear Ψ is nonincreasing.

Proposition 4.4. *$\rho_e^{\Psi}(Z) = 0$, if $e \notin \mathbb{I}(Z)$.*

Proof: Because $\mathbb{I}(Z)$ is a feasible solution for the attack problem, $\Psi(Z \cup \{e\}) \geq \Theta(\mathbb{I}(Z)) = \Psi(Z)$. Since Ψ is nonincreasing, $\Psi(Z \cup \{e\}) \leq \Psi(Z)$, thus $\Psi(Z \cup \{e\}) = \Psi(Z)$, i.e., $\rho_e(Z) = 0$.

The same observation is made by [76] for RIMF, and it immediately implies Proposition 4.5 and the tree search algorithm [76] is also applicable to solve this problem.

Proposition 4.5. *There exists an optimal solution Z^* where at least one j such that $j \in \mathbb{I}(\emptyset)$ and $j \in Z^*$.*

Proposition 4.6. *Ψ is neither supermodular nor submodular.*

Proof: We prove this by providing two examples. In the first example, we show a case where it is not supermodular. Then, we use another example to show it is not submodular.

Example 1: Consider an instance with one customer, three facilities, and an emergency facility as shown by Figure 4.2, i.e., $c_{ij} = j$, and $c_{i0} = 4$. Let $r = 3$, $w = 0.5$ and $q_j = 0$ for $j = 1, 2, 3$. Thus, if a facility is fortified, then the probability of fail $p_j = 0.5$, otherwise it fails for sure. Consider facility 2 as e, it is easy to compute Table 4.1.

Table 4.1: Function values with different T for example 1

	$T = \emptyset$	$\{1\}$	$\{1, 3\}$
$\Psi(T)$	4	2.5	2.25
$\Psi(\{e\} \cup T)$	3	2	1.875
$\rho_e^{\Psi}(T)$	-1	-0.5	-0.375

Because $\rho_e^{\Psi}(\emptyset) < \rho_e^{\Psi}(\{1\}) < \rho_e^{\Psi}(\{1, 3\})$, Ψ violates the definition of a supermodular function.

Example 2: Consider an instance with the same structure as example 1. However, we assume there is only one attack unit, i.e., $r = 1$. Let $w = 0$ and $q_j = 0.5$ for $j = 1, 2, 3$. In

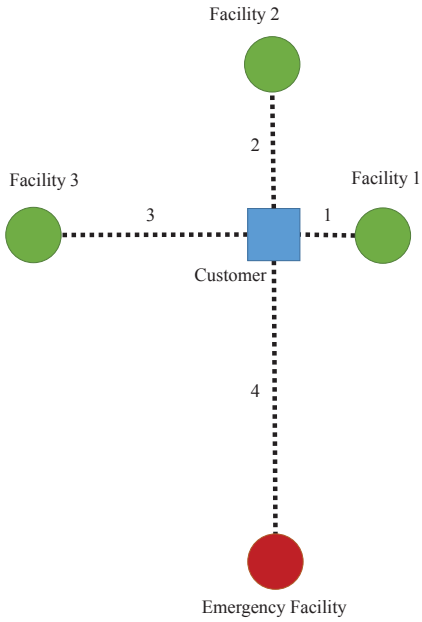

Figure 4.2: A instance of fortification problem

this case, if a facility is fortified, it is immune to the attack, while it fails at probability of 0.5 even no attack is lunched of that facility. For this case, we have following Table 4.2.

Table 4.2: Function values with different T for example 2

	$T = \emptyset$	$\{1\}$	$\{1, 3\}$
$\Psi(T)$	2.75	2.25	2.25
$\Psi(\{e\} \cup T)$	2.75	2	1.875
$\rho_e^{\Psi}(T)$	0	-0.25	-0.375

Because $\rho_e^{\Psi}(\emptyset) > \rho_e^{\Psi}(\{1\}) > \rho_e^{\Psi}(\{1, 3\})$, Ψ violates the definition of a submodular function.

Proposition 4.7. *When attack solution is fixed, let $\bar{\Psi}$ be the function for the corresponding fortification problem. $\bar{\Psi}$ is a nonincreasing supermodular function.*

Proof: Let $p_j^{\uparrow}$ be the probability of disruption for facility j when j is not fortified, and $p_j^{\downarrow}$ be the the probability when it is fortified. When we prove Proposition 4.2, the result of attacking a facility is equivalent to increase the probability of disruption for that facility. Correspondingly, fortifying a facility is equivalent to decrease the probability. Thus, we can construct nondecreasing supermodular function $\Theta(S)$ such that a facility fails with probability $p_j^{\uparrow}$ for $j \in S$, and $p_j^{\downarrow}$ for $j \notin S$. Let $S^{\complement}$ be the complement set of S. $\bar{\Psi}(S) = \Theta(S^{\complement})$. Because Θ is a nondecreasing supermodular, $\bar{\Psi}$ is nonincreasing supermodular.

4.3 Solution Methods

4.3.1 For the Attacker's Problem

We have shown that the AP is equivalent to maximizing a nondecreasing supermodular function with cardinality constraints. Recall that $-f$ is submodular if f is supermodular. The problem is also equivalent to minimizing a nonincreasing submodular function with cardinality constraints. Because a supermodular function has its equivalent submodular counterpart, in the literature, the term submodular is commonly used instead of supermodular to maintain consistency in terminology.

The relationship can be summarized with the following four-way contingency Table 4.3. The diagonal elements are equivalent.

Table 4.3: Relationship of submodular and supermodular optimization

	Objective	
Function	Minimize	Maximize
Submodular	Polynomial solvable	NP-hard
Supermodular	NP-hard	Polynomial solvable

Minimizing a general submodular function can be solved in strong polynomial time [44, 69]. The most theoretically efficient algorithm known is the one developed by [69], which runs in $O(n^5 EO + n^6)$, where n is the size of the ground set and EO is the time to evaluate the function with a given subset. However, methods for incorporating additional constraints, such as cardinality constraints or, more generally, knapsack constraints, are not discussed. In fact, the problem may become significantly harder when side constraints are added. There are cases of submodular function minimization with side constraints that are NP-hard McCormick [58]. Some research considers knapsack constraints in the context of submodular optimization; however, most of this research relates to maximizing a submodular function [87, 49, 45, 47]. To the best of our knowledge, there is no combinatorial method developed for minimizing a monotone, nonincreasing, submodular function with cardinality constraints or knapsack constraints.

Next we outline a cutting plane algorithm that solves the AP exactly. A similar approach has been developed for solving supply chain design problem where a cost term appears to be submodular [92]. The cutting plane algorithm iteratively solves a series of MIP master problems and subproblems. Because the solution methods uses a MIP master problem, it can easily include additional side constraints, which can be useful in practice.

To be consistent with terminology used in the literature, instead of maximizing a supermodular function, we reformulate the problem as a submodular function minimization. Define set function $\Omega(S) = -\Theta(S) + \Theta(\emptyset)$, such that Ω is a nonincreasing submodular function with $\Omega(\emptyset) = 0$. The AP is equivalent to

$$\min\{\Omega(S) : |S| \leq r, S \subseteq J\}.$$

Definition 4.8. *Given a submodular function f with $f(\emptyset) = 0$, the submodular polyhedron $P(f)$ is defined as $P(f) = \{v : v \in \mathbb{R}^n, \forall S \subseteq U, v(S) \leq f(S)\}$, where $v(S) = \sum_{i \in S} v_i$.*

With a slight abuse of notation, we define $f(s) = f(S)$, where s is the incidence vector for the set S.

The convex lower envelope for the submodular function is:

$$Q_f = \{(s,t), \in \{0,1\}^n \times \mathbb{R} : f(s) \leq t\}$$

.

Theorem 4.9. *The inequality $t \geq \sum_{i \in U} v_i s_i$ is a valid inequality for Q_f if and only if $v \in P(f)$. AtamtüRk and Narayanan [7]*

The inequality defined above is called an extended polymatroid inequality. Given a solution $(\bar{s}, \bar{t})$, we can check whether in Q_f or we find a an extended polymatroid inequality to cut off the infeasible solution. The problem is written as:

$$t^* = \max\{\sum_{i \in U} v_i \bar{s}_i : v \in P(f)\},$$

and can be solved with the Greedy Algorithm [27]:

Step 1 For a given s, sort U such that $s_{l_1} \geq s_{l_2} \geq .. \geq s_{l_n}$

Step 2 Compute $\bar{v_i}$ as $f(\{l_1, l_2, .., l_i\})$ - $f(\{l_1, l_2, .., l_{i-1}\})$, which is the incremental value by adding l_i to $\{l_1, l_2, .., l_{i-1}\}$.

Because s_i only takes value in 0 and 1, Step 1 runs in $O(n)$, and Step 2 can be done with n sequential calls to function evaluation. Thus, the algorithm runs with a time bound $O(nEO)$, and we have $\sum_{i \in U} \bar{v}_i \bar{s}_i = f(S)$. If $\bar{t} \geq t^*$, no extended polymatroid inequality is violated, or we find a cut in the following form:

$$t \geq \bar{v_1} s_1 + \bar{v_2} s_2 + ... + \bar{v_n} s_n.$$

The master problem (AP-master) is formulated as:

$$\min t \tag{4.10}$$

$$\begin{aligned} \text{s.t.} \quad & t \geq \bar{v_1}^k s_1 + \bar{v_2}^k s_2 + ... + \bar{v_n}^k s_n && \forall k \in K, \\ & \textstyle\sum_{j \in J^+} s_j \leq r && \forall j \in J^+, \\ & s_j \in \{0, 1\}, && \forall j \in J^+, \end{aligned} \tag{4.11}$$

where K is the collection of cuts from previous iterations. The cutting plane algorithm starts by solving AP-master with constraints (4.11) being empty. Let $(\bar{\mathbf{s}}, \bar{t})$ be the optimal solution to the master problem. Note that the problem can be unbounded in the first iteration, in this case $t = -\infty$. Add a cut if necessary with the Greedy Algorithm as outlined above. The process repeat until a solution does not violate any extended polymatroid inequality. The algorithm then terminates with an optimal solution to the problem.

