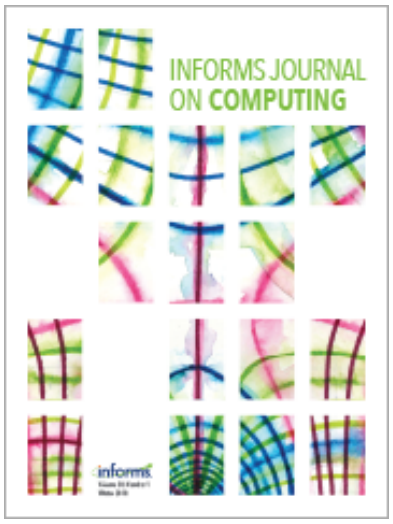


INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

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

Kaike Zhang, Xueping Li, Mingzhou Jin

To cite this article:
Kaike Zhang, Xueping Li, Mingzhou Jin (2021) Efficient Solution Methods for a General r-Interdiction Median Problem with Fortification. INFORMS Journal on Computing

Published online in Articles in Advance 03 Dec 2021

. https://doi.org/10.1287/ijoc.2021.1111

http://pubsonline.informs.org/journal/ijoc

INFORMS JOURNAL ON COMPUTING
Articles in Advance, pp. 1–19
ISSN 1091-9856 (print), ISSN 1526-5528 (online)

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

Kaike Zhang,[a] Xueping Li,[a] Mingzhou Jin[a]

[a] Department of Industrial and Systems Engineering, University of Tennessee, Knoxville, Tennessee 37996
Contact: kzhang7@vols.utk.edu (KZ); xueping.li@utk.edu, https://orcid.org/0000-0003-1990-0159 (XL); jin@utk.edu, https://orcid.org/0000-0002-2387-8129 (MJ)

Received: November 5, 2019
Revised: August 24, 2020; March 18, 2021; May 30, 2021; June 20, 2021
Accepted: June 21, 2021
Published Online in Articles in Advance: December 3, 2021

https://doi.org/10.1287/ijoc.2021.1111

Abstract. This study generalizes the *r*-interdiction median (RIM) problem with fortification to simultaneously consider two types of risks: probabilistic exogenous disruptions and endogenous disruptions caused by intentional attacks. We develop a bilevel programming model that includes a lower-level interdiction problem and a higher-level fortification problem to hedge against such risks. We then prove that the interdiction problem is supermodular and subsequently adopt the cuts associated with supermodularity to develop an efficient cutting-plane algorithm to achieve exact solutions. For the fortification problem, we adopt the logic-based Benders decomposition (LBBD) framework to take advantage of the two-level structure and the property that a facility should not be fortified if it is not attacked at the lower level. Numerical experiments show that the cutting-plane algorithm is more efficient than benchmark methods in the literature, especially when the problem size grows. Specifically, with regard to the solution quality, LBBD outperforms the greedy algorithm in the literature with an up-to 13.2% improvement in the total cost, and it is as good as or better than the tree-search implicit enumeration method.

Summary of Contribution: This paper studies an *r*-interdiction median problem with fortification (RIMF) in a supply chain network that simultaneously considers two types of disruption risks: random disruptions that occur probabilistically and disruptions caused by intentional attacks. The problem is to determine the allocation of limited facility fortification resources to an existing network. It is modeled as a bilevel programming model combining a defender's problem and an attacker's problem, which generalizes the *r*-interdiction median problem with probabilistic fortification. This paper is suitable for IJOC in mainly two aspects: (1) The lower-level attacker's interdiction problem is a challenging high-degree nonlinear model. In the literature, only a total enumeration method has been applied to solve a special case of this problem. By exploring the special structural property of the problem, namely, the supermodularity of the transportation cost function, we developed an exact cutting-plane method to solve the problem to its optimality. Extensive numerical studies were conducted. Hence, this paper fits in the intersection of operations research and computing. (2) We developed an efficient logic-based Benders decomposition algorithm to solve the higher-level defender's fortification problem. Overall, this study generalizes several important problems in the literature, such as RIM, RIMF, and RIMF with probabilistic fortification (RIMF-*p*).

History: Accepted by David Alderson, Area Editor for Network Optimization: Algorithms and Applications.
Funding: This work was supported in part by the National Science Foundation [Grant CMMI 1634975].
Supplemental Material: The online supplement is available at https://doi.org/10.1287/ijoc.2021.1111.

Keywords: *r*-interdiction median problem with fortification • facility location problem • supply chain disruption • cutting-plane method • Benders decomposition

1. Introduction

Production and distribution facilities are among the most critical components in supply chain networks. When such facilities fail, whether caused by natural disasters, such as hurricanes and earthquakes, or by intentional or unintentional human actions, such as labor strikes, fires, cyberattacks, and terrorist strikes, massive negative effects often follow, including substantial increases in both service costs and customer dissatisfaction. To improve supply chain reliability and resilience, both proactive and reactive mitigation options can be used to counter the risk of facility disruptions. One proactive option is to add redundant facilities in the network design phase, so that the system can still perform 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 in enhanced facility infrastructure, and working to assure rapid recovery from disruption. In this article, we study how to fortify existing facilities against potential disruptions.

The issue of supply chain reliability considering disruption risks has received increasing attention in the past few decades from researchers. However, most studies have considered only the probabilistic risk that models natural disasters, as in Snyder and Daskin (2005), Cui et al. (2010), and Aboolian et al. (2013), and the worst-case risk that models man-made attacks, as in Church and Scaparra (2007), Scaparra and Church (2008a), and Zhu et al. (2013). In reality, however, disruption risks from different sources often exist simultaneously. Therefore, we argue that it is beneficial to have a general model that is able to consider different disruption risks at the same time.

The model we present in this work considers a generalized r-interdiction median problem with fortification (RIMF). The RIMF was first introduced by Church and Scaparra (2007). In the RIMF, a network of k 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. The RIMF solves the problem of allocating protective resources to the most critical h facilities in anticipation of the worst-case loss when r facilities are attacked. In the original RIMF, it was assumed that an attack on a protected facility would have no effect and that an attack on an unprotected facility would certainly be successful.

Despite precautions, a fortified facility may still be disrupted by an attack. Therefore, Zhu et al. (2013) have proposed the r-interdiction median problem with probabilistic fortification (RIMF-p), in which any facility, even a protected one, can be disrupted by an attack with some given probability of success. The RIMF-p adds uncertainty in the forms 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 (Church et al. 2004) as special cases. We present a bilevel mixed-integer nonlinear programming (MINLP) model for the problem. Due to its high degree of nonlinearity, the model is difficult to solve using standard approaches for bilevel programming models. Even for the lower-level problem alone, that is, the attacker's problem, there is no efficient solution method proposed in the literature. In this paper, we present an efficient cutting-plane method that exactly solves the attacker's problem. For the upper-level problem, that is, the network defender's problem, we propose solution methods based on the logic-based Benders decomposition framework.

Based on how a disruption risk is modeled, the research on supply chain disruptions is divided into two main streams. Snyder and Daskin (2005), Cui et al. (2010), Shen et al. (2011), and Aboolian et al. (2013) have modeled facility disruption as a probabilistic random event. The objective is to minimize the expected costs under all failure scenarios, and the problem is referred to as the *reliable facility location problem* (RFLP). Snyder et al. (2016) have classified 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 problem takes the failure risk into consideration at the supply chain design phase, and the solution to the problem determines the optimal facility locations. The facility fortification option is absent in most of these models, except for those in Lim et al. (2010) and Li et al. (2013). Lim et al. (2010) have studied the reliable facility location problem, in which both unreliable and reliable facilities (i.e., those that cannot be disrupted) can be set up. In the model, each customer is assigned to at most one unreliable facility, as well as one reliable facility as a backup. However, this approach may be unrealistic when an unreliable facility fails but a customer can be served by a closer unreliable facility. Li et al. (2013) extended Lim et al. (2010)'s problem by considering a limited fortification budget.

Another stream originates from the work of Church et al. (2004) and Church and Scaparra (2007). Their models consider 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. (2016) have classified this kind of risk as endogenous. Anticipating worst-case attacks, their objective is to fortify an existing network to minimize network costs. Church et al. (2004) proposed two interdiction models: the RIM problem and the r-interdiction covering problem (RIC). In the RIM problem, the attacker needs to choose r facilities to interdict among the k existing facilities to maximize the increase in the weighted distance. Church and Scaparra (2007) extended the previous study of the RIM problem (Church et al. 2004) by adding a level of fortification decision. This problem is referred to as the *RIMF problem*. In the RIMF problem, the defender wants to protect the network by

fortifying h facilities to minimize costs in the worst-case scenario, that is, when an attacker destroys r unfortified facilities. Scaparra and Church (2008b) 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 (2007). Scaparra and Church (2008a) formulate the RIMF problem as a bilevel integer programming model and propose a solution method based on a tree-search implicit enumeration procedure. Aksen et al. (2010) studied the fortification problem with a budget constraint in which the model determined the optimal number of facilities to fortify instead of using a predetermined number. They assumed that there was a capacity expansion cost when a customer was reassigned to a noninterdicted facility. Liberatore et al. (2012) considered the correlation effects between the facilities, as well as the fact that one attack may affect more than one facility. The facilities being affected may simply lose some of their capacity.

Zhu et al. (2013) introduced 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 proposed a model that generalizes the RIMF's bilevel formulation with probabilistic factors. However, they did not provide an efficient algorithm to solve the lower-level model, so a brute force total enumeration was used to find the solution. Zhang et al. (2014) considered random attacks, which may occur because of 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.

As we can see, there is ample established research in the literature to handle risks classified as *endogenous* by Snyder et al. (2016), such as RIM (Church et al. 2004), RIMF (Church and Scaparra 2007), and RIMF-p (Zhu et al. 2013). On the other hand, *exogenous* risks are considered in the RFLP, such as by Snyder and Daskin (2005), Cui et al. (2010), Lim et al. (2010), Shen et al. (2011), and Aboolian et al. (2013). In the past, these two lines of research were developed rather independently, and two types of risk were considered separately. In practice, however, both types of disruption risks exist and can happen at the same time, including nature disasters (e.g., winter storms, hurricanes, and earthquakes), and intentional or unintentional human actions (e.g., cyberattacks, labor strikes, tariffs). Thus, there is a need for a holistic approach to handle both uncertainties simultaneously, which we demonstrate in the online supplement. Models with or without the consideration of both uncertainties simultaneously can lead to different fortification and attack decisions.

In this work, we propose a bilevel model for a general RIMF problem that considers probabilistic disruptions and intelligent interdictions simultaneously. Efficient algorithms are developed for solving both lower-level and upper-level problems. Hence, the major contribution of the paper is twofold. First, we prove the supermodularity for the attacker's problem, to which, thus far in the literature, only total enumeration methods were applied. Second, we develop efficient solution methods to the attacker's interdiction problem and the defender's fortification problem. Thus, this research generalizes several important problems in the literature, including the RIM, RIMF, and RIMF-p. Moreover, the overall framework is applicable to other network optimization problems such as critical infrastructure and cyber-physical systems that face both natural disasters and intentional attacks.

The remainder of the paper is organized as follows: In Section 2, a general r-interdiction median with fortification problem is described and formulated. Section 3 discusses some properties of the model. Solution algorithms are developed in Section 4. The computational study is reported in Section 5. The conclusion and recommendations for future research are discussed in Section 6.

2. Problem Definition and Formulation

Consider an existing supply network with a set of operating facilities $J = \{1, \ldots, k\}$ and a set of customer demand aggregation points $I = \{1, \ldots, |I|\}$. 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 the customers and the facilities. The capacity of a facility is assumed to be unlimited. Thus, a customer is always served by the nearest working facility. Furthermore, we assume that there is an emergency facility 0 to model the costs of lost sales. If the demand of customer $i \in I$ is not served, then 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 of open facilities, including the emergency facility; that is, $J^+ = \{0, 1, \ldots, k\}$.

Facilities face two types of disruption risks simultaneously: (1) probabilistic, exogenous risks and (2) worst-case, endogenous risks. The two types of risks are independent of each other. A facility may fail 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 interdiction 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, then an attack succeeds with certainty. However, if the facility is fortified, then 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 and to anticipate worst-case attacks.

The problem can be viewed as a leader-follower game, as illustrated by Figure 1: the leader, that is, the network planner or defender, aims to minimize costs by making fortification decisions; the follower, that is, the attacker, then tries to interdict the network to maximize the damage. The following bilevel model is formulated by adding the probabilistic disruption factor to the RIMF-p model proposed by Zhu et al. (2013).