4.3.2 For the Fortification Problem

Compared with the AP, the FP is much harder for two reasons. First, unlike AP in which the objective value can be directly computed in polynomial time with a given solution, to evaluate a solution for FP one needs to solve a corresponding AP. Second, because we have shown that FP is neither submodular nor supermodular, there is not much structural information we can use. We next develop a logic-based Benders decomposition algorithm for solving the problem.

A logic-based Benders decomposition

The logic-based Benders decomposition introduced by [42] is a generalization of the Benders decomposition. Like Benders decomposition, the logic-based Benders decomposition decomposes the original problem into a master problem and (a) subproblem(s) with corresponding variables denoted as x and y. The master problem is solved to obtain a solution for variables x, then subproblems are solved for y given the fixed x values.

The cuts in the Benders decomposition are based on linear duality, which requires the subproblem to be a linear programming problem whereas the subproblem of a logic-based Benders decomposition can be in any form of a mathematical program. The *inference duals* are used instead of the linear duals. Because of this, there is no standard algorithm to generate cuts for the logic-based Benders decomposition and one has to derive cuts based on knowledge of the underlying problem. The cutting plane methods we propose in the previous section are actually a special case of logic Benders decompositions where the *inference duals* is readily provided by the supermodular property.

The efficiency of a logic-based Bender decomposition depends on how the master problem and subproblem are defined and is highly affected by the tightness of the cut. We can write the master problem of FP as:

$$\min \eta \tag{4.12}$$

$$\begin{aligned} \text{s.t.} \quad & \text{logic-based Benders cuts} && \\ & \textstyle\sum_{j \in J^+} z_j \leq h && \forall j \in J^+ \\ & z_j \in \{0,1\}, && \forall j \in J^+ \end{aligned} \tag{4.13}$$

where, η is a nonnegative continuous variable to simulate transportation costs. The relationship between η and decision variables x and z is established by the logic-based Benders cuts.

Given a solution $(\bar{z}, \bar{\eta})$ to the master problem, we compute the transportation costs by solving an AP, and let $\eta^* = F(\bar{z})$. If $\eta^* > \bar{\eta}$, then the current solution is infeasible and a cut that eliminates such a solution should be added to the master problem.

With the assumption that the master problem has a finite domain, Chu and Xia [17] shows that the algorithm is guaranteed to converge to an optimal solution if the cuts satisfy the following two conditions:

1 If the current master problem solution x is infeasible, then the cut must exclude at least x;

2 Any feasible solution x satisfies the cut.

Clearly the result holds for our problem even though it has a continuous variable η because the model always tries to minimize η. A cut that satisfies both conditions is called valid.

Theorem 4.10. *Cuts in the form of (4.14) guarantee that the algorithm finds an optimal solution in finite iterations.*

$$\eta \geq \eta^* - \eta^*(\sum_{j\in\bar{z}_j=0} z_j + \sum_{j\in\bar{z}_j=1}(1-z_j)) \tag{4.14}$$

Proof: Clearly, the cut satisfies for condition 1, for if incumbent solution $(\bar{z}, \bar{\eta})$ is infeasible, i.e., $\bar{\eta} < \eta^*$, the cut ensures the solution is cut off by requiring $\eta \geq \eta^*$ for solution $(\bar{z})$. For any solution other than $(\bar{z})$, the cut is inactive. Thus, it does not eliminate any feasible solution.

Obviously, the algorithm based on cut (4.14) is inefficient since it does no better than simple enumeration of all of the possible combinations of z, i.e., all subsets of J. Now consider a family of cuts in the following form:

$$\eta \geq \eta^* + \sum_{j\in\bar{z}_j=0} \Delta_j^z z_j + \sum_{j\in\bar{z}_j=1} \nabla_j^z (1-z_j) \tag{4.15}$$

, where Δ_j^z is a coefficient used to capture the change of transportation costs by fortifying facility j and ∇_j^z is used to capture the change of transportation costs when the defense resource is removed from facility j. Clearly, $\Delta_j^z \leq 0$ and $\nabla_j^z \geq 0$. When set $\Delta_j^z = -\eta^*$ for $j, \bar{z}_j = 0$ and $\nabla_j^z = 0$ for $j, \bar{z}_j = 1$, it is easy to verify it is a valid cut and is at least as tight as (4.14). For simplicity, we let $\nabla_j^z = 0$ and focus on analyzing Δ_j^z.

$$\eta \geq \eta^* + \sum_{j \in \bar{z}_j = 0} \Delta_j^z z_j \tag{4.16}$$

Let Z denote the corresponding fortified locations for the incumbent solution $\bar{z}$.

Theorem 4.11. *Cuts in the form of* (4.16) *with* $\Delta_j^z = \widehat{\Delta_j^z}$ *are valid.*

$$\widehat{\Delta_j^z} = \min_{T, Z \subseteq T \subseteq J,} \{\Psi(T \cup \{j\}) - \Psi(T)\} \text{ for } j \in J, j \notin Z \tag{4.17}$$

Proof: Clearly the cut eliminates infeasible incumbent solutions. We next show that it does not eliminate any feasible solution.

Consider a cut generated with an incumbent solution Z. For another solution Z', $Z \subseteq Z'$. Let $Z_\Delta = Z' \setminus Z$. Consider an arbitrary order of $Z_\Delta, \{l_1, l_2, .., l_n\}$. Let $L^i = Z \cup \{l_1, l_2, ..., l_i\}$. We have

$$\Psi(Z') = \Psi(Z) + \sum_{i=1}^{n} \rho_{l_i}^{\Psi}(L^{i-1})$$

By the definition of $\widehat{\Delta_j^z}$, we have $\widehat{\Delta_{l_i}^z} \leq \rho_{l_i}^{\Psi}(L^{i-1})$

$$\Psi(Z') \geq \eta^* + \sum_{j \in \bar{z}_j = 0} \hat{\Delta}_j^z z_j$$

Thus, cut (4.17) does not eliminate solution Z' for any $Z \subsetneq Z'$. Because Ψ is a nonincreasing function regarding both set variables, if $Z \not\subseteq Z'$, the inequality still holds by adding missing elements in Z to Z'. Thus, the cut does not eliminate any feasible solution.

However, computing $\widehat{\Delta_j^z}$ is hard in general. Next, we propose a family of cuts that can be computed in no more than a linear number of calls to the AP. The basic idea is to find an approximation to the lower bound $\widehat{\Delta_j^z}$.

Because of Proposition 4.4, an optimal solution T to (4.17) must satisfy $j \in \mathbb{I}(T)$. The motivation for the following cut by forcing j to be included by $\mathbb{I}(Z)$.

Define the restricted attack problem $F_j^{\Diamond}(z)$ by forcing facility j to be attacked. This can be achieved by adding the constraint in the master model of AP:

$$s_j \geq 1. \tag{4.18}$$

Let Ψ_j denote the set function for the corresponding fortification problem. We always have $j \in \mathbb{I}(T)$ for $j \notin T$.

$$\widetilde{\Delta_j^z} = \Psi_j(Z \cup \{j\}) - \Psi_j(Z) \text{ for } j \in J, j \notin Z \tag{4.19}$$

It is still an open question whether cuts with coefficient $\widetilde{\Delta_j^z}$ are valid cut or not for the problem. To compute a coefficient, we need to solve two APs. Because we only want to find a lower bound approximation for the coefficient, we can first compute $\Psi_j(Z)$, and find the corresponding interdiction set $\mathbb{I}(Z)$, and compute a lower bound of $\Psi_j(Z \cup \{j\})$ by fixing the interdiction solution to $\mathbb{I}(Z)$.

4.4 Computational Study

In this section, we test the performance of the proposed algorithms. We implement the algorithms in Scala and solve all of the problem instances on a Dell OptiPlex 9010 with one Intel 3.40 GHz CPU and 4G of memory running the Ubuntu operating system. The MIP problems are solved using the ILOG CPLEX Academic Initiative Edition 12.6. The U.S. dataset is used to test the algorithms' performance. The U.S. dataset is based on 2000 census data and contains the largest cities in the U.S. and is used in related works [1, 24, 80]. The exogenous probabilistic disruption rate is computed as $p_j = \alpha e^{-D_j/\theta}$, where D_j is the distance from the location to New Orleans.

4.4.1 Solving the Attacker's Problem

Because there is no solution method other than enumeration available in the literature, we compare the cutting plane algorithm with enumeration. The algorithms are tested with instances ranging from 50 nodes to 150 nodes. In the dataset, a node is a customer and a candidate location. Thus, we first solve a p-median problem to select p facilities. We assume that no facility is fortified, and we test three values for the number of facilities attacked r, i.e., 3, 6, and 9, which covers the range of r values used in most related works. The results are summarized in Table 4.4. Columns tilted "nbEO" reports the number of calls to evaluate a solution (z, s), i.e, to compute objective function (4.6). For the enumeration method, the number of calls equals the combination number $\binom{p}{r}$. It is not surprising that the cutting plane algorithm always finds an optimal solution. While the solution time for enumeration

increases exponentially with respect to p and r, the increase is very mild for the cutting plane algorithm.