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:

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

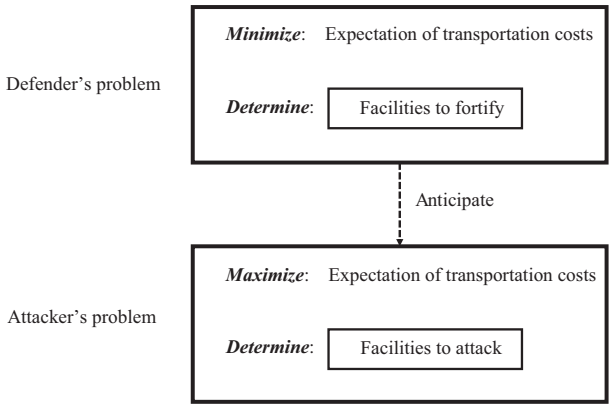
The fortification problem (FP) is written as

$$\min F(z) \tag{1}$$

$$\text{subject to (s.t.)} \quad \sum_{j\in J} z_j \le h, \quad z_j \in \{0,1\} \tag{2}$$

We denote by $F(z)$ the expected transportation cost under the worst-case attack for a fortification decision z; $F(z)$ is computed by solving the attacker's problem.

Figure 1. The Bilevel Problem: Defender's Problem and Attacker's Problem



Denote by p_j the failure probability of facility j, which is an expression related to s_j:

$$p_j = 1 - (1 - q_j)(1 - s_j w^{z_j}). \tag{3}$$

When $z_j = 0$, that is, when the facility is not fortified, $w^{z_j} = 1$, and the facility is disrupted with certainty when it is attacked; that is, $p_j = 1$, as long as $s_j = 1$. Otherwise, the facility fails when either the probabilistic disruption, with probability q_j, takes place or when an intentional attack succeeds, with probability w. We want to note that the extension in Equation (3) from the literature is nontrivial. As it provides us a holistic approach to handle both *endogenous* and *exogenous* uncertainties simultaneously, such a unified model brings modeling complexities. The presence of both fortification decisions z_j and probabilistic disruptions q_j leads to computational challenges, which motivates us to further develop efficient algorithms.

We can then compute the probability that customer i is assigned to its vth closest facility as the following:

$$\beta_i^v = (1 - p_{i_v}) \times \prod_{l=1}^{v-1} p_{i_l}, \tag{4}$$

where i_v is the index of vth closest facility of i. Defining $c_i^v = c_{i,i_v} d_i$, the expected transportation costs can be expressed as follows:

$$\sum_{i\in I} \sum_{v=1}^{|J^+|} c_i^v \beta_i^v = \sum_{i\in I} \sum_{v=1}^{|J^+|} c_i^v (1 - p_{i_v}) \times \prod_{l=1}^{v-1} p_{i_l}. \tag{5}$$

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

$$F(z) = \max \sum_{i\in I} \sum_{v=1}^{|J^+|} c_i^v (1 - p_{i_v}) \times \prod_{l=1}^{v-1} p_{i_l} \tag{6}$$

$$\text{s.t.} \quad \sum_{j\in J} s_j \le r \tag{7}$$

$$p_j = 1 - (1 - q_j)(1 - s_j w^{z_j}) \quad \forall j \in J^+, \quad s_j \in \{0,1\}, \quad \forall j \in J^+. \tag{8}$$

In the bilevel model, the defender makes an upper-level decision, and the objective is to minimize expected transportation costs under a worst-case attack by solving the attacker's problem. Constraints (2) and (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. Even for a special case such as the one studied by Zhu et al. (2013), in which only disruptions caused by intentional attacks are considered, there is no efficient solution method developed 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 with certainty if a fortified facility is attacked. The problem has been studied by Church et al. (2004), Church and Scaparra (2007), Scaparra and Church (2008a, b), and Liberatore et al. (2011). 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, some relatively efficient solution methods exist in the literature (e.g., Scaparra and Church 2008a, b). Our formulation makes the RIMF-p more general to incorporate not only exogenous failures but also the heterogeneity among facilities. In the literature, the risk is considered the same across all facilities. It is worth noting that the generalized formulation, which captures more real-world features, does not add complexity in terms of computational complexity, as it only impacts the calculation of a linear term in the attacker's problem. In fact, the model and solution method that we propose next is able to handle other facility failure considerations, as long as the failure probability can be expressed as a linear term of attack decision. For example, the formulation can consider the case in which the endogenous risks are heterogeneous among facilities. In practice, the success probability of an attack could be related to a facility's conditions, hardware or software, no matter if they are fortified or not.

2.1. Linear Attacker's Problem Formulation

Given values of z_j, we can rewrite the failure probability of facility i expressed by (3) as

$$p_j = 1 - (1 - q_j)(1 - s_j w^{z_j}) = q_j + w_j s_j, \tag{9}$$

where

$$w_j = \begin{cases} 1 - q_j, & z_j = 0 \\ w - q_j w, & z_j = 1, \end{cases}$$

and they become known for the attacker's problem.

Let θ_i^v be the probability that the demand at customer i is not satisfied by its first vth closest facilities. We have

$$\theta_i^v = \prod_l^v p_{i_l}, \tag{10}$$

which is equivalent to

$$\begin{cases} \theta_i^1 = p_{i_1} \\ \theta_i^v = \theta_i^{v-1} \cdot p_{i_v} \end{cases} \tag{11}$$

The attacker's problem becomes

$$\begin{aligned} F(z) = F(w_j) &= max \sum_{i \in I} \sum_{v=1}^{|J^+|} c_i^v (\theta_i^{v-1} - \theta_i^v) \\ &= max \sum_{i \in I} \sum_{v=1}^{|J^+|} \left[(c_i^{v+1} - c_i^v)\theta_i^v + c_i^1 \right] \end{aligned} \tag{12}$$

s.t. (7) in AP

$$\theta_i^v \le \theta_i^{v-1} q_{i_v} + \theta_i^{v-1} w_{i_v} + M(1 - s_{i_v}) \quad \forall i \in I, 1 \le v \le |J^+| \tag{13}$$

$$\theta_i^v \le \theta_i^{v-1} q_{i_v} + M s_{i_v} \quad \forall i \in I, 1 \le v \le |J^+| \tag{14}$$

$$\theta_i^v \in [0,1], \; s_j \in \{0,1\}.$$

The rationale for Constraints (13) and (14) is from the following equation:

$$\begin{aligned} \theta_i^v &= \theta_i^{v-1} \cdot p_{i_v} \\ &= \theta_i^{v-1} q_{i_v} + \theta_i^{v-1} s_{i_v}. \end{aligned}$$

Since all coefficients of θ_i^v, that is, $c_i^{v+1} - c_v$, are greater than or equal to 0 in the objective function, the attacker has the incentive to maximize all θ_i^v. Therefore, (13) and (14) are enough to stipulate the equation.

3. Model Properties

In this section, we discuss the structural properties of the model from a set-function point of view; this will provide a theoretical cornerstone for the solution methods developed in the next section. Supermodular properties and their counterparts, submodular properties, are the properties for a set function, and are conceptually similar to the convex and concave properties in the optimization of a real domain.

First, we recall some well-known results of supermodular set functions (see Nemhauser and Wolsey 1988, chapter III, section 3.1). Let U be a finite set, and let f be a real-valued function on the subsets of set U. Denote by $\rho_e^f(S) := f(S \cup \{e\}) - f(S)$ the incremental value of adding an element e to the set S. We may omit the superscript and write it as $\rho_e(S)$ when the relevant set function is clear in the context. We have the following definition and properties of supermodular set functions.

Definition 1. A set function f is supermodular if one of the following equivalent statements is satisfied:
(1) $f(S) + f(T) \le f(S \cap T) + f(S \cup T)$, for all $S, T \subseteq U$;
(2) $\rho_e(S) \le \rho_e(T)$, for all $S \subseteq T \subseteq U$ and $e \in U \setminus T$.

Lemma 1. *A positive linear combination of supermodular functions is supermodular.*

Definition 2. A set function f is defined as nondecreasing if $f(S) \le f(T)$, for all $S \subseteq T$.

3.1. Attacker's Problem

Consider a known fortification decision, and define $\Theta(S)$ as the expected transportation costs when facilities in S are attacked. Next, we show that $\Theta(S)$ is supermodular.

Proposition 1. *We have that $\Theta(S)$ is nondecreasing.*

Proof. It is sufficient to show that $\Theta(S)$ is nondecreasing if $\Theta(S \cup \{e\}) \ge \Theta(S)$ for any $S \in J^+, e \notin S$. Let $\Gamma_i(S)$ be a set function corresponding to the expected

transportation costs for customer i, $\Gamma_i(S) = \sum_{l=1}^{|J^+|} \beta_i^l c_i^l$, where β_i^l is the probability that customer i is served by its lth closest facility when the attacked set is S; thus, $\Theta(S) = \sum_{i\in I} \Gamma_i(S)$. It is sufficient to show that $\Gamma_i(S)$ is nondecreasing for any i.

Without loss of generality, assume that facility e is the kth closest facility of the customer i, let p_e be the probability that facility e fails when e is not attacked, and let 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 that

$$\Gamma_i(S\cup\{e\}) = \sum_{l=1}^{k-1} \beta_i^l c_i^l + \frac{1-p'_e}{1-p_e}\beta_i^k c_i^k + \frac{p'_e}{p_e}\sum_{l=k+1}^{|J^+|} \beta_i^l c_i^l.$$

The first component represents the transportation costs for demand that is satisfied by facilities that are closer than e. Since attacking e does not change the probabilities of failure for facilities that are closer than e, the term is unchanged. The second term presents the costs for demand that are satisfied by e, that is, the kth closest facility. The coefficient $\frac{1-p'_e}{1-p_e}$ updates the probability of serving demand with e when its failure probability changes from p_e to p'_e. The last term calculates the transportation costs for demands that are served by further facilities.

Thus,

$$\begin{aligned}\rho_e^{\Gamma_i}(S) = \Gamma_i(S\cup\{e\}) - \Gamma_i(S) &= \frac{p_e-p'_e}{1-p_e}\beta_i^k c_i^k + \frac{p'_e-p_e}{p_e}\sum_{l=k+1}^{|J^+|}\beta_i^l c_i^l \\ &\geq c_i^k\left(\frac{p_e-p'_e}{1-p_e}\beta_i^k + \frac{p'_e-p_e}{p_e}\sum_{l=k+1}^{|J^+|}\beta_i^l\right) \\ &= c_i^k\left(\frac{p_e-p'_e}{1-p_e}\beta_i^k + \frac{p'_e-p_e}{p_e}\times\frac{p_e}{1-p_e}\times\beta_i^k\right) \\ &= 0.\end{aligned}$$

The second line holds because, by the definition of c_i^k, we have that $c_i^1 \leq c_i^2 \ldots \leq c_i^{|J^+|}$. Because the probablity of not using the first k-closest facilities to serve customer i, $\sum_{l=k+1}^{|J^+|}\beta_i^l$, is the same as the probability that all first k-closest facilities to customer i fail, $\prod_{l=1}^{k} p_{i_l}$, we have that

$$\sum_{l=k+1}^{|J^+|}\beta_i^l = \prod_{l=1}^{k} p_{i_l} = \frac{p_e}{1-p_e}\beta_i^k, \tag{15}$$

where the second equation is from (4) This completes the proof. □

Proposition 2. *We have that $\Theta(S)$ is supermodular.*

Proof. We prove by using the setting of the proof of Proposition 1. Consider another set T, $T = S\cup\{\sigma\}, \sigma \neq e$, $\sigma, e \notin S$, where σ is the uth closest facility. Similarly, we have the following:

$$\Gamma_i(T) = \sum_{l=1}^{u-1}\beta_i^l c_i^l + \frac{1-p'_\sigma}{1-p_\sigma}\beta_i^u c_i^u + \frac{p'_\sigma}{p_\sigma}\sum_{l=u+1}^{|J^+|}\beta_i^l c_i^l.$$

In case 1, $u < k$:

$$\Gamma_i(T\cup\{e\}) = \sum_{l=1}^{u-1}\beta_i^l c_i^l + \frac{1-p'_\sigma}{1-p_\sigma}\beta_i^u c_i^u + \frac{p'_\sigma}{p_\sigma}\sum_{l=u+1}^{k-1}\beta_i^l c_i^l + \frac{p'_\sigma}{p_\sigma}\times\frac{p'_e}{p_e}\sum_{l=k+1}^{|J^+|}\beta_i^l c_i^l,$$

$$\rho_e^{\Gamma_i}(T) = \Gamma_i(T\cup\{e\}) - \Gamma_i(T) = \frac{p'_\sigma}{p_\sigma}\left(\frac{p_e-p'_e}{1-p_e}\beta_i^k c_i^k + \frac{p'_e-p_e}{p_e}\sum_{l=k+1}^{|J^+|}\beta_i^l c_i^l\right).$$