We also test the algorithm's performance by varying parameters that affect the disruption probabilities, such as w, α, θ. Three combinations of α and θ are tested to simulate low, moderate, and high probabilistic disruption risks: $(0.1, 200)$, $(0.2, 400)$, and $(0.3, 800)$ respectively. We randomly fortify $\lfloor \frac{p}{2} \rfloor$ facilities. The results are reported in Table 4.5- 4.8. Columns tilted "nbCuts" report the number of cuts in the form of (4.11) is added to the master problem. From the results, we observe that when w increases, the objective value increases, and the solution time and the number of cuts generated increases in general. Recall that w is the probability that an attack succeeds when a facility is fortified. The objective value increases with the probabilistic disruption risk factors α and θ. However, no obvious pattern is observed in terms of how the these factors affect the solution time and the number of cuts generated.

Table 4.4: Cutting plane algorithm's performance on solving AP

Nodes	p	r	Enumeration			Cutting plane		
			Obj.	Time	nbEO	$\Delta_{Obj.}$	Time	nbEO
50	15	3	1101845.24	0.14	455	0	0.29	374
50	15	6	1976813.06	0.55	5005	0	0.11	740
50	15	9	3240988.49	0.42	5005	0	0.11	969
50	20	3	792317.18	0.07	1140	0	0.07	552
50	20	6	1455117.90	1.70	38760	0	0.14	1503
50	20	9	2202803.09	7.44	167960	0	0.20	2043
50	30	3	431071.16	0.29	4060	0	0.19	1774
50	30	6	845343.21	41.89	593775	0	0.48	4224
50	30	9	1423410.35	1012.98	14307150	0	0.66	5002
75	15	3	1265758.87	0.03	455	0	0.03	339
75	15	6	2303228.54	0.24	5005	0	0.06	805
75	15	9	3246383.92	0.24	5005	0	0.09	966
75	20	3	897856.49	0.08	1140	0	0.06	656
75	20	6	1732421.59	2.53	38760	0	0.15	1665
75	20	9	2355708.23	11.04	167960	0	0.25	2109
75	30	3	514184.54	0.40	4060	0	0.25	2177
75	30	6	1132851.13	59.78	593775	0	0.49	3700
75	30	9	1707084.16	1443.23	14307150	0	1.18	7727
100	15	3	1372013.35	0.03	455	0	0.05	548
100	15	6	2502580.27	0.30	5005	0	0.12	1272
100	15	9	3534156.55	0.30	5005	0	0.13	1245
100	20	3	1002426.94	0.10	1140	0	0.09	825
100	20	6	1918525.84	3.30	38760	0	0.22	1942
100	20	9	2558151.56	14.32	167960	0	0.34	2198
100	30	3	593566.69	0.51	4060	0	0.33	2273
100	30	6	1257326.92	75.66	593775	0	0.65	4194
100	30	9	1914434.62	1825.49	14307150	0	1.25	6927
150	15	3	1546415.88	0.05	455	0	0.05	372
150	15	6	2698907.14	0.50	5005	0	0.14	1079
150	15	9	3977272.72	0.50	5005	0	0.18	1254
150	20	3	1187705.98	0.15	1140	0	0.14	762
150	20	6	2234857.77	5.01	38760	0	0.23	1454
150	20	9	2887852.42	21.86	167960	0	0.66	3559
150	30	3	697791.79	0.75	4060	0	0.55	2672
150	30	6	1465985.90	109.53	593775	0	0.87	3913
150	30	9	2219730.26	2670.02	14307150	0	1.67	7013

Table 4.5: Algorithm's performance on different parameters – instances with 50 nodes

Nodes	p	r	α	θ	$w = 0.4$				$w = 0.6$				$w = 0.8$			
					Obj.	Time	nbCuts	nbEO	Obj.	Time	nbCuts	nbEO	Obj.	Time	nbCuts	nbEO
50	15	3	0.1	200	646491.81	0.21	11	180	792218.03	0.10	14	231	937944.26	0.08	18	294
50	15	3	0.2	400	668362.21	0.06	14	227	812856.55	0.05	18	292	957350.90	0.06	28	454
50	15	3	0.3	800	750129.29	0.03	11	180	887448.01	0.03	15	242	1024766.74	0.07	31	500
50	15	6	0.1	200	964558.04	0.04	16	259	1216939.34	0.06	28	455	1544510.95	0.06	28	456
50	15	6	0.2	400	991650.28	0.04	18	290	1247395.32	0.07	37	595	1575842.61	0.09	35	567
50	15	6	0.3	800	1084601.75	0.03	20	325	1346233.23	0.06	38	616	1673069.58	0.08	51	822
50	15	9	0.1	200	1223619.83	0.03	12	198	1551609.28	0.07	32	518	2186703.53	0.05	25	407
50	15	9	0.2	400	1254648.24	0.03	15	245	1577136.36	0.10	45	729	2235828.91	0.09	37	601
50	15	9	0.3	800	1364131.70	0.04	17	278	1717114.39	0.14	56	908	2373736.64	0.10	68	1096
50	20	3	0.1	200	467394.61	0.02	6	127	568948.23	0.03	12	257	670501.84	0.02	14	298
50	20	3	0.2	400	487809.61	0.01	6	127	589312.13	0.02	12	256	690814.66	0.03	19	404
50	20	3	0.3	800	557381.67	0.02	8	169	655578.85	0.04	21	446	753776.03	0.05	30	634
50	20	6	0.1	200	649288.37	0.03	20	425	854882.36	0.04	20	427	1116002.03	0.05	29	615
50	20	6	0.2	400	670232.07	0.04	22	469	879303.35	0.04	24	510	1142008.75	0.09	50	1057
50	20	6	0.3	800	748289.64	0.02	16	342	961221.39	0.07	46	973	1218942.84	0.09	64	1354
50	20	9	0.1	200	816174.32	0.02	14	298	1139635.64	0.05	35	740	1528079.96	0.07	39	825
50	20	9	0.2	400	841676.76	0.03	20	425	1166018.29	0.04	24	509	1555371.82	0.10	49	1036
50	20	9	0.3	800	929345.70	0.04	26	553	1251554.55	0.07	42	889	1645403.09	0.21	119	2514
50	30	3	0.1	200	197745.54	0.05	12	375	234906.98	0.09	22	687	307495.06	0.09	33	1029
50	30	3	0.2	400	214197.71	0.04	13	406	251285.12	0.06	22	686	323823.76	0.13	48	1495
50	30	3	0.3	800	266417.92	0.04	14	436	303305.38	0.05	19	593	375245.69	0.14	50	1557
50	30	6	0.1	200	366943.59	0.05	17	532	466957.56	0.08	27	842	580947.81	0.18	60	1865
50	30	6	0.2	400	385218.63	0.05	17	531	485931.33	0.08	27	844	600128.88	0.19	62	1927
50	30	6	0.3	800	444770.54	0.07	20	627	545229.68	0.08	29	904	656731.60	0.21	68	2114
50	30	9	0.1	200	458843.48	0.07	25	780	644215.57	0.15	52	1619	878623.87	0.23	69	2148
50	30	9	0.2	400	477460.29	0.14	48	1495	662224.00	0.11	37	1151	895736.85	0.38	116	3605
50	30	9	0.3	800	540754.61	0.11	40	1244	722042.89	0.14	45	1399	948530.98	0.39	112	3478

Table 4.6: Algorithm's performance on different parameters – instances with 75 nodes

Nodes	p	r	α	θ	$w = 0.4$				$w = 0.6$				$w = 0.8$			
					Obj.	Time	nbCuts	nbEO	Obj.	Time	nbCuts	nbEO	Obj.	Time	nbCuts	nbEO
75	15	3	0.1	200	841154.86	0.02	13	210	934062.69	0.02	17	279	1083869.38	0.03	28	452
75	15	3	0.2	400	865100.88	0.02	14	226	958201.38	0.02	20	326	1109851.56	0.03	25	407
75	15	3	0.3	800	943327.40	0.02	14	229	1045096.59	0.03	23	375	1193996.58	0.04	32	518
75	15	6	0.1	200	1279956.32	0.02	11	179	1518375.05	0.03	21	341	1828591.49	0.05	40	645
75	15	6	0.2	400	1315230.17	0.02	11	179	1553551.90	0.03	26	425	1865672.19	0.05	39	630
75	15	6	0.3	800	1426897.78	0.02	16	260	1660920.48	0.03	25	407	1966862.85	0.05	41	664
75	15	9	0.1	200	1644592.67	0.02	17	276	2032639.17	0.03	23	373	2502527.64	0.04	30	486
75	15	9	0.2	400	1686103.69	0.02	16	262	2070857.62	0.04	31	506	2540367.20	0.08	55	889
75	15	9	0.3	800	1801648.37	0.02	19	311	2183489.78	0.05	33	539	2643466.74	0.11	75	1213
75	20	3	0.1	200	543153.23	0.02	8	170	650409.41	0.03	12	257	757665.59	0.04	18	384
75	20	3	0.2	400	567202.54	0.02	7	148	673835.08	0.03	14	296	780467.61	0.05	26	554
75	20	3	0.3	800	647595.74	0.02	10	213	749665.85	0.03	17	361	851735.95	0.05	27	570
75	20	6	0.1	200	753486.52	0.04	21	445	975119.92	0.07	32	682	1290260.77	0.07	32	680
75	20	6	0.2	400	780615.24	0.04	17	362	1004593.15	0.08	32	678	1318672.77	0.15	76	1605
75	20	6	0.3	800	868683.57	0.04	21	443	1096677.77	0.09	44	932	1401777.70	0.12	51	1078
75	20	9	0.1	200	946363.40	0.05	24	509	1295043.67	0.09	44	932	1749010.59	0.12	56	1189
75	20	9	0.2	400	973857.52	0.06	27	578	1325374.67	0.09	38	804	1778576.55	0.13	58	1226
75	20	9	0.3	800	1070560.97	0.09	48	1018	1421409.38	0.10	46	972	1865410.36	0.18	78	1646
75	30	3	0.1	200	248626.69	0.13	33	1030	289686.58	0.22	57	1775	379864.07	0.21	55	1711
75	30	3	0.2	400	266482.03	0.15	39	1219	307752.06	0.17	44	1370	398293.52	0.15	39	1214
75	30	3	0.3	800	320628.93	0.13	34	1060	365406.71	0.18	47	1461	457970.34	0.24	64	1993
75	30	6	0.1	200	370752.02	0.10	25	779	482767.05	0.15	36	1123	724110.45	0.25	58	1807
75	30	6	0.2	400	390536.39	0.15	39	1217	501293.43	0.22	56	1747	743765.98	0.26	64	1995
75	30	6	0.3	800	454649.26	0.13	33	1029	562133.23	0.15	35	1093	808828.24	0.42	103	3206
75	30	9	0.1	200	500797.23	0.12	29	905	638355.71	0.35	79	2459	1001377.41	0.39	83	2584
75	30	9	0.2	400	520446.55	0.11	27	843	657530.48	0.33	73	2275	1020613.97	0.44	95	2954
75	30	9	0.3	800	587472.52	0.13	32	1001	720537.44	0.45	89	2770	1082735.19	0.55	113	3513

Table 4.7: Algorithm's performance on different parameters – instances with 100 nodes

Nodes	p	r	α	θ	$w = 0.4$				$w = 0.6$				$w = 0.8$			
					Obj.	Time	nbCuts	nbEO	Obj.	Time	nbCuts	nbEO	Obj.	Time	nbCuts	nbEO
100	15	3	0.1	200	851938.73	0.02	10	163	1016052.90	0.03	21	342	1180167.06	0.04	29	468
100	15	3	0.2	400	883125.84	0.02	15	243	1046088.34	0.02	14	228	1209050.85	0.04	26	423
100	15	3	0.3	800	990082.76	0.02	13	209	1145873.29	0.03	18	292	1301663.82	0.03	23	373
100	15	6	0.1	200	1207367.79	0.04	27	436	1445538.08	0.04	27	437	1895994.80	0.07	47	756
100	15	6	0.2	400	1232800.09	0.03	19	309	1485930.75	0.06	42	678	1936730.71	0.07	46	742
100	15	6	0.3	800	1332409.80	0.04	20	324	1608562.79	0.08	52	839	2050312.24	0.12	84	1354
100	15	9	0.1	200	1496908.67	0.02	14	228	1883939.12	0.07	42	675	2499328.26	0.08	48	775
100	15	9	0.2	400	1535290.62	0.03	21	341	1932646.31	0.07	45	722	2540767.89	0.13	68	1101
100	15	9	0.3	800	1671863.58	0.03	23	374	2087906.66	0.09	54	872	2673027.83	0.14	82	1321
100	20	3	0.1	200	609454.96	0.02	6	127	720915.31	0.04	15	320	832375.65	0.07	25	533
100	20	3	0.2	400	635698.99	0.02	6	127	746529.58	0.04	16	344	857360.18	0.07	28	596
100	20	3	0.3	800	722272.07	0.02	9	190	828418.26	0.04	15	321	934564.45	0.08	30	636
100	20	6	0.1	200	842712.64	0.06	24	510	1071876.96	0.11	43	913	1406135.68	0.14	48	1016
100	20	6	0.2	400	870520.87	0.05	21	447	1103987.03	0.10	41	870	1437096.25	0.19	80	1690
100	20	6	0.3	800	960711.79	0.06	24	510	1203369.67	0.09	36	765	1526960.28	0.20	81	1711
100	20	9	0.1	200	1078848.11	0.04	15	319	1412434.51	0.15	55	1164	1898835.73	0.18	60	1270
100	20	9	0.2	400	1109366.44	0.03	12	257	1445471.09	0.11	41	866	1931032.68	0.20	67	1417
100	20	9	0.3	800	1209879.67	0.04	17	361	1549416.04	0.08	29	613	2025199.92	0.24	80	1690
100	30	3	0.1	200	344839.05	0.09	18	568	371196.29	0.19	40	1246	444896.69	0.22	48	1493
100	30	3	0.2	400	363700.45	0.12	26	813	390420.60	0.14	30	936	464865.86	0.27	58	1807
100	30	3	0.3	800	419218.05	0.11	21	659	449422.09	0.14	29	903	529538.06	0.29	63	1960
100	30	6	0.1	200	458796.66	0.24	44	1374	605081.02	0.15	31	967	856012.39	0.26	52	1621
100	30	6	0.2	400	477548.54	0.27	54	1680	625445.13	0.17	35	1091	877726.26	0.42	84	2614
100	30	6	0.3	800	539078.99	0.23	47	1465	692520.93	0.26	51	1588	949886.66	0.57	114	3542
100	30	9	0.1	200	605321.88	0.15	30	939	779989.27	0.26	48	1495	1185328.34	0.39	77	2396
100	30	9	0.2	400	627581.86	0.17	34	1061	800507.95	0.35	67	2086	1206046.11	0.38	71	2212
100	30	9	0.3	800	699988.30	0.19	37	1153	868913.17	0.46	79	2458	1273243.94	0.56	105	3267

Table 4.8: Algorithm's performance on different parameters – instances with 150 nodes

Nodes	p	r	α	θ	$w = 0.4$				$w = 0.6$				$w = 0.8$			
					Obj.	Time	nbCuts	nbEO	Obj.	Time	nbCuts	nbEO	Obj.	Time	nbCuts	nbEO
150	15	3	0.1	200	978091.13	0.02	10	163	1157034.39	0.03	13	214	1335977.64	0.05	24	391
150	15	3	0.2	400	1013516.61	0.03	14	227	1191149.70	0.05	24	393	1368782.79	0.05	25	405
150	15	3	0.3	800	1133839.75	0.05	22	354	1303359.69	0.05	23	375	1472879.62	0.06	31	500
150	15	6	0.1	200	1319742.13	0.08	36	582	1647836.05	0.09	42	681	2057133.77	0.11	48	775
150	15	6	0.2	400	1349494.47	0.08	37	599	1682488.02	0.10	46	742	2103901.45	0.14	63	1016
150	15	6	0.3	800	1464954.11	0.09	39	630	1799061.45	0.12	52	840	2231052.10	0.17	79	1270
150	15	9	0.1	200	1671972.32	0.04	19	307	2113443.08	0.12	51	821	2804872.55	0.14	54	871
150	15	9	0.2	400	1708514.78	0.04	20	322	2156616.49	0.12	52	836	2852898.90	0.15	65	1047
150	15	9	0.3	800	1848990.67	0.08	36	583	2292580.04	0.18	73	1175	3004224.55	0.28	103	1657
150	20	3	0.1	200	694426.32	0.04	11	234	834210.25	0.05	13	275	975199.10	0.09	25	530
150	20	3	0.2	400	722874.51	0.05	12	254	863337.46	0.20	15	319	1003800.41	0.11	31	656
150	20	3	0.3	800	817204.06	0.04	11	232	953210.56	0.06	19	403	1089217.07	0.11	32	676
150	20	6	0.1	200	967912.36	0.04	12	257	1235387.61	0.10	29	611	1541933.86	0.20	53	1120
150	20	6	0.2	400	998526.51	0.05	15	320	1266038.70	0.11	31	658	1572660.68	0.34	90	1901
150	20	6	0.3	800	1098849.57	0.05	13	279	1363699.07	0.09	27	570	1666501.60	0.26	73	1538
150	20	9	0.1	200	1180653.86	0.06	17	362	1508890.43	0.18	46	968	2048708.93	0.29	68	1436
150	20	9	0.2	400	1205621.14	0.06	17	360	1536997.97	0.18	48	1015	2079084.15	0.35	94	1984
150	20	9	0.3	800	1294461.72	0.08	22	466	1632219.54	0.24	60	1268	2174244.26	0.39	99	2089
150	30	3	0.1	200	414869.50	0.12	18	562	438490.91	0.31	45	1400	530829.62	0.36	54	1680
150	30	3	0.2	400	436341.13	0.12	17	532	459956.72	0.25	37	1153	553688.50	0.36	53	1651
150	30	3	0.3	800	498675.63	0.15	21	657	523762.45	0.26	39	1214	627217.23	0.38	57	1774
150	30	6	0.1	200	544925.55	0.24	34	1060	712363.14	0.22	32	999	1000880.61	0.37	53	1655
150	30	6	0.2	400	566447.44	0.25	36	1123	735667.92	0.27	39	1217	1025812.40	0.43	62	1933
150	30	6	0.3	800	636080.11	0.29	42	1309	811999.41	0.25	37	1153	1108433.96	0.55	80	2486
150	30	9	0.1	200	675773.69	0.27	36	1123	903661.83	0.47	62	1930	1370628.88	0.49	63	1965
150	30	9	0.2	400	699827.98	0.29	40	1249	927251.44	0.46	62	1931	1394749.76	0.76	103	3203
150	30	9	0.3	800	774115.34	0.30	41	1276	1005210.48	0.63	84	2611	1472280.18	1.01	135	4195

4.4.2 Solving the Fortification Problem

Two methods in the literature can be adapted to solve FP: the tree-search implicit enumeration method proposed by Scaparra and Church [76], which is an exact algorithm to solve FP; and the greedy search heuristic algorithm proposed by Zhu et al. [96]. The proposed logic-based Benders decomposition algorithm is compared with these two algorithms, and the results are reported in Table 4.9- 4.12. The column titled "nbAPs" reports the number of calls required to solve an AP. All of the APs are solved by the cutting plane method. The column titled "$\Delta_{Obj.}$" reports the difference between the objective value obtain by the relevant algorithm and the objective value from the tree search algorithm. The algorithms are set with a time limit of 3600 seconds. Because the tree search may need to explore $(r^{h+1}-1)/(r-1)$ number of nodes [76], as expected, we observe that the tree search algorithm is very sensitive to parameters h and r. The greedy algorithm is fastest among the three methods, and the least sensitive to increases in parameters h and r. However, about 30% of the solutions are not optimal. The logic-based Benders decomposition algorithm requires a solution time comparable with the tree search solution time for instances with small values of h and r. However, the solution time only increases moderately with h and r; thus, it has an advantage in solving instances with large h and r. For the instances for which tree search completes in a given time limit, that is, the solution is proven to be optimal, the logic-based Benders decomposition algorithm always finds a solution with the same objective value. The logic-based Benders decomposition algorithm finds even better solutions for three instances when the tree search terminates due to the time limit.