Recall that

$$\begin{aligned}\rho_e^{\Gamma_i}(S) &= \Gamma_i(S\cup\{e\}) - \Gamma_i(S) \\ &= \frac{p_e-p'_e}{1-p_e}\beta_i^k c_i^k + \frac{p'_e-p_e}{p_e}\sum_{l=k+1}^{|J^+|}\beta_i^l c_i^l.\end{aligned}$$

We have that

$$\rho_e^{\Gamma_i}(T) = \frac{p'_\sigma}{p_\sigma}\rho_e^{\Gamma_i}(S) \geq \rho_e^{\Gamma_i}(S).$$

In case 2, $u > k$:

$$\begin{aligned}\Gamma_i(T\cup\{e\}) = &\sum_{l=1}^{k-1}\beta_i^l c_i^l + \frac{1-p'_e}{1-p_e}\beta_i^k c_i^k + \frac{p'_e}{p_e}\sum_{l=k+1}^{u-1}\beta_i^l c_i^l + \frac{p'_e}{p_e} \\ &\times\frac{1-p'_\sigma}{1-p_\sigma}\beta_i^u c_i^u + \frac{p'_e}{p_e}\times\frac{p'_\sigma}{p_\sigma}\times\sum_{l=u+1}^{|J^+|}\beta_i^l c_i^l\end{aligned}$$

$$\begin{aligned}\rho_e^{\Gamma_i}(T) = \Gamma_i(T\cup\{e\}) - \Gamma_i(T) = &\frac{p_e-p'_e}{1-p_e}\beta_i^k c_i^k \\ &+\frac{p'_e-p_e}{p_e}\sum_{l=k+1}^{u-1}\beta_i^l c_i^l + \frac{p'_e-p_e}{p_e}\times\frac{1-p'_\sigma}{1-p_\sigma}\beta_i^u c_i^u + \frac{p'_e-p_e}{p_e}\times\frac{p_\sigma}{p_\sigma}\sum_{l=u+1}^{|J^+|}\beta_i^l c_i^l.\end{aligned}$$

Thus,

$$\begin{aligned}\rho_e^{\Gamma_i}(T) - \rho_e^{\Gamma_i}(S) &= \frac{p'_e-p_e}{p_e}\left(\frac{p_\sigma-p'_\sigma}{1-p_\sigma}\beta_i^u c_i^u + \frac{p'_\sigma-p_\sigma}{p_\sigma}\sum_{l=u+1}^{|J^+|}\beta_i^l c_i^l\right) \\ &\geq \frac{p'_e-p_e}{p_e}\left(\frac{p_\sigma-p'_\sigma}{1-p_\sigma}\beta_i^u c_i^u + c_i^u\times\frac{p'_\sigma-p_\sigma}{p_\sigma}\sum_{l=u+1}^{|J^+|}\beta_i^l\right), \\ &= \frac{p'_e-p_e}{p_e}\times c_i^u\left(\frac{p_\sigma-p'_\sigma}{1-p_\sigma}\beta_i^u + \frac{p'_\sigma-p_\sigma}{p_\sigma}\times\frac{p_\sigma}{p'_\sigma-p_\sigma}\right) \\ &= 0.\end{aligned}$$

The inequality holds because $c_i^u \leq c_i^l$ when $l > u$ by definition.

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 $V = T \setminus S = \{v_1, v_2, \ldots, v_n\}$. We have that

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

Summing the inequalities, we have $\rho_e^{\Gamma_i}(T) \geq \rho_e^{\Gamma_i}(S)$ for any $S \subset T$ and $e \notin T$. Since $\rho_e^{\Gamma}(T) = \sum_{i\in I}\rho_e^{\Gamma_i}(T)$, according to the second condition in Definition 1, we complete the proof. □

3.2. 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 attacker's problem when Z is the fortification set.

Proposition 3. *We have that $\Psi(Z)$ is nonincreasing.*

Proposition 3 holds, since attacking an additional facility increases the chance of the facility being disrupted, and the demand it originally serves has to be routed to another facility that is no closer than the one being attacked. Thus, the overall cost is nondecreasing.

Proposition 4. *We have that $\rho_e^{\Psi}(Z) = 0$, 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, $\Psi(Z \cup \{e\}) = \Psi(Z)$; that is, $\rho_e(Z) = 0$. □

The same observation is made by Scaparra and Church (2008a) for RIMF, and it immediately implies Proposition 5; the tree-search algorithm (Scaparra and Church 2008a) is also applicable to solve this problem.

Proposition 5. *There exists an optimal solution Z^* in which at least one j exists such that $j \in \mathbb{I}(\emptyset)$ and $j \in Z^*$.*

Proposition 6. *We have that Ψ is neither supermodular nor submodular.*

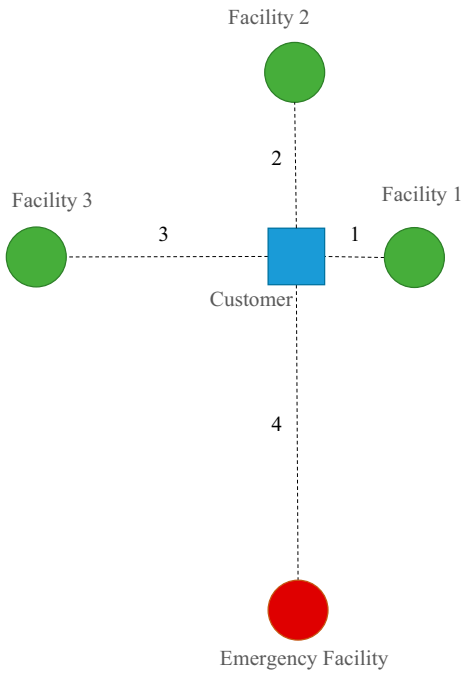
Proof. We prove this by providing two counterexamples. In the first example, we show a case where it is not supermodular. Then, we use another example to show that it is not submodular.

Counterexample 1: Consider an instance with one customer, three facilities, and an emergency facility, as shown in Figure 2; that is, $c_{ij} = j$ 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 failure $p_j = 0.5$; otherwise, it fails with certainty. Considering facility 2 as e, it is easy to compute Table 1. Because $r = |J|$, nonemergency facilities are always attacked in all scenarios. Because $\rho_e^{\Psi}(\emptyset) < \rho_e^{\Psi}(\{1\}) < \rho_e^{\Psi}(\{1,3\})$, Ψ violates the definition of a supermodular function.

Counterexample 2: Consider an instance with the same structure as Example 1. However, we assume that there is only one attack unit; that is, $r = 1$. Let $w = 0$ and $q_j = 0.5$ for $j = 1, 2, 3$. In this case, if a facility is fortified, then it is immune to the attack, whereas it fails at a probability of 0.5, even if no attack is launched at that facility. For this case, we have Table 2. In this example, the closest facility that is not fortified is attacked.

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

Figure 2. (Color online) An Instance of the Fortification Problem to Show That ψ Is Neither Supermodular nor Submodular



4. Solution Methods

4.1. For the Attacker's Problem

We have shown that the AP, as defined in Equations (6)–(8), is equivalent to minimizing a nonincreasing submodular function with cardinality constraints.

Minimizing a general submodular function without side constraints can be solved in strong polynomial time (Iwata and Orlin 2009, Orlin 2009). The most theoretically efficient algorithm is the one developed by Orlin (2009), which runs in $O(n^5EO + n^6)$, where n is the size of the ground set and EO is the time to evaluate the function for a given subset. 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 2005). To the best of our knowledge, no combinatorial method has been developed for minimizing a monotone, nonincreasing, submodular function with cardinality or knapsack constraints.

We adopt an exact cutting-plane algorithm proposed by Atamtürk and Narayanan (2008) to solve the AP, since we know that the AP is submodular. A similar approach has been developed for solving a supply chain design problem where a cost term appears to be

Table 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

Table 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

submodular (Wu and Zhang 2014). The cutting-plane algorithm iteratively solves an MIP master problem and subproblems. The MIP master problem can easily include additional side constraints such as the r-interdiction constraint in this study.

For completeness, we briefly describe the approach that is mostly adopted from Atamtürk and Narayanan (2008) in the context of our AP problem. To be consistent with the terminology used in the literature, instead of maximizing a supermodular function, we reformulate the problem to minimize a submodular function. Define a 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

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

Definition 3. Given a submodular function f with $f(\emptyset)=0$, the submodular polyhedron $P(f)$ is defined as $P(f)=\{v: v\in\mathbb{R}^n,\ \forall S\subseteq J, v(S)\le f(S)\}$, 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. Define $Q_f=\text{conv}\{(s,t)\in\{0,1\}^n\times\mathbb{R}: f(s)\le t\}$, where Q_f is the convex lower envelope for the submodular function f.

Theorem 1. *The inequality $t\ge\sum_{i\in J}v_i s_i$ is a valid inequality for Q_f if and only if $v\in P(f)$ (Atamtürk and Narayanan 2008).*

The inequality defined in Theorem 1 is called an *extended polymatroid inequality*. Atamtürk and Narayanan (2008) showed that, for a given a solution $(\bar{s},\bar{t})$, one can check whether such a solution is in Q_f and if not, find an extended polymatroid inequality to cut off the infeasible solution by solving the following problem with the greedy algorithm (Edmonds 1970):

$$t^*=\max\left\{\sum_{i\in J}v_i\bar{s}_i : v\in P(f)\right\}.$$

The greedy algorithm is detailed as follows:

Step 1. For a given $\bar{s}$, sort elements in J in the form of $l_1,l_2,\ldots,l_n$ such that $\bar{s}_{l_1}\ge\bar{s}_{l_2}\ge\ldots\ge\bar{s}_{l_n}$.

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

Because s_i only takes a value of 0 or 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 $t^*=\sum_{i\in J}\bar{v}_i\bar{s}_i=f(\bar{S})$, where $\bar{S}$ is the set corresponding to $\bar{s}_i$. If $\bar{t}\ge t^*$, then no extended polymatroid inequality is violated and we have found the optimal solution. Otherwise, we find a cut in the following form:

$$t\ge\bar{v}_1 s_1+\bar{v}_2 s_2+\ldots+\bar{v}_n s_n.$$

The master problem (AP-master) is formulated as

$$\min t \tag{16}$$

$$\text{s.t.}\quad t\ge\bar{v_1}^k s_1+\bar{v_2}^k s_2+\ldots+\bar{v_n}^k s_n \quad \forall k\in K \tag{17}$$

$$\sum_{j\in J^+}s_j\le r$$

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

where K is the collection of cuts from previous iterations.

This iterative procedure might be inefficient for large instances, since, in each iteration, an integer programming problem must be solved. However, commercial MIP solvers like CPLEX provide a lazy-constraints feature that can be used to implement the cutting-plane algorithm much more efficiently.

4.2. For the Fortification Problem

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

4.2.1. A Logic-Based Benders Decomposition. The logic-based Benders decomposition (LBBD) method introduced by Hooker and Ottosson (2003) is a generalization of the Benders decomposition. Like the Benders decomposition, LBBD decomposes the original problem into a master problem and a subproblem or multiple subproblems with corresponding variables denoted by x and y. The master problem is solved to obtain a solution for variables x, and then subproblems are solved for y given the fixed x values. For some recent developments and a full review of the LBBD, refer to Angulo et al. (2016), Rebennack (2016), Rahmaniani et al. (2017), and Soares et al. (2017).

The cuts in the Benders decomposition are based on linear duality, which requires a subproblem to be a linear programming problem, whereas a subproblem of LBBD can be in any form of mathematical program. The *inference duals* are used instead of the linear duals. Because of this, there is no standard algorithm to generate cuts for LBBD, and cuts must be derived based on knowledge of the underlying problem. The

cutting-plane methods that we proposed in the previous section are actually a special case of LBBD, where the *inference duals* are readily provided by the supermodular property of the attack problem.

The efficiency of LBBD depends on how the master problem and subproblem are defined and is highly affected by the tightness of the cuts. The two-stage decision-making nature of the fortification and attack problems and the inference duals provided by the supermodular property of the attack problem motivated us to adopt LBBD. Its efficiency was also well justified by numerical experiment results in Section 5.2. We can write the master problem of FP as

$$\min \eta \tag{18}$$

$$\text{s.t.} \quad \text{logic-based Benders cuts} \tag{19}$$

$$\sum_{j\in J} z_j \le h$$

$$z_j \in \{0,1\}, \tag{20}$$

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 from the AP subproblem.

Given a solution $(\bar{z},\bar{\eta})$ to the LBDD master problem, we compute the transportation costs by solving an AP, and we let $\eta^*(\bar{z}) = F(\bar{z})$. If $\eta^*(\bar{z}) > \bar{\eta}$, 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 numbers of solutions, Chu and Xia (2004) showed 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 $(\bar{z},\bar{\eta})$ is infeasible, then the newly generated cut must exclude at least $(\bar{z},\bar{\eta})$.

2. Any feasible solution $(\bar{z},\bar{\eta})$ satisfies all cuts.

Clearly, the cut defined by (21) satisfies both conditions and therefore is valid.