Table 4.9: Algorithms performance comparison for FP – instances with 50 nodes

Nodes	p	h	r	Tree search			Greedy search			LBD		
				Time	nbAPs	Obj.	Time	nbAPs	$\Delta_{Obj.}$	Time	nbAPs	$\Delta_{Obj.}$
50	15	3	3	1.57	25	576359.81	1.03	14	0	2.83	123	0
50	15	3	6	7.14	205	1014705.36	1.21	28	0	2.34	99	0
50	15	3	9	17.91	522	1545002.81	1.67	41	0	6.53	345	0
50	15	6	3	1.12	53	513927.51	0.52	27	16030.92	4.43	411	0
50	15	6	6	30.56	1200	754878.18	1.50	52	0	7.45	481	0
50	15	6	9	127.08	5387	990033.80	2.57	72	0	6.11	413	0
50	15	9	3	1.32	69	468233.12	0.66	40	0	5.31	637	0
50	15	9	6	51.94	2587	632548.83	1.92	71	0	4.31	291	0
50	15	9	9	208.78	11364	803304.32	3.28	101	0	3.70	354	0
50	20	3	3	0.87	24	487809.61	0.42	14	0	7.39	349	0
50	20	3	6	15.12	237	859628.19	1.89	28	0	16.93	333	0
50	20	3	9	56.45	563	1096057.21	4.62	41	0	18.58	225	0
50	20	6	3	1.52	43	391879.01	0.78	25	0	15.69	747	0
50	20	6	6	65.11	1303	561885.61	3.48	49	0	17.29	361	0
50	20	6	9	478.56	6401	748734.01	8.82	68	0	39.06	507	0
50	20	9	3	1.43	40	385594.23	1.08	37	0	317.88	21841	0
50	20	9	6	81.84	1730	496970.88	4.37	71	0	9.34	332	0
50	20	9	9	1104.28	18814	629121.70	10.84	101	0	24.95	563	0
50	30	3	3	6.12	36	248040.67	2.36	17	0	18.73	230	0
50	30	3	6	63.51	200	416827.89	8.60	30	26102.88	89.33	316	0
50	30	3	9	276.88	568	679664.45	19.62	43	0	139.11	343	0
50	30	6	3	14.70	97	221107.64	4.36	31	0	538.63	6936	0
50	30	6	6	434.89	1523	367165.16	16.06	52	9143.87	598.85	2766	0
50	30	6	9	3600.30	8176	495450.15	31.66	73	41090.27	985.66	2515	0
50	30	9	3	25.17	161	177946.84	6.67	46	0	82.92	1252	0
50	30	9	6	1486.63	6694	283285.57	23.53	72	0	588.83	2353	0
50	30	9	9	3600.72	10272	379873.48	52.35	107	4335.09	971.75	2556	0

Table 4.10: Algorithms performance comparison for FP – instances with 75 nodes

Nodes	p	h	r	Tree search			Greedy search			LBD		
				Time	nbAPs	Obj.	Time	nbAPs	$\Delta_{Obj.}$	Time	nbAPs	$\Delta_{Obj.}$
75	15	3	3	0.83	31	771614.08	0.34	14	0	3.65	226	0
75	15	3	6	9.56	215	1226376.57	1.22	28	0	3.99	140	0
75	15	3	9	26.15	513	1792170.32	2.26	41	0	6.65	269	0
75	15	6	3	1.93	76	633034.67	0.61	29	31059.66	8.08	656	0
75	15	6	6	45.29	1364	923224.84	1.97	51	0.00	9.23	431	0
75	15	6	9	186.50	5700	1173857.88	3.33	71	0	7.86	378	0
75	15	9	3	2.27	98	573274.94	0.84	43	0	3.14	306	0
75	15	9	6	78.71	3120	779724.55	2.46	72	0	4.07	270	0
75	15	9	9	293.14	12016	956783.96	4.24	101	2348.67	4.22	345	0
75	20	3	3	1.39	28	567202.54	0.67	14	0	5.89	186	0
75	20	3	6	23.62	237	1033377.15	3.04	28	0	36.58	458	0
75	20	3	9	58.58	441	1304001.37	5.88	41	0	31.51	335	0
75	20	6	3	2.59	48	467689.76	1.40	28	0	16.22	537	0
75	20	6	6	119.54	1473	663518.28	4.46	49	64167.79	22.52	359	0
75	20	6	9	519.26	4917	879042.78	10.81	71	0	48.71	667	0
75	20	9	3	2.69	53	464545.73	1.80	39	0	135.12	6942	0
75	20	9	6	204.22	3114	607673.73	5.29	69	0	27.04	692	0
75	20	9	9	1621.88	22524	739335.97	12.82	100	4083.85	25.18	448	0
75	30	3	3	11.38	35	308702.30	3.92	16	0	29.53	204	0
75	30	3	6	88.95	130	519644.36	17.92	30	0	126.43	233	0
75	30	3	9	374.38	403	737304.84	40.47	43	0	263.70	288	0
75	30	6	3	39.11	140	293699.82	6.31	28	3328.73	2602.49	21164	0
75	30	6	6	873.67	1435	419216.68	33.61	52	24859.59	558.01	1594	0
75	30	6	9	3600.33	3581	592307.48	84.34	74	41585.32	3616.80	5367	0
75	30	9	3	68.16	271	255942.65	9.60	42	0	334.70	3271	0
75	30	9	6	3260.57	8170	370632.39	46.91	77	5317.03	1132.90	3937	0
75	30	9	9	3600.12	5257	494146.25	105.93	110	23736.04	2636.10	5160	-12784.06

Table 4.11: Algorithms performance comparison for FP – instances with 100 nodes

Nodes	p	h	r	Tree search			Greedy search			LBD		
				Time	nbAPs	Obj.	Time	nbAPs	$\Delta_{Obj.}$	Time	nbAPs	$\Delta_{Obj.}$
100	15	3	3	1.21	29	860663.28	0.52	14	0	6.66	305	0
100	15	3	6	12.94	204	1357508.07	1.76	28	0	6.08	152	0
100	15	3	9	33.99	509	2003174.51	3.01	41	0	10.13	321	0
100	15	6	3	3.92	108	705433.14	0.92	31	43642.05	5.73	334	0
100	15	6	6	65.05	1384	1021655.03	2.78	52	0	11.33	381	0
100	15	6	9	249.63	5599	1287003.25	4.64	71	0	7.76	315	0
100	15	9	3	3.86	118	654041.10	1.36	46	0	4.99	368	0
100	15	9	6	102.92	2929	874605.92	3.54	73	0	4.78	270	0
100	15	9	9	390.22	12043	1081252.95	6.01	102	0	3.05	262	0
100	20	3	3	2.49	32	635698.99	1.01	14	0	7.58	168	0
100	20	3	6	33.45	241	1170888.29	4.60	30	10693.21	78.81	674	0
100	20	3	9	83.77	450	1456946.56	9.26	43	79151.73	38.27	280	0
100	20	6	3	4.98	60	545069.83	2.38	28	0	34.22	733	0
100	20	6	6	172.04	1461	745984.03	7.93	50	0	35.57	345	0
100	20	6	9	870.46	5647	985138.79	16.45	72	0	50.47	466	0
100	20	9	3	4.82	60	527275.04	3.06	39	0	120.87	4161	0
100	20	9	6	280.86	2994	685388.47	9.68	70	0	42.20	816	0
100	20	9	9	2537.95	25147	838072.48	19.51	100	111130.75	35.38	545	0
100	30	3	3	12.22	32	377883.17	4.60	16	0	38.35	231	0
100	30	3	6	104.17	118	601722.02	23.02	30	0	170.51	259	0
100	30	3	9	462.61	368	828137.57	55.21	43	0	404.14	316	0
100	30	6	3	39.16	102	341513.52	10.27	30	0	1880.08	9415	0
100	30	6	6	1098.38	1217	496235.51	45.21	52	14034.94	967.08	1691	0
100	30	6	9	3600.12	3167	662208.77	118.95	72	32942.93	2637.81	2698	0
100	30	9	3	73.78	201	309779.23	14.59	42	0	899.73	5848	0
100	30	9	6	3600.19	6082	423395.01	53.96	72	2519.00	1042.40	2402	0
100	30	9	9	3600.17	3574	562349.98	127.60	102	19955.23	3499.95	4822	-8245.34

Table 4.12: Algorithms performance comparison for FP – instances with 150 nodes

Nodes	p	h	r	Tree search			Greedy search			LBD		
				Time	nbAPs	Obj.	Time	nbAPs	$\Delta_{Obj.}$	Time	nbAPs	$\Delta_{Obj.}$
150	15	3	3	1.99	31	1013516.61	0.84	14	0	10.35	280	0
150	15	3	6	22.65	220	1608989.32	3.00	28	0	10.15	163	0
150	15	3	9	50.68	487	2258751.20	5.18	41	0	13.81	280	0
150	15	6	3	4.13	67	843480.11	1.53	28	0	27.27	917	0
150	15	6	6	96.53	1114	1241652.98	5.02	51	0	38.36	821	0
150	15	6	9	361.47	5077	1526860.23	7.93	71	0	14.64	386	0
150	15	9	3	4.31	75	807567.25	1.94	41	0	13.36	801	0
150	15	9	6	149.77	2589	1052999.80	5.44	69	0	8.38	347	0
150	15	9	9	573.06	11708	1289876.08	9.67	99	0	6.66	339	0
150	20	3	3	3.69	23	722874.51	1.95	14	0	17.08	205	0
150	20	3	6	53.69	200	1218049.89	7.79	28	0	65.42	279	0
150	20	3	9	154.10	491	1698605.53	15.13	41	0	67.09	313	0
150	20	6	3	5.90	34	656362.21	3.60	25	0	123.11	1511	0
150	20	6	6	277.92	1126	892969.86	15.61	48	0	76.73	448	0
150	20	6	9	1909.65	7372	1167456.35	27.66	70	916.97	85.45	524	0
150	20	9	3	6.21	40	632827.03	5.56	41	0	249.59	5276	0
150	20	9	6	625.29	3773	825139.47	23.14	74	4261.55	166.76	1737	0
150	20	9	9	3600.06	22324	1015841.44	35.02	101	22164.32	131.88	1235	0
150	30	3	3	19.94	33	448488.62	6.79	16	0	62.30	259	0
150	30	3	6	162.58	119	712366.40	35.60	30	0	426.34	427	0
150	30	3	9	683.84	373	969595.10	85.02	43	0	637.96	344	0
150	30	6	3	68.59	111	414030.45	14.21	29	0	964.17	3764	0
150	30	6	6	2218.82	1596	594611.90	67.78	52	18955.78	2260.65	2741	0
150	30	6	9	3602.11	2402	774668.70	158.81	72	55835.42	3179.83	2440	0
150	30	9	3	107.56	184	370223.08	21.92	43	0	460.65	2350	0
150	30	9	6	3600.71	3959	511672.90	81.96	72	4171.87	3067.12	4884	0
150	30	9	9	3600.08	2221	682826.96	164.74	101	998.87	3607.46	4190	-18626.93

Chapter 5

Conclusions and Future Research

5.1 Conclusions

In Chapter 3, we study a reliable facility location problem that generalizes the classical UFLP by considering random facility disruptions. We study the problem with the disruption distribution in its general form, that is, heterogeneous failure rates and correlated failures. We propose a novel way to characterize the available information of the disruption distribution. With that, we propose a robust reliable facility location model that includes several important problems studied in the literature. We propose a cutting plane algorithm based on the supermodularity of the problem. For the cases in which the distribution is fully known, the computational results show that the cutting plane algorithm not only outperforms the best-known algorithm in the literature that solves uncorrelated disruptions, but also efficiently solves moderate-sized problems with correlated disruptions. For the cases in which only marginal failure probabilities are known, the problem is equivalent to solving a stochastic model with a special distribution. The computational results show that the

cutting plane algorithm outperforms the Benders decomposition. The heuristic algorithm, multi-start tabu search, is shown to be very efficient and effective in solving large instances for the instances in which failures are independent.

In Chapter 4, we present a model along with solution methods for determining the optimal fortification plans for a distribution network considering both random facility disruptions and worst-case intelligent attacks. The model generalizes several important problems studied in the literature such as RIM, RIMF and RIMF-p. We show that the attacker's problem is equivalent to a supermodular function maximization problem with coordinate constraints, and an exact cutting plane algorithm is proposed. As shown by computational study, the cutting plane algorithm is very efficient in solving the attacker's problem and significantly outperforms the enumeration method. For the overall fortification problem, we show that the tree search algorithm developed for RIMF is also applicable for solving the problem, and a logic-based Benders decomposition algorithm is proposed. Computational study demonstrates that the logic-based Benders decomposition algorithm has advantages over the tree search algorithm when h, the number of facilities to fortify and r, the number of facilities to attack, are relatively large.

5.2 Future Research Direction

The problems we study only consider the effects of facility disruption on transportation costs; in reality, the effects can be manifold and complex. Thus, it would be interesting to take more realistic cost components into consideration. It would also be interesting to study the problems in a multi-product and/or multi-layer setting.

It would also be interesting to extend the models studied in this work to solve critical infrastructure problems. The location and protection of facilities is important from a national security perspective and one mission of the Department of Homeland Security (DHS) is to improve critical infrastructure security and resilience. The DHS has identified 16 critical infrastructure sectors (CIs) that play a critical role in national security. Almost all of them are associated with certain facility types, such as cellular towers in the communications sector and power plants in the energy sector. One important characteristic of CIs is that they are interconnected and interdependent on multiple levels. The interdependency of CIs is either caused by physical proximity or operational interaction. Because the models we studied are relevant for contexts with correlated disruptions, the solution methods may be adaptable to solve location and fortification problems for CIs.

For both problems investigated by this study, facilities are assumed to have unlimited capacity. The models and theoretical results in this work will no longer be applicable when facility capacity is considered. When capacity is considered, the nearest assignment property is no longer holds and one must solve an assignment problem to evaluate a solution. Moreover, the supermodular properties are very probably violated. Therefore, it is much harder to solve the problem once facility capacity is considered.

For the fortification problem, there are several possible research directions. First, h and r are predetermined/known in this study, but in practice, h is actually a decision made by the network planner and r is often hard to accurately estimate when the network planner makes the fortification decision. It would therefore be interesting to extend the model to explicitly model h as a decision variable and r with a probability distribution. Second, the network/facilities are predetermined. It is interesting to extend this work to consider facility

location decisions, in other words, to simultaneously make the location and fortification decisions

Bibliography

[1] Aboolian, R., Cui, T., and Shen, Z.-J. M. (2013). An efficient approach for solving reliable facility location models. *INFORMS Journal on Computing*, 25(4):720–729. 5, 6, 13, 20, 40, 46, 47, 48, 50, 59, 61, 63, 87

[2] Ahmadi-Javid, A. and Seddighi, A. H. (2013). A location-routing problem with disruption risk. *Transportation Research Part E: Logistics and Transportation Review*, 53:63–82. 15

[3] Ahuja, R. K., Orlin, J. B., Pallottino, S., Scaparra, M. P., and Scutellà, M. G. (2004). A multi-exchange heuristic for the single-source capacitated facility location problem. *Management Science*, 50(6):749–760. 11

[4] Aksen, D., Piyade, N., and Aras, N. (2010). The budget constrained r-interdiction median problem with capacity expansion. *Central European Journal of Operations Research*, 18(3):269–291. 17

[5] Al-Sultan, K. S. and Al-Fawzan, M. A. (1999). A tabu search approach to the uncapacitated facility location problem. *Annals of Operations Research*, 86:91–103. 11, 40

[6] Arkin, E., Joneja, D., and Roundy, R. (1989). Computational complexity of uncapacitated multi-echelon production planning problems. *Operations research letters*, 8(2):61–66. 11

[7] AtamtüRk, A. and Narayanan, V. (2008). Polymatroids and mean-risk minimization in discrete optimization. *Operations Research Letters*, 36(5):618–622. 80

[8] Aydin, N. and Murat, A. (2013). A swarm intelligence based sample average approximation algorithm for the capacitated reliable facility location problem. *International Journal of Production Economics*, 145(1):173–183. 14

[9] Barros, A. I. (1998). *Discrete and fractional programming techniques for location models*, volume 3. Springer. 22

[10] Berman, O., Krass, D., and Menezes, M. B. (2007). Facility reliability issues in network p-median problems: strategic centralization and co-location effects. *Operations Research*, 55(2):332–350. 13

[11] Berman, O., Krass, D., and Menezes, M. B. (2009). Locating facilities in the presence of disruptions and incomplete information. *Decision Sciences*, 40(4):845–868. 14

[12] Berman, O., Krass, D., and Menezes, M. B. (2013). Location and reliability problems on a line: Impact of objectives and correlated failures on optimal location patterns. *Omega*, 41(4):766–779. 14

[13] Canel, C., Khumawala, B. M., Law, J., and Loh, A. (2001). An algorithm for the capacitated, multi-commodity multi-period facility location problem. *Computers & Operations Research*, 28(5):411–427. 11

[14] Cashell, B. W. and Labonte, M. (2005). The macroeconomic effects of hurricane katrina. Congressional Research Service, Library of Congress. 3

[15] Chen, Q., Li, X., and Ouyang, Y. (2011). Joint inventory-location problem under the risk of probabilistic facility disruptions. *Transportation Research Part B: Methodological*, 45(7):991–1003. 15

[16] Christopher, M. and Peck, H. (2004). Building the resilient supply chain. *The International Journal of Logistics Management*, 15(2):1–14. 3

[17] Chu, Y. and Xia, Q. (2004). Generating benders cuts for a general class of integer programming problems. *Lecture notes in computer science*, 3011:127–141. 83

[18] Chudak, F. A. and Shmoys, D. B. (2003). Improved approximation algorithms for the uncapacitated facility location problem. *SIAM Journal on Computing*, 33(1):1–25. 11

[19] Church, R., Scaparra, M., and Middleton, R. (2004). The r-interdiction median problem and the rinterdiction covering problem. *Annals of the Association of American Geographers*, 94:491–502. 7, 16, 71

[20] Church, R. and Velle, C. R. (1974). The maximal covering location problem. *Papers in regional science*, 32(1):101–118. 10