Theorem 2. *Cuts in the form of* (21) *guarantee that LBBD finds an optimal solution in finite iterations:*

$$\eta \ge \eta^*(\bar{z}) - \eta^*(\bar{z}) * \left(\sum_{j\in J|\bar{z}_j=0} z_j + \sum_{j\in J|\bar{z}_j=1} (1-z_j) \right). \tag{21}$$

Proof. Clearly, the cut satisfies condition 1. If the incumbent solution $(\bar{z},\bar{\eta})$ is infeasible, that is, $\bar{\eta} < \eta^*(\bar{z})$, then the cut ensures that the solution is cut off by requiring $\eta \ge \eta^*(\bar{z})$ for solution $\bar{z}$. For any solutions other than $\bar{z}$, the cut is inactive. Thus, it does not eliminate any feasible solution. □

Obviously, the LBBD based on (21) is inefficient, since it is not better than simply enumerating all the possible combinations of z, that is, all subsets of J in the worst case. Now consider a family of cuts in the following form:

$$\eta \ge \eta^*(\bar{z}) + \sum_{j\in J|\bar{z}_j=0} \Delta_j^{\bar{z}} z_j + \sum_{j\in J|\bar{z}_j=1} \nabla_j^{\bar{z}} (1-z_j), \tag{22}$$

where $\Delta_j^{\bar{z}}$ is a coefficient used to capture the change of transportation costs from solution $\bar{z}$ by fortifying facility j and $\nabla_j^{\bar{z}}$ is used to capture the change of transportation costs when the defense resource is removed from facility j. Clearly, $\Delta_j^{\bar{z}} \le 0$ and $\nabla_j^{\bar{z}} \ge 0$. When we set $\Delta_j^{\bar{z}} = -\eta^*(\bar{z})$, for j whose $\bar{z}_j = 0$, and $\nabla_j^{\bar{z}} = 0$, for j whose $\bar{z}_j = 1$, it is easy to verify that it is a valid cut and is at least as tight as (21). For simplicity, we let $\nabla_j^{\bar{z}} = 0$ and focus on how to tighten $\Delta_j^{\bar{z}}$ to have the following inequality:

$$\eta \ge \eta^*(\bar{z}) + \sum_{j\in J|\bar{z}_j=0} \Delta_j^{\bar{z}} z_j. \tag{23}$$

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

Theorem 3. *Cuts in the form of* (23) *with* $\Delta_j^{\bar{z}} = \widehat{\Delta_j^{\bar{z}}}$ *are valid, where*

$$\widehat{\Delta_j^{\bar{z}}} = \min_{Z\subseteq T\subseteq J,} \{\Psi(T\cup\{j\}) - \Psi(T)\} \text{ for } j\in J, j\notin Z. \tag{24}$$

Proof. Clearly the cut eliminates the infeasible incumbent solution. We next show that it does not eliminate any feasible solution. Let $T_j^*(\bar{z}) = \arg\min_{Z\subseteq T\subseteq J} \{\Psi(T\cup\{j\}) - \Psi(T)\}$ for $j\in J, j\notin \bar{Z}$. Consider a cut generated with an incumbent solution $\bar{z}$ and another solution $\bar{z}'$ such that $\bar{Z}\subseteq\bar{Z}'$, where Z' is the set of fortified facilities for solution $\bar{z}'$. Let $Z_\Delta = Z'\setminus Z$. Consider an arbitrary order of Z_Δ, and let $Z_\Delta = \{l_1, l_2, \ldots, l_{|Z_\Delta|}\}$. Define $L^i = \bar{Z}\cup\{l_1, l_2, \ldots, l_i\}$. We have that

$$\Psi(Z') = \Psi(\bar{Z}) + \sum_{i=1}^{|Z_\Delta|} \rho_{l_i}^{\Psi}(L^{i-1}).$$

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

$$\Psi(Z') \ge \eta^* + \sum_{j\in J|\bar{z}_j=0} \widehat{\Delta_j^{\bar{z}}} z_j.$$

Thus, the cut in (24) does not eliminate solution Z' for any $\bar{Z}\subsetneq Z'$. Because Ψ is a nonincreasing function regarding both set variables, if $\bar{Z}\not\subseteq Z'$, then the inequality still holds by adding missing elements in $\bar{Z}$ to Z'. Thus, the cut does not eliminate any feasible solution. □

However, computing $\widehat{\Delta_j^{\bar{z}}}$ is hard in general, because it calls the AP for $O(2^{|J-\bar{Z}|})$ times. Next, we propose a family of cuts that can be computed in $|J-\bar{Z}|$ calls to the AP. The basic idea is to find an approximation to the lower bound of $\widehat{\Delta_j^{\bar{z}}}$.

Because of Proposition 4, $T_j^*(\bar{z})$ must satisfy $j \in \mathbb{I}(T_j^*(\bar{z}))$. The following cut is motivated to force j to be included in $\mathbb{I}(T_j^*(\bar{z}))$.

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

$$s_j \geq 1. \tag{25}$$

Let Ψ_j denote the set function for the corresponding fortification problem with facility j being attacked. We always have $j \in \mathbb{I}(T_j^*(\bar{z}))$ for $j \notin \bar{Z}$. We have the following approximation for $\Delta_j^{\bar{z}}$:

$$\widetilde{\Delta_j^{\bar{z}}} = \Psi_j(\bar{Z} \cup \{j\}) - \Psi_j(\bar{Z}) \text{ for } j \in J, j \notin \bar{Z}. \tag{26}$$

To compute each $\widetilde{\Delta_j^{\bar{z}}}$, we need to solve two APs. Because we only want to find a lower-bound approximation for the coefficient of $\Delta_j^{\bar{z}}$, we can first compute $\Psi_j(\bar{Z})$ to find the corresponding interdiction set $\mathbb{I}(\bar{Z})$ that includes j and then compute a lower bound of $\Psi_j(\bar{Z} \cup \{j\})$ by fixing the interdiction solution to $\mathbb{I}(\bar{Z})$. The LBBD cuts become (23) with $\Delta_j^{\bar{z}} = \widetilde{\Delta_j^{\bar{z}}}$ for a given incumbent solution $\bar{z}$.

Motivated by Proposition 5, we can enhance the FP master problem by the following cut. We first solve $\Psi(\emptyset)$ and obtain $\mathbb{I}(\emptyset)$. Cut (27) ensures that the optimal fortification decision contains at least one j such that $j \in \mathbb{I}(\emptyset)$:

$$\sum_{j \in \mathbb{I}(\emptyset)} z_j \geq 1. \tag{27}$$

5. Computational Study

In this section, we tested the performance of the proposed algorithms. We implemented the algorithms in Scala and solved all of the problem instances on a Dell OptiPlex 9010 with one Intel 3.40-gigahertz CPU and four gigabytes of memory running the Ubuntu operating system. The MIP problems were solved using the ILOG CPLEX Academic Initiative Edition version 12.6. Both algorithms outlined in Sections 4.1 and 4.2 are implemented using the lazy-constraints feature provided by CPLEX. The U.S. data set, which is based on 2000 census data and contains the largest cities in the United States, was used to test the algorithms' performance. The U.S. data set has been used in related works (Cui et al. 2010, Shen et al. 2011, Aboolian et al. 2013). The exogenous probabilistic disruption rate was computed as $q_j = \alpha e^{-D_j/\theta}$, where D_j is the distance from the location to New Orleans. Note that, in this data set, a candidate facility site is located at one of the demand points, which is not necessarily true in all applications. The data instances and source code in this paper can be found on the GitHub repository (Zhang et al. 2021).

5.1. Solving the Attacker's Problem

Because there is no solution method other than enumeration available in the literature, we compared the proposed solution methods, the linear model proposed in Section 2.1 and the cutting-plane method outlined in Section 4.1, with enumeration. The algorithms were tested with instances ranging from 50 to 150 demand points. Thus, we first solved a p-median problem to select k facilities. We assumed that no facility was fortified, and we tested three values for the number of facilities attacked r, that is, 3, 6, and 9, which covered the range of r values used in most related works. The results are summarized in Table 3. The column titled "nbEO" reports the number of calls to evaluate a solution (z, s), that is, to compute the objective function in (6). For the enumeration method, the number of calls equals the combination number $\binom{k}{r}$. It is not surprising that the cutting-plane algorithm and the linear model always find an optimal solution. Although the solution time for enumeration increases exponentially with respect to k and r, the increase is very slight for the cutting-plane algorithm. It can also be observed that the cutting-plane algorithm shows advantages in solution time for larger instances. Therefore, we adopt the cutting-plane method for solving the attacker's problem in the remaining studies. Numerical examples to illustrate the cutting-plane algorithms can be found in the online supplement.

We also tested the cutting-plane method's performance by varying parameters that affected the disruption probabilities, such as w, α, and $\Theta(S)$. Three combinations of α and $\Theta(S)$ were tested to simulate low, moderate, and high probabilistic disruption risks: $(0.1, 200)$, $(0.2, 400)$, and $(0.3, 800)$, respectively. We randomly fortified $\lfloor \frac{k}{2} \rfloor$ facilities. The results are reported in Tables 4 and 5. The columns titled "nbCuts" report the number of cuts in the form of (17) added to the master problem. From the results, we observed that, when w increased, the objective value increased, and the solution time and number of cuts generated increased in general. Recall that w is the probability that an attack will succeed when a facility is fortified. The objective value increases with the probabilistic disruption risk factors α and $\Theta(S)$. However, no obvious pattern was observed in terms of how these factors affected the solution time and the number of cuts generated.

5.2. Solving the Fortification Problem

Two methods in the literature can be adapted to solve the FP: the tree-search implicit enumeration method proposed by Scaparra and Church (2008a), which is an exact algorithm to solve the FP; and the greedy-search heuristic algorithm proposed by Zhu et al. (2013). The proposed LBBD was compared with

Table 3. Cutting-Plane Algorithm's Performance on Solving the AP

			Enumeration			Cutting plane			Linear model	
$\|I\|$	k	r	Obj.	Time	nbEO	$\Delta_{Obj.}$	Time	nbEO	$\Delta_{Obj.}$	Time
50	15	3	1,101,845.24	0.14	455	0	0.29	374	0	0.09
50	15	6	1,976,813.06	0.55	5,005	0	0.11	740	0	0.22
50	15	9	3,240,988.49	0.42	5,005	0	0.11	969	0	0.29
50	20	3	792,317.18	0.07	1,140	0	0.07	552	0	0.13
50	20	6	1,455,117.90	1.70	38,760	0	0.14	1,503	0	0.19
50	20	9	2,202,803.09	7.44	167,960	0	0.20	2,043	0	0.76
50	30	3	431,071.16	0.29	4,060	0	0.19	1,774	0	0.18
50	30	6	845,343.21	41.89	593,775	0	0.48	4,224	0	0.83
50	30	9	1,423,410.35	1,012.98	14,307,150	0	0.66	5,002	0	1.34
75	15	3	1,265,758.87	0.03	455	0	0.03	339	0	0.07
75	15	6	2,303,228.54	0.24	5,005	0	0.06	805	0	0.41
75	15	9	3,246,383.92	0.24	5,005	0	0.09	966	0	0.96
75	20	3	897,856.49	0.08	1,140	0	0.06	656	0	0.22
75	20	6	1,732,421.59	2.53	38,760	0	0.15	1,665	0	0.47
75	20	9	2,355,708.23	11.04	167,960	0	0.25	2,109	0	2.07
75	30	3	514,184.54	0.40	4,060	0	0.25	2,177	0	0.30
75	30	6	1,132,851.13	59.78	593,775	0	0.49	3,700	0	0.49
75	30	9	1,707,084.16	1,443.23	14,307,150	0	1.18	7,727	0	2.95
100	15	3	1,372,013.35	0.03	455	0	0.05	548	0	0.11
100	15	6	2,502,580.27	0.30	5,005	0	0.12	1,272	0	0.56
100	15	9	3,534,156.55	0.30	5,005	0	0.13	1,245	0	1.61
100	20	3	1,002,426.94	0.10	1,140	0	0.09	825	0	0.26
100	20	6	1,918,525.84	3.30	38,760	0	0.22	1,942	0	0.73
100	20	9	2,558,151.56	14.32	167,960	0	0.34	2,198	0	2.84
100	30	3	593,566.69	0.51	4,060	0	0.33	2,273	0	0.40
100	30	6	1,257,326.92	75.66	593,775	0	0.65	4,194	0	0.63
100	30	9	1,914,434.62	1,825.49	14,307,150	0	1.25	6,927	0	4.66
150	15	3	1,546,415.88	0.05	455	0	0.05	372	0	0.21
150	15	6	2,698,907.14	0.50	5,005	0	0.14	1,079	0	1.57
150	15	9	3,977,272.72	0.50	5,005	0	0.18	1,254	0	3.33
150	20	3	1,187,705.98	0.15	1,140	0	0.14	762	0	0.39
150	20	6	2,234,857.77	5.01	38,760	0	0.23	1,454	0	0.83
150	20	9	2,887,852.42	21.86	167,960	0	0.66	3,559	0	5.33
150	30	3	697,791.79	0.75	4,060	0	0.55	2,672	0	0.74
150	30	6	1,465,985.90	109.53	593,775	0	0.87	3,913	0	1.11
150	30	9	2,219,730.26	2,670.02	14,307,150	0	1.67	7,013	0	11.45

these two algorithms, and the results are reported in Tables 6 and 7.

The column titled "nbAPs" reports the number of calls required to solve an AP. All of the APs were solved by the cutting-plane method. The column titled "$\Delta_{Obj.}$" reports the difference between the objective value obtained by the relevant algorithm and the objective value from the tree-search algorithm. The algorithms were set with a time limit of 3,600 seconds. Because the tree search needs to explore up to $(r^{h+1} - 1)/(r-1)$ number of nodes in its tree structure (Scaparra and Church 2008a), as expected, we observed that the tree-search algorithm was very sensitive to parameters h and r. The greedy algorithm was the fastest among the three methods and the least sensitive to increases in parameters h and r. However, about 30% of the solutions were not optimal. The LBBD 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 the tree search finishes within a given time limit, that is, the solution is proved to be optimal, the LBBD always finds a solution with the same objective value. The LBBD finds even better solutions for three instances when the tree search terminates due to the time limit. The LBBD outperforms the greedy algorithm with an up-to 13.2% improvement in the total cost, as shown in Tables 6 and 7.

We also test our algorithm on special cases that the proposed model is generalizing. We set $q_j = 0$, $j \in J$, to generate computational instances of RIMF-p and report the computational results in Table 8 with $w = 0.4$. In addition, we set $q = 0$ and $w = 0$ to generate computational instances of RIMF and report results in Table 9. From both Tables 8 and 9, we can make similar observations to Tables 6 and 7 that the LBBD

outperforms the benchmark methods, especially when h and r are large. This shows that our algorithm is still efficient when it is applied to solve RIMF and RIMF-p, which are the special cases of our problems. It is also interesting to note that the advantage of the LBBD is salient for solving RIMF instances, that is, $q_j = 0$ and $w = 0$. Specifically, as we can see from Table 9 that, for all the instances, the LBBD produces the same

Table 4. Algorithm's Performance on Different Parameters (Small Instances)

					$w = 0.4$				$w = 0.6$				$w = 0.8$			
$\|I\|$	k	r	α	θ	Obj.	Time	nbCuts	nbEO	Obj.	Time	nbCuts	nbEO	Obj.	Time	nbCuts	nbEO
50	15	3	0.1	200	646,491.81	0.21	11	180	792,218.03	0.10	14	231	937,944.26	0.08	18	294
50	15	3	0.2	400	668,362.21	0.06	14	227	812,856.55	0.05	18	292	957,350.90	0.06	28	454
50	15	3	0.3	800	750,129.29	0.03	11	180	887,448.01	0.03	15	242	1,024,766.74	0.07	31	500
50	15	6	0.1	200	964,558.04	0.04	16	259	1,216,939.34	0.06	28	455	1,544,510.95	0.06	28	456
50	15	6	0.2	400	991,650.28	0.04	18	290	1,247,395.32	0.07	37	595	1,575,842.61	0.09	35	567
50	15	6	0.3	800	1,084,601.75	0.03	20	325	1,346,233.23	0.06	38	616	1,673,069.58	0.08	51	822
50	15	9	0.1	200	1,223,619.83	0.03	12	198	1,551,609.28	0.07	32	518	2,186,703.53	0.05	25	407
50	15	9	0.2	400	1,254,648.24	0.03	15	245	1,577,136.36	0.10	45	729	2,235,828.91	0.09	37	601
50	15	9	0.3	800	1,364,131.70	0.04	17	278	1,717,114.39	0.14	56	908	2,373,736.64	0.10	68	1,096
50	20	3	0.1	200	467,394.61	0.02	6	127	568,948.23	0.03	12	257	670,501.84	0.02	14	298
50	20	3	0.2	400	487,809.61	0.01	6	127	589,312.13	0.02	12	256	690,814.66	0.03	19	404
50	20	3	0.3	800	557,381.67	0.02	8	169	655,578.85	0.04	21	446	753,776.03	0.05	30	634
50	20	6	0.1	200	649,288.37	0.03	20	425	854,882.36	0.04	20	427	1,116,002.03	0.05	29	615
50	20	6	0.2	400	670,232.07	0.04	22	469	879,303.35	0.04	24	510	1,142,008.75	0.09	50	1,057
50	20	6	0.3	800	748,289.64	0.02	16	342	961,221.39	0.07	46	973	1,218,942.84	0.09	64	1,354
50	20	9	0.1	200	816,174.32	0.02	14	298	1,139,635.64	0.05	35	740	1,528,079.96	0.07	39	825
50	20	9	0.2	400	841,676.76	0.03	20	425	1,166,018.29	0.04	24	509	1,555,371.82	0.10	49	1,036
50	20	9	0.3	800	929,345.70	0.04	26	553	1,251,554.55	0.07	42	889	1,645,403.09	0.21	119	2,514
50	30	3	0.1	200	197,745.54	0.05	12	375	234,906.98	0.09	22	687	307,495.06	0.09	33	1,029
50	30	3	0.2	400	214,197.71	0.04	13	406	251,285.12	0.06	22	686	323,823.76	0.13	48	1,495
50	30	3	0.3	800	266,417.92	0.04	14	436	303,305.38	0.05	19	593	375,245.69	0.14	50	1,557
50	30	6	0.1	200	366,943.59	0.05	17	532	466,957.56	0.08	27	842	580,947.81	0.18	60	1,865
50	30	6	0.2	400	385,218.63	0.05	17	531	485,931.33	0.08	27	844	600,128.88	0.19	62	1,927
50	30	6	0.3	800	444,770.54	0.07	20	627	545,229.68	0.08	29	904	656,731.60	0.21	68	2,114
50	30	9	0.1	200	458,843.48	0.07	25	780	644,215.57	0.15	52	1,619	878,623.87	0.23	69	2,148
50	30	9	0.2	400	477,460.29	0.14	48	1,495	662,224.00	0.11	37	1,151	895,736.85	0.38	116	3,605
50	30	9	0.3	800	540,754.61	0.11	40	1,244	722,042.89	0.14	45	1,399	948,530.98	0.39	112	3,478
75	15	3	0.1	200	841,154.86	0.02	13	210	934,062.69	0.02	17	279	1,083,869.38	0.03	28	452
75	15	3	0.2	400	865,100.88	0.02	14	226	958,201.38	0.02	20	326	1,109,851.56	0.03	25	407
75	15	3	0.3	800	943,327.40	0.02	14	229	1,045,096.59	0.03	23	375	1,193,996.58	0.04	32	518
75	15	6	0.1	200	1,279,956.32	0.02	11	179	1,518,375.05	0.03	21	341	1,828,591.49	0.05	40	645
75	15	6	0.2	400	1,315,230.17	0.02	11	179	1,553,551.90	0.03	26	425	1,865,672.19	0.05	39	630
75	15	6	0.3	800	1,426,897.78	0.02	16	260	1,660,920.48	0.03	25	407	1,966,862.85	0.05	41	664
75	15	9	0.1	200	1,644,592.67	0.02	17	276	2,032,639.17	0.03	23	373	2,502,527.64	0.04	30	486
75	15	9	0.2	400	1,686,103.69	0.02	16	262	2,070,857.62	0.04	31	506	2,540,367.20	0.08	55	889
75	15	9	0.3	800	1,801,648.37	0.02	19	311	2,183,489.78	0.05	33	539	2,643,466.74	0.11	75	1,213
75	20	3	0.1	200	543,153.23	0.02	8	170	650,409.41	0.03	12	257	757,665.59	0.04	18	384
75	20	3	0.2	400	567,202.54	0.02	7	148	673,835.08	0.03	14	296	780,467.61	0.05	26	554
75	20	3	0.3	800	647,595.74	0.02	10	213	749,665.85	0.03	17	361	851,735.95	0.05	27	570
75	20	6	0.1	200	753,486.52	0.04	21	445	975,119.92	0.07	32	682	1,290,260.77	0.07	32	680
75	20	6	0.2	400	780,615.24	0.04	17	362	1,004,593.15	0.08	32	678	1,318,672.77	0.15	76	1,605
75	20	6	0.3	800	868,683.57	0.04	21	443	1,096,677.77	0.09	44	932	1,401,777.70	0.12	51	1,078
75	20	9	0.1	200	946,363.40	0.05	24	509	1,295,043.67	0.09	44	932	1,749,010.59	0.12	56	1,189
75	20	9	0.2	400	973,857.52	0.06	27	578	1,325,374.67	0.09	38	804	1,778,576.55	0.13	58	1,226
75	20	9	0.3	800	1,070,560.97	0.09	48	1,018	1,421,409.38	0.10	46	972	1,865,410.36	0.18	78	1,646
75	30	3	0.1	200	248,626.69	0.13	33	1,030	289,686.58	0.22	57	1,775	379,864.07	0.21	55	1,711
75	30	3	0.2	400	266,482.03	0.15	39	1,219	307,752.06	0.17	44	1,370	398,293.52	0.15	39	1,214
75	30	3	0.3	800	320,628.93	0.13	34	1,060	365,406.71	0.18	47	1,461	457,970.34	0.24	64	1,993
75	30	6	0.1	200	370,752.02	0.10	25	779	482,767.05	0.15	36	1,123	724,110.45	0.25	58	1,807
75	30	6	0.2	400	390,536.39	0.15	39	1,217	501,293.43	0.22	56	1,747	743,765.98	0.26	64	1,995
75	30	6	0.3	800	454,649.26	0.13	33	1,029	562,133.23	0.15	35	1,093	808,828.24	0.42	103	3,206
75	30	9	0.1	200	500,797.23	0.12	29	905	638,355.71	0.35	79	2,459	1,001,377.41	0.39	83	2,584
75	30	9	0.2	400	520,446.55	0.11	27	843	657,530.48	0.33	73	2,275	1,020,613.97	0.44	95	2,954
75	30	9	0.3	800	587,472.52	0.13	32	1,001	720,537.44	0.45	89	2,770	1,082,735.19	0.55	113	3,513

Table 5. Algorithm's Performance on Different Parameters (Large Instances)