[21] Church, R. L. and Scaparra, M. P. (2007). Protecting critical assets: the r-interdiction median problem with fortification. *Geographical Analysis*, 39(2):129–146. 6, 16, 17, 71

[22] Contreras, I. and Fernández, E. (2014). Hub location as the minimization of a supermodular set function. *Operations Research*, 62(3):557–570. 25

[23] Cornuéjols, G., Nemhauser, G. L., and Wolsey, L. A. (1983). The uncapacitated facility location problem. Technical report, Carnegie-mellon univ pittsburgh pa management sciences research group. 11

[24] Cui, T., Ouyang, Y., and Shen, Z.-J. M. (2010). Reliable facility location design under the risk of disruptions. *Operations Research*, 58(4):998–1011. 5, 6, 13, 20, 87

[25] Drezner, Z. (1984). The planar two-center and two-median problems. *Transportation Science*, 18(4):351–361. 10

[26] Drezner, Z. (1987). Heuristic solution methods for two location problems with unreliable facilities. *Journal of the Operational Research Society*, 38(6):509–514. 12

[27] Edmonds, J. (1970). Submodular functions, matroids, and certain polyhedra. *Edited by G. Goos, J. Hartmanis, and J. van Leeuwen*, 11. 80

[28] Efroymson, M. and Ray, T. (1966). A branch-bound algorithm for plant location. *Operations Research*, 14(3):361–368. 11

[29] Erlenkotter, D. (1978). A dual-based procedure for uncapacitated facility location. *Operations Research*, 26(6):992–1009. 11

[30] Escobar, J. W., Linfati, R., Baldoquin, M. G., and Toth, P. (2014). A granular variable tabu neighborhood search for the capacitated location-routing problem. *Transportation Research Part B: Methodological*, 67:344–356. 40

[31] Fader, P. S. and Hardie, B. G. (2007). How to project customer retention. *Journal of Interactive Marketing*, 21(1):76–90. 55, 56

[32] Farahani, R. Z., Asgari, N., Heidari, N., Hosseininia, M., and Goh, M. (2012). Covering problems in facility location: A review. *Computers & Industrial Engineering*, 62(1):368–407. 11

[33] Gade, D. and Pohl, E. (2009). Sample average approximation applied to the capacitated-facilities location problem with unreliable facilities. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 223(4):259–269. 14

[34] Gao, L.-L. and Robinson, E. P. (1992). A dual-based optimization procedure for the two-echelon uncapacitated facility location problem. *Naval Research Logistics (NRL)*, 39(2):191–212. 11

[35] Geoffrion, A. and Bride, R. M. (1978). Lagrangean relaxation applied to capacitated facility location problems. *AIIE transactions*, 10(1):40–47. 11

[36] Glover, F. (1989). Tabu search-part i. *ORSA Journal on Computing*, 1(3):190–206. 40

[37] Glover, F. (1990). Tabu search-part ii. *ORSA Journal on Computing*, 2(1):4–32. 40

[38] Griffiths, D. (1973). Maximum likelihood estimation for the beta-binomial distribution and an application to the household distribution of the total number of cases of a disease. *Biometrics*, pages 637–648. 55

[39] Hakimi, S. L. (1964). Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations research*, 12(3):450–459. 10

[40] Hakimi, S. L. (1965). Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations Research*, 13(3):462–475. 10

[41] Ho, S. C. (2015). An iterated tabu search heuristic for the single source capacitated facility location problem. *Applied Soft Computing*, 27:169–178. 40

[42] Hooker, J. N. and Ottosson, G. (2003). Logic-based benders decomposition. *Mathematical Programming*, 96(1):33–60. 82

[43] Hormozi, A. M. and Khumawala, B. M. (1996). An improved algorithm for solving a multi-period facility location problem. *IIE transactions*, 28(2):105–114. 11

[44] Iwata, S. and Orlin, J. B. (2009). A simple combinatorial algorithm for submodular function minimization. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1230–1237. Society for Industrial and Applied Mathematics. 79

[45] Iyer, R. K. and Bilmes, J. A. (2013). Submodular optimization with submodular cover and submodular knapsack constraints. In *Advances in Neural Information Processing Systems*, pages 2436–2444. 79

[46] Krarup, J. and Pruzan, P. M. (1983). The simple plant location problem: survey and synthesis. *European journal of operational research*, 12(1):36–81. 11

[47] Kulik, A., Shachnai, H., and Tamir, T. (2009). Maximizing submodular set functions subject to multiple linear constraints. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 545–554. Society for Industrial and Applied Mathematics. 79

[48] Latour, A. (2001). Trial by fire: A blaze in albuquerque sets off major crisis for cell-phone giants. *Wall Street Journal*, 1(29):2001. 3

[49] Lee, J., Mirrokni, V. S., Nagarajan, V., and Sviridenko, M. (2009). Non-monotone submodular maximization under matroid and knapsack constraints. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 323–332. ACM. 79

[50] Li, Q., Zeng, B., and Savachkin, A. (2013). Reliable facility location design under disruptions. *Computers & Operations Research*, 40(4):901–909. 16

[51] Li, X. and Ouyang, Y. (2010). A continuum approximation approach to reliable facility location design under correlated probabilistic disruptions. *Transportation Research Part B: Methodological*, 44(4):535–548. 5, 14, 40, 53, 55

[52] Liberatore, F., Scaparra, M. P., and Daskin, M. S. (2011). Analysis of facility protection strategies against an uncertain number of attacks: The stochastic r-interdiction median problem with fortification. *Computers & Operations Research*, 38(1):357–366. 71

[53] Liberatore, F., Scaparra, M. P., and Daskin, M. S. (2012). Hedging against disruptions with ripple effects in location analysis. *Omega*, 40(1):21–30. 17

[54] Lim, M., Daskin, M. S., Bassamboo, A., and Chopra, S. (2010). A facility reliability problem: formulation, properties, and algorithm. *Naval Research Logistics (NRL)*, 57(1):58–70. 16

[55] Lu, M., Ran, L., and Shen, Z.-J. M. (2015). Reliable facility location design under uncertain correlated disruptions. *Manufacturing & Service Operations Management*. 14, 20, 24, 37, 46, 47, 53, 58, 59, 60

[56] Manne, A. S. (1964). Plant location under economies-of-scaledecentralization and computation. *Management Science*, 11(2):213–235. 11

[57] Martí, R. (2003). Multi-start methods. In *Handbook of Metaheuristics*, pages 355–368. Springer. 43

[58] McCormick, S. T. (2005). Submodular function minimization. *Handbooks in operations research and management science*, 12:321–391. 79

[59] Megiddo, N. and Supowit, K. J. (1984). On the complexity of some common geometric location problems. *SIAM journal on computing*, 13(1):182–196. 10

[60] Melo, M. T., Nickel, S., and Saldanha-Da-Gama, F. (2009). Facility location and supply chain management–a review. *European journal of operational research*, 196(2):401–412. 12

[61] Michallet, J., Prins, C., Amodeo, L., Yalaoui, F., and Vitry, G. (2014). Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services. *Computers & Operations Research*, 41:196–207. 43

[62] Michel, L. and Van Hentenryck, P. (2004). A simple tabu search for warehouse location. *European Journal of Operational Research*, 157(3):576–591. 40, 42

[63] Minieka, E. (1970). The m-center problem. *Siam Review*, 12(1):138–139. 10

[64] Nanto, D. K. (2011). *Japan's 2011 Earthquake and Tsunami: Economic Effects and Implications for the United States*. DIANE Publishing. 3

[65] Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and combinatorial optimization*, volume 18. Wiley New York. 22

[66] Nickel, S. and da Gama, F. S. (2015). Multi-period facility location. In *Location science*, pages 289–310. Springer. 11

[67] Nicola, V. F. and Goyal, A. (1990). Modeling of correlated failures and community error recovery in multiversion software. *Software Engineering, IEEE Transactions on*, 16(3):350–359. 55

[68] O'Hanley, J. R., Scaparra, M. P., and García, S. (2013). Probability chains: A general linearization technique for modeling reliability in facility location and related problems. *European Journal of Operational Research*, 230(1):63–75. 13

[69] Orlin, J. B. (2009). A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251. 79

[70] Owen, S. H. and Daskin, M. S. (1998). Strategic facility location: A review. *European journal of operational research*, 111(3):423–447. 12

[71] Pirkul, H. and Jayaraman, V. (1996). Production, transportation, and distribution planning in a multi-commodity tri-echelon system. *Transportation Science*, 30(4):291–302. 11

[72] Qi, L., Shen, Z.-J. M., and Snyder, L. V. (2010). The effect of supply disruptions on supply chain design decisions. *Transportation Science*, 44(2):274–289. 15

[73] Reese, J. (2006). Solution methods for the p-median problem: An annotated bibliography. *Networks*, 48(3):125–142. 10

[74] Resende, M. G. and Werneck, R. F. (2006). A hybrid multistart heuristic for the uncapacitated facility location problem. *European Journal of Operational Research*, 174(1):54–68. 43

[75] Revelle, C. S., Eiselt, H. A., and Daskin, M. S. (2008). A bibliography for some fundamental problem categories in discrete location science. *European Journal of Operational Research*, 184(3):817–848. 10

[76] Scaparra, M. P. and Church, R. L. (2008a). A bilevel mixed-integer program for critical infrastructure protection planning. *Computers & Operations Research*, 35(6):1905–1923. 6, 17, 71, 76, 94

[77] Scaparra, M. P. and Church, R. L. (2008b). An exact solution approach for the interdiction median problem with fortification. *European Journal of Operational Research*, 189(1):76–92. 17, 71

[78] Sheffi, Y. (2001). Supply chain management under the threat of international terrorism. *The International Journal of Logistics Management*, 12(2):1–11. 3