					$w = 0.4$				$w = 0.6$				$w = 0.8$			
$\|I\|$	k	r	α	θ	Obj.	Time	nbCuts	nbEO	Obj.	Time	nbCuts	nbEO	Obj.	Time	nbCuts	nbEO
100	15	3	0.1	200	851,938.73	0.02	10	163	1,016,052.90	0.03	21	342	1,180,167	0.04	29	468
100	15	3	0.2	400	883,125.84	0.02	15	243	1,046,088.34	0.02	14	228	1,209,051	0.04	26	423
100	15	3	0.3	800	990,082.76	0.02	13	209	1,145,873.29	0.03	18	292	1,301,664	0.03	23	373
100	15	6	0.1	200	1,207,367.79	0.04	27	436	1,445,538.08	0.04	27	437	1,895,995	0.07	47	756
100	15	6	0.2	400	1,232,800.09	0.03	19	309	1,485,930.75	0.06	42	678	1,936,731	0.07	46	742
100	15	6	0.3	800	1,332,409.80	0.04	20	324	1,608,562.79	0.08	52	839	2,050,312	0.12	84	1,354
100	15	9	0.1	200	1,496,908.67	0.02	14	228	1,883,939.12	0.07	42	675	2,499,328	0.08	48	775
100	15	9	0.2	400	1,535,290.62	0.03	21	341	1,932,646.31	0.07	45	722	2,540,768	0.13	68	1,101
100	15	9	0.3	800	1,671,863.58	0.03	23	374	2,087,906.66	0.09	54	872	2,673,028	0.14	82	1,321
100	20	3	0.1	200	609,454.96	0.02	6	127	720,915.31	0.04	15	320	832,375.7	0.07	25	533
100	20	3	0.2	400	635,698.99	0.02	6	127	746,529.58	0.04	16	344	857,360.2	0.07	28	596
100	20	3	0.3	800	722,272.07	0.02	9	190	828,418.26	0.04	15	321	934,564.5	0.08	30	636
100	20	6	0.1	200	842,712.64	0.06	24	510	1,071,876.96	0.11	43	913	1,406,136	0.14	48	1,016
100	20	6	0.2	400	870,520.87	0.05	21	447	1,103,987.03	0.10	41	870	1,437,096	0.19	80	1,690
100	20	6	0.3	800	960,711.79	0.06	24	510	1,203,369.67	0.09	36	765	1,526,960	0.20	81	1,711
100	20	9	0.1	200	1,078,848.11	0.04	15	319	1,412,434.51	0.15	55	1,164	1,898,836	0.18	60	1,270
100	20	9	0.2	400	1,109,366.44	0.03	12	257	1,445,471.09	0.11	41	866	1,931,033	0.20	67	1,417
100	20	9	0.3	800	1,209,879.67	0.04	17	361	1,549,416.04	0.08	29	613	2,025,200	0.24	80	1,690
100	30	3	0.1	200	344,839.05	0.09	18	568	371,196.29	0.19	40	1,246	444,896.7	0.22	48	1,493
100	30	3	0.2	400	363,700.45	0.12	26	813	390,420.60	0.14	30	936	464,865.9	0.27	58	1,807
100	30	3	0.3	800	419,218.05	0.11	21	659	449,422.09	0.14	29	903	529,538.1	0.29	63	1,960
100	30	6	0.1	200	458,796.66	0.24	44	1,374	605,081.02	0.15	31	967	856,012.4	0.26	52	1,621
100	30	6	0.2	400	477,548.54	0.27	54	1,680	625,445.13	0.17	35	1,091	877,726.3	0.42	84	2,614
100	30	6	0.3	800	539,078.99	0.23	47	1,465	692,520.93	0.26	51	1,588	949,886.7	0.57	114	3,542
100	30	9	0.1	200	605,321.88	0.15	30	939	779,989.27	0.26	48	1,495	1,185,328	0.39	77	2,396
100	30	9	0.2	400	627,581.86	0.17	34	1,061	800,507.95	0.35	67	2,086	1,206,046	0.38	71	2,212
100	30	9	0.3	800	699,988.30	0.19	37	1,153	868,913.17	0.46	79	2,458	1,273,244	0.56	105	3,267
150	15	3	0.1	200	978,091.13	0.02	10	163	1,157,034.39	0.03	13	214	1,335,978	0.05	24	391
150	15	3	0.2	400	1,013,516.61	0.03	14	227	1,191,149.70	0.05	24	393	1,368,783	0.05	25	405
150	15	3	0.3	800	1,133,839.75	0.05	22	354	1,303,359.69	0.05	23	375	1,472,880	0.06	31	500
150	15	6	0.1	200	1,319,742.13	0.08	36	582	1,647,836.05	0.09	42	681	2,057,134	0.11	48	775
150	15	6	0.2	400	1,349,494.47	0.08	37	599	1,682,488.02	0.10	46	742	2,103,901	0.14	63	1,016
150	15	6	0.3	800	1,464,954.11	0.09	39	630	1,799,061.45	0.12	52	840	2,231,052	0.17	79	1,270
150	15	9	0.1	200	1,671,972.32	0.04	19	307	2,113,443.08	0.12	51	821	2,804,873	0.14	54	871
150	15	9	0.2	400	1,708,514.78	0.04	20	322	2,156,616.49	0.12	52	836	2,852,899	0.15	65	1,047
150	15	9	0.3	800	1,848,990.67	0.08	36	583	2,292,580.04	0.18	73	1,175	3,004,225	0.28	103	1,657
150	20	3	0.1	200	694,426.32	0.04	11	234	834,210.25	0.05	13	275	975,199.1	0.09	25	530
150	20	3	0.2	400	722,874.51	0.05	12	254	863,337.46	0.20	15	319	1,003,800	0.11	31	656
150	20	3	0.3	800	817,204.06	0.04	11	232	953,210.56	0.06	19	403	1,089,217	0.11	32	676
150	20	6	0.1	200	967,912.36	0.04	12	257	1,235,387.61	0.10	29	611	1,541,934	0.20	53	1,120
150	20	6	0.2	400	998,526.51	0.05	15	320	1,266,038.70	0.11	31	658	1,572,661	0.34	90	1,901
150	20	6	0.3	800	1,098,849.57	0.05	13	279	1,363,699.07	0.09	27	570	1,666,502	0.26	73	1,538
150	20	9	0.1	200	1,180,653.86	0.06	17	362	1,508,890.43	0.18	46	968	2,048,709	0.29	68	1,436
150	20	9	0.2	400	1,205,621.14	0.06	17	360	1,536,997.97	0.18	48	1,015	2,079,084	0.35	94	1,984
150	20	9	0.3	800	1,294,461.72	0.08	22	466	1,632,219.54	0.24	60	1,268	2,174,244	0.39	99	2,089
150	30	3	0.1	200	414,869.50	0.12	18	562	438,490.91	0.31	45	1,400	530,829.6	0.36	54	1,680
150	30	3	0.2	400	436,341.13	0.12	17	532	459,956.72	0.25	37	1,153	553,688.5	0.36	53	1,651
150	30	3	0.3	800	498,675.63	0.15	21	657	523,762.45	0.26	39	1,214	627,217.2	0.38	57	1,774
150	30	6	0.1	200	544,925.55	0.24	34	1,060	712,363.14	0.22	32	999	1,000,881	0.37	53	1,655
150	30	6	0.2	400	566,447.44	0.25	36	1,123	735,667.92	0.27	39	1,217	1,025,812	0.43	62	1,933
150	30	6	0.3	800	636,080.11	0.29	42	1,309	811,999.41	0.25	37	1,153	1,108,434	0.55	80	2,486
150	30	9	0.1	200	675,773.69	0.27	36	1,123	903,661.83	0.47	62	1,930	1,370,629	0.49	63	1,965
150	30	9	0.2	400	699,827.98	0.29	40	1,249	927,251.44	0.46	62	1,931	1,394,750	0.76	103	3,203
150	30	9	0.3	800	774,115.34	0.30	41	1,276	1,005,210.48	0.63	84	2,611	1,472,280	1.01	135	4,195

objective value as the tree-search method on which the total enumeration is implicitly based.

In general, the LBBD is more efficient computationally, particularly with larger h and r parameters. However, when r is small, the tree-search method is faster than LBBD generally. For example, in Table 8, for $|I| = 50$, $k = 50$, $h = 9$, and $r = 3$, the tree-search method takes about 21.36 seconds, whereas the LBBD

Table 6. Algorithms Performance Comparison for the FP with $w = 0.4$, q_j Calculated with $\alpha = 0.2, \theta = 400$ (Small Instances)

				Tree search			Greedy search			LBBD		
$\|I\|$	k	h	r	Time	nbAPs	Obj.	Time	nbAPs	$\Delta_{Obj.}$	Time	nbAPs	$\Delta_{Obj.}$
50	15	3	3	1.57	25	576,359.81	1.03	14	0.00	2.83	123	0.00
50	15	3	6	7.14	205	1,014,705.36	1.21	28	0.00	2.34	99	0.00
50	15	3	9	17.91	522	1,545,002.81	1.67	41	0.00	6.53	345	0.00
50	15	6	3	1.12	53	513,927.51	0.52	27	16,030.92	4.43	411	0.00
50	15	6	6	30.56	1,200	754,878.18	1.50	52	0.00	7.45	481	0.00
50	15	6	9	127.08	5,387	990,033.80	2.57	72	0.00	6.11	413	0.00
50	15	9	3	1.32	69	468,233.12	0.66	40	0.00	5.31	637	0.00
50	15	9	6	51.94	2,587	632,548.83	1.92	71	0.00	4.31	291	0.00
50	15	9	9	208.78	11,364	803,304.32	3.28	101	0.00	3.70	354	0.00
50	20	3	3	0.87	24	487,809.61	0.42	14	0.00	7.39	349	0.00
50	20	3	6	15.12	237	859,628.19	1.89	28	0.00	16.93	333	0.00
50	20	3	9	56.45	563	1,096,057.21	4.62	41	0.00	18.58	225	0.00
50	20	6	3	1.52	43	391,879.01	0.78	25	0.00	15.69	747	0.00
50	20	6	6	65.11	1,303	561,885.61	3.48	49	0.00	17.29	361	0.00
50	20	6	9	478.56	6,401	748,734.01	8.82	68	0.00	39.06	507	0.00
50	20	9	3	1.43	40	385,594.23	1.08	37	0.00	317.88	21,841	0.00
50	20	9	6	81.84	1,730	496,970.88	4.37	71	0.00	9.34	332	0.00
50	20	9	9	1,104.28	18,814	629,121.70	10.84	101	0.00	24.95	563	0.00
50	30	3	3	6.12	36	248,040.67	2.36	17	0.00	18.73	230	0.00
50	30	3	6	63.51	200	416,827.89	8.60	30	26,102.88	89.33	316	0.00
50	30	3	9	276.88	568	679,664.45	19.62	43	0.00	139.11	343	0.00
50	30	6	3	14.70	97	221,107.64	4.36	31	0.00	538.63	6,936	0.00
50	30	6	6	434.89	1,523	367,165.16	16.06	52	9,143.87	598.85	2,766	0.00
50	30	6	9	3,600.30	8,176	495,450.15	31.66	73	41,090.27	985.66	2,515	0.00
50	30	9	3	25.17	161	177,946.84	6.67	46	0.00	82.92	1,252	0.00
50	30	9	6	1,486.63	6,694	283,285.57	23.53	72	0.00	588.83	2,353	0.00
50	30	9	9	3,600.72	10,272	379,873.48	52.35	107	4,335.09	971.75	2,556	0.00
75	15	3	3	0.83	31	771,614.08	0.34	14	0.00	3.65	226	0.00
75	15	3	6	9.56	215	1,226,376.57	1.22	28	0.00	3.99	140	0.00
75	15	3	9	26.15	513	1,792,170.32	2.26	41	0.00	6.65	269	0.00
75	15	6	3	1.93	76	633,034.67	0.61	29	31,059.66	8.08	656	0.00
75	15	6	6	45.29	1,364	923,224.84	1.97	51	0.00	9.23	431	0.00
75	15	6	9	186.50	5,700	1,173,857.88	3.33	71	0.00	7.86	378	0.00
75	15	9	3	2.27	98	573,274.94	0.84	43	0.00	3.14	306	0.00
75	15	9	6	78.71	3,120	779,724.55	2.46	72	0.00	4.07	270	0.00
75	15	9	9	293.14	12,016	956,783.96	4.24	101	2,348.67	4.22	345	0.00
75	20	3	3	1.39	28	567,202.54	0.67	14	0.00	5.89	186	0.00
75	20	3	6	23.62	237	1,033,377.15	3.04	28	0.00	36.58	458	0.00
75	20	3	9	58.58	441	1,304,001.37	5.88	41	0.00	31.51	335	0.00
75	20	6	3	2.59	48	467,689.76	1.40	28	0.00	16.22	537	0.00
75	20	6	6	119.54	1,473	663,518.28	4.46	49	64,167.79	22.52	359	0.00
75	20	6	9	519.26	4,917	879,042.78	10.81	71	0.00	48.71	667	0.00
75	20	9	3	2.69	53	464,545.73	1.80	39	0.00	135.12	6,942	0.00
75	20	9	6	204.22	3,114	607,673.73	5.29	69	0.00	27.04	692	0.00
75	20	9	9	1,621.88	22,524	739,335.97	12.82	100	4,083.85	25.18	448	0.00
75	30	3	3	11.38	35	308,702.30	3.92	16	0.00	29.53	204	0.00
75	30	3	6	88.95	130	519,644.36	17.92	30	0.00	126.43	233	0.00
75	30	3	9	374.38	403	737,304.84	40.47	43	0.00	263.70	288	0.00
75	30	6	3	39.11	140	293,699.82	6.31	28	3,328.73	2,602.49	21,164	0.00
75	30	6	6	873.67	1,435	419,216.68	33.61	52	24,859.59	558.01	1,594	0.00
75	30	6	9	3,600.33	3,581	592,307.48	84.34	74	41,585.32	3,616.80	5,367	0.00
75	30	9	3	68.16	271	255,942.65	9.60	42	0.00	334.70	3,271	0.00
75	30	9	6	3,260.57	8,170	370,632.39	46.91	77	5,317.03	1,132.90	3,937	0.00
75	30	9	9	3,600.12	5,257	494,146.25	105.93	110	23,736.04	2,636.10	5,160	−12,784.06

takes about 64.67 seconds. Similarly, for $|I| = 70$, $k = 30$, $h = 9$, and $r = 3$, the tree-search method takes about 114.36 seconds, whereas the LBBD takes about 292.67 seconds. As noted, the tree-search method explores up to $(r^{h+1} - 1)/(r - 1)$ nodes and evaluates their solutions, whereas the LBBD solves a series of MIPs based on cuts to close bounds. We can see that the tree-search method is very sensitive to parameters h and r, whereas the LBBD is rather robust. In fact, the tree-search method clearly shows a fast increase in

Table 7. Algorithm's Performance Comparison for the FP with $w = 0.4$, q_j Calculated with $\alpha = 0.2, \theta = 400$ (Large Instances)

				Tree search			Greedy search			LBBD		
$\|I\|$	k	h	r	Time	nbAPs	Obj.	Time	nbAPs	$\Delta_{Obj.}$	Time	nbAPs	$\Delta_{Obj.}$
100	15	3	3	1.21	29	860,663.28	0.52	14	0.00	6.66	305	0.00
100	15	3	6	12.94	204	1,357,508.07	1.76	28	0.00	6.08	152	0.00
100	15	3	9	33.99	509	2,003,174.51	3.01	41	0.00	10.13	321	0.00
100	15	6	3	3.92	108	705,433.14	0.92	31	43,642.05	5.73	334	0.00
100	15	6	6	65.05	1,384	1,021,655.03	2.78	52	0.00	11.33	381	0.00
100	15	6	9	249.63	5,599	1,287,003.25	4.64	71	0.00	7.76	315	0.00
100	15	9	3	3.86	118	654,041.10	1.36	46	0.00	4.99	368	0.00
100	15	9	6	102.92	2,929	874,605.92	3.54	73	0.00	4.78	270	0.00
100	15	9	9	390.22	12,043	1,081,252.95	6.01	102	0.00	3.05	262	0.00
100	20	3	3	2.49	32	635,698.99	1.01	14	0.00	7.58	168	0.00
100	20	3	6	33.45	241	1,170,888.29	4.60	30	10,693.21	78.81	674	0.00
100	20	3	9	83.77	450	1,456,946.56	9.26	43	79,151.73	38.27	280	0.00
100	20	6	3	4.98	60	545,069.83	2.38	28	0.00	34.22	733	0.00
100	20	6	6	172.04	1,461	745,984.03	7.93	50	0.00	35.57	345	0.00
100	20	6	9	870.46	5,647	985,138.79	16.45	72	0.00	50.47	466	0.00
100	20	9	3	4.82	60	527,275.04	3.06	39	0.00	120.87	4,161	0.00
100	20	9	6	280.86	2,994	685,388.47	9.68	70	0.00	42.20	816	0.00
100	20	9	9	2,537.95	25,147	838,072.48	19.51	100	111,130.75	35.38	545	0.00
100	30	3	3	12.22	32	377,883.17	4.60	16	0.00	38.35	231	0.00
100	30	3	6	104.17	118	601,722.02	23.02	30	0.00	170.51	259	0.00
100	30	3	9	462.61	368	828,137.57	55.21	43	0.00	404.14	316	0.00
100	30	6	3	39.16	102	341,513.52	10.27	30	0.00	1,880.08	9,415	0.00
100	30	6	6	1,098.38	1,217	496,235.51	45.21	52	14,034.94	967.08	1,691	0.00
100	30	6	9	3,600.12	3,167	662,208.77	118.95	72	32,942.93	2,637.81	2,698	0.00
100	30	9	3	73.78	201	309,779.23	14.59	42	0.00	899.73	5,848	0.00
100	30	9	6	3,600.19	6,082	423,395.01	53.96	72	2,519.00	1,042.40	2,402	0.00
100	30	9	9	3,600.17	3,574	562,349.98	127.60	102	19,955.23	3,499.95	4,822	−8,245.34
150	15	3	3	1.99	31	1,013,516.61	0.84	14	0.00	10.35	280	0.00
150	15	3	6	22.65	220	1,608,989.32	3.00	28	0.00	10.15	163	0.00
150	15	3	9	50.68	487	2,258,751.20	5.18	41	0.00	13.81	280	0.00
150	15	6	3	4.13	67	843,480.11	1.53	28	0.00	27.27	917	0.00
150	15	6	6	96.53	1,114	1,241,652.98	5.02	51	0.00	38.36	821	0.00
150	15	6	9	361.47	5,077	1,526,860.23	7.93	71	0.00	14.64	386	0.00
150	15	9	3	4.31	75	807,567.25	1.94	41	0.00	13.36	801	0.00
150	15	9	6	149.77	2,589	1,052,999.80	5.44	69	0.00	8.38	347	0.00
150	15	9	9	573.06	11,708	1,289,876.08	9.67	99	0.00	6.66	339	0.00
150	20	3	3	3.69	23	722,874.51	1.95	14	0.00	17.08	205	0.00
150	20	3	6	53.69	200	1,218,049.89	7.79	28	0.00	65.42	279	0.00
150	20	3	9	154.10	491	1,698,605.53	15.13	41	0.00	67.09	313	0.00
150	20	6	3	5.90	34	656,362.21	3.60	25	0.00	123.11	1,511	0.00
150	20	6	6	277.92	1,126	892,969.86	15.61	48	0.00	76.73	448	0.00
150	20	6	9	1,909.65	7,372	1,167,456.35	27.66	70	916.97	85.45	524	0.00
150	20	9	3	6.21	40	632,827.03	5.56	41	0.00	249.59	5,276	0.00
150	20	9	6	625.29	3,773	825,139.47	23.14	74	4,261.55	166.76	1,737	0.00
150	20	9	9	3,600.06	22,324	1,015,841.44	35.02	101	22,164.32	131.88	1,235	0.00
150	30	3	3	19.94	33	448,488.62	6.79	16	0.00	62.30	259	0.00
150	30	3	6	162.58	119	712,366.40	35.60	30	0.00	426.34	427	0.00
150	30	3	9	683.84	373	969,595.10	85.02	43	0.00	637.96	344	0.00
150	30	6	3	68.59	111	414,030.45	14.21	29	0.00	964.17	3,764	0.00
150	30	6	6	2,218.82	1,596	594,611.90	67.78	52	18,955.78	2,260.65	2,741	0.00
150	30	6	9	3,602.11	2,402	774,668.70	158.81	72	55,835.42	3,179.83	2,440	0.00
150	30	9	3	107.56	184	370,223.08	21.92	43	0.00	460.65	2,350	0.00
150	30	9	6	3,600.71	3,959	511,672.90	81.96	72	4,171.87	3,067.12	4,884	0.00
150	30	9	9	3,600.08	2,221	682,826.96	164.74	101	998.87	3,607.46	4,190	−18,626.93

solution time when r increases, whereas the LBBD shows a linear increase generally. For instance, when $|I| = 50$, $k = 50$, $h = 9$, and $r = 6$, the tree-search method takes about 1,306 seconds, whereas LBBD takes about 472 seconds. For $|I| = 50$, $k = 50$, $h = 9$, and $r = 9$, the tree-search method uses up to 3,600 seconds, whereas the LBBD takes about 636 seconds. In summary, the tree-search method is advantageous over the LBBD

Table 8. Algorithm's Performance Comparison for the FP with $q_j = 0$, $w = 0.4$

				Tree search			Greedy search			LBBD		
$\|I\|$	k	h	r	Time	nbAPs	Obj.	Time	nbAPs	$\Delta_{Obj.}$	Time	nbAPs	$\Delta_{Obj.}$
50	15	3	3	0.90	27	553,690.28	0.23	14	0	1.38	97	0
50	15	3	6	6.58	197	991,477.44	0.98	28	0	2.60	153	0
50	15	3	9	15.56	535	1,523,560.55	2.01	41	0	4.81	372	0
50	15	6	3	1.33	65	489,576.70	0.46	29	38,285	3.53	381	0
50	15	6	6	30.68	1,127	736,644.47	1.28	52	0	6.53	483	0
50	15	6	9	122.29	5,325	964,500.04	2.02	72	0	5.03	432	0
50	15	9	3	1.30	79	448,504.84	0.61	45	0	4.82	648	0
50	15	9	6	56.82	2,538	611,719.19	1.71	75	0	3.21	279	0
50	15	9	9	200.62	10,932	777,804.20	2.60	101	0	2.56	309	0
50	20	3	3	0.70	26	463,362.84	0.36	14	0	4.54	277	0
50	20	3	6	12.99	233	837,066.04	2.03	28	0	14.30	384	0
50	20	3	9	46.23	565	1,071,421.88	3.77	41	0	10.66	188	0
50	20	6	3	1.42	43	369,896.80	0.73	25	0	14.02	733	0
50	20	6	6	65.85	1,265	533,187.56	3.51	49	0	7.12	207	0
50	20	6	9	468.04	6,416	717,374.14	6.51	68	0	25.67	494	0
50	20	9	3	1.61	40	363,173.69	1.72	38	0	277.61	22,073	0
50	20	9	6	82.45	1,697	474,378.20	3.65	72	0	6.08	274	0
50	20	9	9	1,084.15	20,336	599,775.74	7.23	100	79,207	15.28	443	0
50	30	3	3	4.84	36	230,543.34	1.86	17	0	20.30	286	0
50	30	3	6	55.97	213	397,748.65	6.89	30	26,075	78.15	372	0
50	30	3	9	232.89	562	656,346.53	15.16	43	0	106.01	343	0
50	30	6	3	12.76	99	201,738.88	3.65	31	0	469.95	6,964	0
50	30	6	6	365.44	1,595	350,259.62	12.62	52	10,158	547.29	3,213	0
50	30	6	9	3,356.38	10,208	485,306.91	25.62	74	0	961.69	3,239	0
50	30	9	3	21.36	158	158,458.31	5.40	47	9,329	64.67	1,032	0
50	30	9	6	1,306.63	7,049	263,705.80	17.81	73	0	472.34	2,398	0
50	30	9	9	3,600.02	12,684	360,193.41	40.27	105	6,059	636.49	2,203	0
75	15	3	3	0.73	32	741,853.99	0.27	14	0	3.32	254	0
75	15	3	6	6.06	205	1,186,485.59	0.87	28	0	2.96	154	0
75	15	3	9	17.71	499	1,749,187.02	1.43	41	0	4.41	267	0
75	15	6	3	1.96	99	603,902.98	0.54	31	32,389	8.31	797	0
75	15	6	6	33.99	1,338	891,530.53	1.40	51	0	6.08	403	0
75	15	6	9	161.08	5,771	1,137,210.53	2.35	71	0	5.68	356	0
75	15	9	3	2.22	119	542,364.31	0.75	47	8,112	2.44	288	0
75	15	9	6	64.90	3,256	748,789.71	2.05	72	0	3.51	316	0
75	15	9	9	261.30	12,330	914,906.31	3.19	101	0	2.29	247	0
75	20	3	3	1.22	31	538,242.18	0.59	14	0	6.16	224	0
75	20	3	6	17.38	237	1,005,228.28	2.23	28	0	30.14	494	0
75	20	3	9	45.74	451	1,268,825.96	4.65	41	0	22.92	370	0
75	20	6	3	2.31	52	440,277.14	1.21	28	0	11.25	463	0
75	20	6	6	100.35	1,518	647,722.01	3.85	49	0	28.74	599	0
75	20	6	9	433.35	5,034	853,040.77	8.02	73	0	27.72	528	0
75	20	9	3	2.49	59	436,826.09	1.63	40	0	117.89	7,127	0
75	20	9	6	194.29	3,645	582,442.80	4.92	71	0	22.46	694	0
75	20	9	9	1,553.66	26,606	710,975.64	9.88	100	12,857	22.01	536	0
75	30	3	3	10.35	37	287,800.16	3.40	16	0	27.05	231	0
75	30	3	6	69.89	130	497,332.84	14.91	30	0	135.11	288	0
75	30	3	9	294.82	403	715,805.38	56.68	43	0	190.69	316	0
75	30	6	3	75.88	144	272,366.83	11.05	27	2,453	1,859.19	17,914	0
75	30	6	6	1,706.35	1,548	396,752.80	62.74	54	0	566.27	1,871	0
75	30	6	9	3,601.75	2,709	566,189.91	105.87	75	0	1,868.40	3,817	0
75	30	9	3	114.36	283	234,408.48	19.12	46	201	292.67	3,582	0
75	30	9	6	3,600.37	4,714	344,614.04	61.82	80	0	816.09	3,417	0
75	30	9	9	3,600.43	2,992	470,508.22	77.04	107	17,929	1,739.92	4,694	−11,452

when r is very small, but the LBBD is faster when r increases. Also, as shown in Table 8, regarding solution quality, the LBBD is as good as or better than the tree-search method. For all the instances in Table 8, the tree-search method yields optimal solutions, except for four instances (when it exceeds the 3,600-second time limit). The LBBD gets a better solution for one of the four instances ($|I| = 75$, $k = 30$, $h = 9$, $r = 9$). Similar