[79] Shen, Z.-J. M. (2005). A multi-commodity supply chain design problem. *Iie Transactions*, 37(8):753–762. 11

[80] Shen, Z.-J. M., Zhan, R. L., and Zhang, J. (2011). The reliable facility location problem: Formulations, heuristics, and approximation algorithms. *INFORMS Journal on Computing*, 23(3):470–482. 4, 5, 12, 13, 14, 20, 87

[81] Snyder, L. V., Atan, Z., Peng, P., Rong, Y., Schmitt, A. J., and Sinsoysal, B. (2016). Or/ms models for supply chain disruptions: A review. *IIE Transactions*, 48(2):89–109. 3, 12, 15, 16

[82] Snyder, L. V. and Daskin, M. S. (2005). Reliability models for facility location: the expected failure cost case. *Transportation Science*, 39(3):400–416. 4, 6, 12, 13, 20

[83] Snyder, L. V. and Daskin, M. S. (2006). Stochastic p-robust location problems. *IIE Transactions*, 38(11):971–985. 12, 14

[84] Sridharan, R. (1995). The capacitated plant location problem. *European Journal of Operational Research*, 87(2):203–213. 11

[85] Sun, M. (2006). Solving the uncapacitated facility location problem using tabu search. *Computers & Operations Research*, 33(9):2563–2589. 40

[86] Sun, M. (2012). A tabu search heuristic procedure for the capacitated facility location problem. *Journal of Heuristics*, 18(1):91–118. 40

[87] Sviridenko, M. (2004). A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43. 79

[88] Tang, C. S. (2006). Robust strategies for mitigating supply chain disruptions. *International Journal of Logistics: Research and Applications*, 9(1):33–45. 3

[89] Tragantalerngsak, S., Holt, J., and Rönnqvist, M. (2000). An exact method for the two-echelon, single-source, capacitated facility location problem. *European Journal of Operational Research*, 123(3):473–489. 11

[90] Tsiakis, P., Shah, N., and Pantelides, C. C. (2001). Design of multi-echelon supply chain networks under demand uncertainty. *Industrial & Engineering Chemistry Research*, 40(16):3585–3604. 11

[91] Weinberg, C. R. and Gladen, B. C. (1986). The beta-geometric distribution applied to comparative fecundability studies. *Biometrics*, pages 547–560. 55

[92] Wu, T. and Zhang, K. (2014). A computational study for common network design in multi-commodity supply chains. *Computers & Operations Research*, 44:206–213. 79

[93] Zhang, X., Zheng, Z., Zhu, Y., and Cai, K.-Y. (2014). Protection issues for supply systems involving random attacks. *Computers & Operations Research*, 43:137–156. 17

[94] Zhang, Y., Snyder, L. V., Qi, M., and Miao, L. (2016a). A heterogeneous reliable location model with risk pooling under supply disruptions. *Transportation Research Part B: Methodological*, 83:151–178. 15

[95] Zhang, Y., Snyder, L. V., Ralphs, T. K., and Xue, Z. (2016b). The competitive facility location problem under disruption risks. *Transportation Research Part E: Logistics and Transportation Review*, 93:453–473. 14

[96] Zhu, Y., Zheng, Z., Zhang, X., and Cai, K. (2013). The r-interdiction median problem with probabilistic protection and its solution algorithm. *Computers & Operations Research*, 40(1):451–462. 6, 7, 17, 68, 71, 94

Appendices

A Parameter-Tuning of the MSTS Algorithm

A.1 Parameter *StabilityLimit*

We first try to tune the parameter *StabilityLimit* – the number of iterations without improvement before completing one start of the tabu search. We test the tabu search with different values of *StabilityLimit* for solving benchmark instances with 50, 75 and 100 nodes with $\alpha = 1$. Table A1 gives the results. For each row, 10000 random runs of the tabu search (1 start) are performed. Column titled "P_{opt}" reports the percentage of runs producing the optimal solution (we compare the objective value and the best objective value obtained by the exact algorithm). Column titled "Iterations" reports the average number of iterations for each run of tabu search. Column titled "Time" reports the average time in seconds for each run of tabu search. It can be seen from the table, in general, both the percentage of runs producing the optimal and the average number of iterations increase with the value of *StabilityLimit*. For each problem scale, the computational time is proportional to number of iterations. Thus, there is a need to choose a value for *StabilityLimit* to balance solution quality and solution time. For example, the 50 nodes instance, when *StabilityLimit* is set to 1, on average, it takes 23 iterations and the probability of finding optimal solution is 3.27%. If we perform 10 runs of the tabu search, the chance of finding optimal solution becomes $P_{opt} = 1 - (1 - 3.27\%)^{10} = 28.28\%$ and it takes 230 iterations on average. Comparing these number with $StabilityLimit = 100$, it is obviously better to set $StabilityLimit = 100$. In table A2, we compute a the chances of finding optimal solution of different value of *StabilityLimit* by assuming a total 500 iterations available. It can be seen from the table, in general, the chances first increase then decrease with the value of

StabilityLimit. *StabilityLimit* = 20 has a very good performance and is chosen for all remaining experiments.

Table A1: Tabu search performance with regard to *StabilityLimit*

Nodes	*StabilityLimit*	$P_{opt}(\%)$	Iterations	Time
50	1	3.27	23	0.01
50	3	8.65	26	0.01
50	5	11.8	30	0.01
50	10	16.36	40	0.01
50	20	65.26	61	0.02
50	50	74.03	96	0.03
50	100	79.39	150	0.05
50	200	78.65	251	0.08
75	1	1.9	34	0.04
75	3	8.59	39	0.04
75	5	12.31	42	0.04
75	10	44.36	54	0.05
75	20	66.27	70	0.06
75	50	68.56	103	0.09
75	100	70.04	156	0.13
75	200	72.22	262	0.20
100	1	9.41	46	0.10
100	3	11.29	50	0.11
100	5	11.68	54	0.11
100	10	11.53	61	0.12
100	20	12.39	74	0.14
100	50	12.04	109	0.19
100	100	12.89	160	0.26
100	200	12.43	263	0.40

Table A2: Chance of finding optimal solutions with 500 iterations

Nodes	*StabilityLimit*							
	1	3	5	10	20	50	100	200
50	0.5195	0.8210	0.8770	0.8920	0.9998	0.9991	0.9949	0.9536
75	0.2458	0.6817	0.7869	0.9958	0.9996	0.9963	0.9790	0.9133
100	0.6597	0.6981	0.6865	0.6332	0.5907	0.4462	0.3495	0.2234

A.2 Parameter *nStarts*

We examine how the number of starts affects the algorithms performance. We solve the benchmark instances with $\alpha = 1$ with difference number of starts. The results are reported in Table A3. In the column titled "Chance", we compute the estimated chance of finding

optimal solution based on Table A1. It can be seen that when the chance of find optimal solution increases with the increase of number of starts and the chances become very close to 1 when the number of starts greater than 20. From the table, we observe that the number of iterations and time used is proportional to the number of starts.

Table A3: Algorithm performance with regard to *nStarts*

Nodes	*nStarts*	Upper Bound			Iterations	Time	Chance
		Min	Ave	Max			
50	3	1,020,180	1,020,180	1,020,180	176.2	0.0693	0.958073
50	5	1,020,180	1,020,180	1,020,180	305.4	0.1060	0.994940
50	10	1,020,180	1,020,180	1,020,180	589.3	0.2014	0.999974
50	20	1,020,180	1,020,180	1,020,180	1250.2	0.4331	1.000000
50	50	1,020,180	1,020,180	1,020,180	3108.3	1.0666	1.000000
50	80	1,020,180	1,020,180	1,020,180	4814.0	1.6802	1.000000
50	100	1,020,180	1,020,180	1,020,180	6137.6	2.1053	1.000000
75	3	1,148,490	1,148,490	1,148,490	218.0	0.2022	0.961625
75	5	1,148,490	1,148,490	1,148,490	369.2	0.3435	0.995634
75	10	1,148,490	1,148,490	1,148,490	711.1	0.6553	0.999981
75	20	1,148,490	1,148,490	1,148,490	1391.9	1.2463	1.000000
75	50	1,148,490	1,148,490	1,148,490	3549.5	3.2131	1.000000
75	80	1,148,490	1,148,490	1,148,490	5653.6	5.1599	1.000000
75	100	1,148,490	1,148,490	1,148,490	6908.1	6.2522	1.000000
100	3	1,252,600	1,255,250	1,259,110	228.2	0.4349	0.327548
100	5	1,252,600	1,253,930	1,260,060	378.6	0.7543	0.483859
100	10	1,252,600	1,252,890	1,253,580	744.3	1.4543	0.733598
100	20	1,252,600	1,252,600	1,252,600	1532.5	2.9813	0.929030
100	50	1,252,600	1,252,600	1,252,600	3687.9	6.9429	0.998658
100	80	1,252,600	1,252,600	1,252,600	5857.5	11.0321	0.999975
100	100	1,252,600	1,252,600	1,252,600	7436.0	13.7748	0.999998

Vita

Kaike Zhang was born in Hunan, China. He graduated in 2012 with a Bachelors degree in Information Management and Information Systems from Southeast University, China. In the fall of 2013, he joined Dr. Xueping Lis group as a graduate assistant at the Department of Industrial and Systems Engineering of University of Tennessee, Knoxville (UTK). Prior to his studies at UTK, he was a research assistant at Institute of Process Engineering, Chinese Academy of Sciences. He is expected to complete his Doctor of Philosophy degree in 2017. His research interests include logistics and supply chain management, computational integer programming. He has published several peer-review journal and conference papers. He is a student member of the Institute for Operations Research and Management Sciences (INFORMS). In 2016, he worked as an Operations Research Analyst Co-op at Monsanto Company.