Table 9. Algorithm's Performance Comparison for the FP with $q_j = 0$, $w = 0$

				Tree search			Greedy search			LBBD		
$\|I\|$	k	h	r	Time	nbAPs	Obj.	Time	nbAPs	$\Delta_{Obj.}$	Time	nbAPs	$\Delta_{Obj.}$
50	15	3	3	1.13	38	514,054.92	0.47	15	0	2.49	268	0
50	15	3	6	3.38	237	753,683.00	0.63	29	0	1.54	188	0
50	15	3	9	4.87	590	1,039,038.47	0.82	42	0	1.30	244	0
50	15	6	3	1.63	178	417,496.01	0.30	27	3,458	1.30	876	0
50	15	6	6	18.86	2,854	542,675.16	0.83	52	17,560	3.50	757	0
50	15	6	9	37.12	10,374	650,059.36	1.06	74	0	2.25	731	0
50	15	9	3	4.30	696	374,094.37	0.66	46	15,714	5.05	1,239	0
50	15	9	6	38.00	11,747	459,406.47	0.63	74	39,019	2.01	865	0
50	15	9	9	66.74	29,489	459,406.47	0.89	103	7,638	1.82	844	0
50	20	3	3	0.67	38	375,195.12	0.37	15	0	2.30	203	0
50	20	3	6	4.78	257	525,720.38	1.07	29	0	3.37	169	0
50	20	3	9	16.66	652	733,801.14	2.31	42	0	5.24	294	0
50	20	6	3	1.82	134	307,899.23	0.52	27	0	22.94	2,998	0
50	20	6	6	50.22	3,534	404,907.22	1.51	50	0	15.55	1,414	0
50	20	6	9	282.41	23,325	518,280.17	2.70	72	1,615	15.98	2,055	0
50	20	9	3	4.77	461	271,957.38	0.65	39	2,270	30.77	5,207	0
50	20	9	6	238.48	27,674	352,670.96	1.91	73	13,835	21.13	3,562	0
50	20	9	9	1,236.93	203,958	422,153.75	3.06	105	5,703	10.29	2,708	0
50	30	3	3	3.79	39	230,543.34	1.60	17	0	25.44	539	0
50	30	3	6	25.23	241	367,213.07	4.41	31	3,522	51.99	541	0
50	30	3	9	77.22	622	505,745.05	9.78	44	0	89.40	654	0
50	30	6	3	11.24	194	166,435.99	2.17	31	14,165	28.78	1,212	0
50	30	6	6	248.98	4,522	239,576.54	5.72	53	28,660	95.29	1,966	0
50	30	6	9	1,561.76	27,537	326,895.34	12.81	75	23,737	134.46	2,416	0
50	30	9	3	35.40	717	141,150.09	2.97	44	10,294	218.56	8,690	0
50	30	9	6	1,636.47	36,577	200,619.50	7.62	76	11,784	189.62	5,332	0
50	30	9	9	3,600.00	75,894	270,018.55	15.38	107	6,434	400.10	11,844	0
75	15	3	3	0.46	37	643,344.00	0.20	15	0	2.51	320	0
75	15	3	6	3.02	245	961,282.39	0.61	29	0	3.79	427	0
75	15	3	9	5.23	603	1,250,612.61	0.86	42	0	1.60	346	0
75	15	6	3	1.42	201	527,321.33	0.26	28	0	4.53	1,013	0
75	15	6	6	19.78	3,296	688,960.68	0.68	51	0	3.93	939	0
75	15	6	9	33.21	10,157	821,520.45	0.96	73	9,646	2.03	770	0
75	15	9	3	3.02	579	487,128.81	0.30	40	5,237	5.90	1,856	0
75	15	9	6	39.60	11,035	602,162.81	0.72	72	4,062	3.58	1,396	0
75	15	9	9	78.01	29,453	602,162.81	1.09	101	0	3.08	1,218	0
75	20	3	3	1.08	43	451,143.85	0.43	15	0	5.23	258	0
75	20	3	6	7.93	269	647,722.01	1.52	29	0	8.08	278	0
75	20	3	9	19.22	535	888,836.14	3.24	42	0	6.78	280	0
75	20	6	3	3.66	198	399,159.59	0.68	27	0	30.13	2,821	0
75	20	6	6	75.55	4,285	521,735.19	2.00	50	0	24.29	1,813	0
75	20	6	9	314.24	20,371	646,481.98	3.59	74	13,295	19.75	1,925	0
75	20	9	3	9.24	685	368,305.48	0.85	39	542	100.66	14,939	0
75	20	9	6	323.59	26,450	464,085.48	2.54	74	12,487	43.03	5,758	0
75	20	9	9	1,389.03	176,911	559,897.07	4.05	104	30,816	27.86	6,084	0
75	30	3	3	7.23	39	287,800.16	2.76	16	0	47.47	595	0
75	30	3	6	53.83	211	446,548.77	8.73	30	0	137.21	707	0
75	30	3	9	182.71	661	620,483.71	15.91	43	29,048	190.48	1,072	0
75	30	6	3	17.75	204	256,606.50	3.28	29	8,121	490.76	13,617	0
75	30	6	6	377.59	4,560	348,998.60	12.50	52	25,567	143.37	2,940	0
75	30	6	9	3,241.10	37,530	428,855.54	20.13	77	29,746	169.09	2,490	0
75	30	9	3	45.77	794	210,965.65	4.24	46	0	175.01	6,671	0
75	30	9	6	2,281.47	42,662	285,617.87	13.12	77	14,643	312.02	6,943	0
75	30	9	9	3,600.03	63,426	365,537.49	23.69	107	35,593	464.60	11,146	0

observations on computational efficiency and solution quality can be made in Tables 6–9. It should be noted that, in addition to h and r, $|I|$ and k also impact the computational cost of an instance, due to the fact that inside this bilevel problem, there is a facility location problem embedded, in which one has to determine a k-combination of a set $|I|$ locations as $\binom{|I|}{k}$.

6. Discussions and Conclusion

This paper has presented a bilevel programming model along with solution methods to determine 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, and can be extended to other network optimization problems such as critical infrastructure systems in which both natural disasters and intentional attacks may occur. We show that the attacker's interdiction 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. The 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.

The proposed framework can be applied to model and solve a bilevel problem: the defender's problem and the attacker's problem. It naturally fits the settings of a supply chain network, in which one considers the fortification decisions while anticipating disruptions, either endogenous or exogenous. The attacker's problem is proved to be supermodular. However, the fortification problem, for which we developed the LBBD solution method, is neither supermodular nor submodular. The overall framework can be applied to other application areas with similar settings. For instance, consider a 5G telecommunication network. Upon natural disasters or cyberattacks, a tower or a critical device may not provide services to users. The users will lose service or have to rely on nearby but farther towers. The service provider (defender) hence seeks a solution to fortify the network against risks and make the system resilient, whereas the attacker aims to bring maximum damage to the system. Another potential application is the critical infrastructure system, in which physical and cyber systems and assets provide essential services that underpin the society, and their destruction or incapacity would have a debilitating impact on almost every aspect of our life. Take the energy sector as an example. It provides an "enabling function" across all critical infrastructure sectors. To hedge against natural disasters and intentional attacks, fortifications, such as the injection of distributed generation based on photovoltaic and energy storage systems, are vital to increase the resilience. The proposed framework can empower the decision makers, as a defender, to determine which critical public resources (such as hospitals for healthcare, police and fire stations for protection, grocery stores for food, and so on) to fortify under certain budgetary constraints and thus minimize the overall impact from disruptions.

There are several possible extensions of this work. First, parameters h and r are already known in this study; however, 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. Thus, it would be interesting to extend the model to explicitly model h as a decision variable and r with a probability distribution. Second, facilities are assumed to have unlimited capacity; the model presented in this work will no longer be applicable when the facility capacity is considered. When the capacity constraint is considered, the nearest assignment property is no longer held and an assignment problem has to be solved in order to evaluate an attack pattern, which is NP-hard in general. Thirdly, the network/facilities were predetermined in this paper. It would be beneficial to extend this work to consider facility location decisions, that is, to make the location and fortification decisions simultaneously.

Acknowledgments

The authors would like to thank the area editor and anonymous referees, whose comments improved the paper significantly.

References

Aboolian R, Cui T, Shen ZJM (2013) An efficient approach for solving reliable facility location models. *INFORMS J. Comput.* 25(4): 720–729.

Aksen D, Piyade N, Aras N (2010) The budget constrained r-interdiction median problem with capacity expansion. *Central Eur. J. Oper. Res.* 18(3):269–291.

Angulo G, Ahmed S, Dey SS (2016) Improving the integer l-shaped method. *INFORMS J. Comput.* 28(3):483–499.

Atamtürk A, Narayanan V (2008) Polymatroids and mean-risk minimization in discrete optimization. *Oper. Res. Lett.* 36(5):618–622.

Chu Y, Xia Q (2004) Generating benders cuts for a general class of integer programming problems. *Lecture Notes Comput. Sci.* 3011: 127–141.

Church RL, Scaparra MP (2007) Protecting critical assets: The *r*-interdiction median problem with fortification. *Geographical Anal.* 39(2):129–146.

Church R, Scaparra M, Middleton R (2004) Identifying critical infrastructure: The median and covering facility interdiction problems. *Ann. Assoc. Amer. Geographers* 94(3):491–502.

Cui T, Ouyang Y, Shen ZJM (2010) Reliable facility location design under the risk of disruptions. *Oper. Res.* 58(4):998–1011.

Edmonds J (1970) Submodular functions, matroids, and certain polyhedra. Goos G, Hartmanis J, van Leeuwen J, eds.

Combinatorial Structure and Their Applications (Gordon and Breach, New York), 69–87.

Hooker JN, Ottosson G (2003) Logic-based benders decomposition. *Math. Programming* 96(1):33–60.

Iwata S, Orlin JB (2009) A simple combinatorial algorithm for submodular function minimization. *Proc. 20th Annual ACM-SIAM Sympos. Discrete Algorithms* (SIAM, Philadelphia), 1230–1237.

Li Q, Zeng B, Savachkin A (2013) Reliable facility location design under disruptions. *Comput. Oper. Res.* 40(4):901–909.

Liberatore F, Scaparra MP, Daskin MS (2011) Analysis of facility protection strategies against an uncertain number of attacks: The stochastic *r*-interdiction median problem with fortification. *Comput. Oper. Res.* 38(1):357–366.

Liberatore F, Scaparra MP, Daskin MS (2012) Hedging against disruptions with ripple effects in location analysis. *Omega* 40(1): 21–30.

Lim M, Daskin MS, Bassamboo A, Chopra S (2010) A facility reliability problem: Formulation, properties, and algorithm. *Naval Res. Logist.* 57(1):58–70 (NRL).

McCormick ST (2005) Submodular function minimization. *Handbooks Oper. Res. Management Sci.* 12:321–391.

Nemhauser GL, Wolsey LA (1988) *Integer and Combinatorial Optimization* (Wiley, New York).

Orlin JB (2009) A faster strongly polynomial time algorithm for submodular function minimization. *Math. Programming* 118(2): 237–251.

Rahmaniani R, Crainic TG, Gendreau M, Rei W (2017) The Benders decomposition algorithm: A literature review. *Eur. J. Oper. Res.* 259(3):801–817.

Rebennack S (2016) Combining sampling-based and scenario-based nested benders decomposition methods: Application to stochastic dual dynamic programming. *Math. Programming* 156: 343–389.

Scaparra MP, Church RL (2008a) A bilevel mixed-integer program for critical infrastructure protection planning. *Comput. Oper. Res.* 35(6):1905–1923.

Scaparra MP, Church RL (2008b) An exact solution approach for the interdiction median problem with fortification. *Eur. J. Oper. Res.* 189(1):76–92.

Shen ZJM, Zhan RL, Zhang J (2011) The reliable facility location problem: Formulations, heuristics, and approximation algorithms. *INFORMS J. Comput.* 23(3):470–482.

Snyder LV, Daskin MS (2005) Reliability models for facility location: the expected failure cost case. *Transportation Sci.* 39(3):400–416.

Snyder LV, Atan Z, Peng P, Rong Y, Schmitt AJ, Sinsoysal B (2016) OR/MS models for supply chain disruptions: A review. *IIE Trans.* 48(2):89–109.

Soares J, Canizes B, Ghazvini MAF, Vale Z, Venayagamoorthy GK (2017) Two-stage stochastic model using benders' decomposition for large-scale energy resource management in smart grids. *IEEE Trans. Indust. Appl.* 53(6):5905–5914.

Wu T, Zhang K (2014) A computational study for common network design in multi-commodity supply chains. *Comput. Oper. Res.* 44:206–213.

Zhang K, Li X, Jin M (2021) Data set and source code for "Efficient solution methods for a general *r*-interdiction median problem with fortification," https://github.com/ILABUTK/IJOC-General-RIMF.

Zhang X, Zheng Z, Zhu Y, Cai KY (2014) Protection issues for supply systems involving random attacks. *Comput. Oper. Res.* 43: 137–156.

Zhu Y, Zheng Z, Zhang X, Cai K (2013) The r-interdiction median problem with probabilistic protection and its solution algorithm. *Comput. Oper. Res.* 40(1):451–462.