A BILEVEL PARTIAL INTERDICTION PROBLEM WITH CAPACITATED FACILITIES AND DEMAND OUTSOURCING

by

Sema Şengül Akca

BS, Mathematical Engineering, Yıldız Teknik University, 2007

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering
Boğaziçi University
2011

A BILEVEL PARTIAL INTERDICTION PROBLEM WITH CAPACITATED FACILITIES AND DEMAND OUTSOURCING

APPROVED BY:

Assoc. Prof. Necati Aras (Thesis Supervisor)	....
Assist. Prof. Deniz Aksen (Thesis Co-supervisor)	....
Prof. Kuban Altınel	....
Assist. Prof. Caner Taşkın	....
Assoc. Prof. Aslı Sencer Erdem	....

DATE OF APPROVAL: 15.06.2011

To the Şengül and Akca families.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude and regards to my thesis supervisor Assoc. Prof. Necati Aras and my thesis co-supervisor Assist. Prof. Deniz Aksen for their continuous guidance, support and endless patience in every stage of my studies. This thesis would not have been completed or written without their encouragement and effort.

I would also like to thank to Prof. Kuban Altınel, Assist. Prof. Caner Taşkın and Assoc. Prof. Aslı Sencer Erdem for their interest and sparing time in examining my thesis and taking part in my thesis jury.

I am also grateful to TUBITAK - Turkish Technological and Scientific Research Institute - for their financial support during my master education.

My special thanks go to my thesis co-supervisor Assist. Prof. Deniz Aksen who guided me about C# programming. I would also like to express my thanks to all my close friends for their intimacy and sharing this unforgettable two years together.

I also gratefully acknowledge the Bogazici University Industrial Engineering faculty, assistants, and secretaries. It has been a great pleasure and honor to be a part of this family.

At last but not the least, I am deeply indebted to my husband and my dear parents, whose love, trust and support made this possible. Their patience deserve the most appreciation. I would like to express how I am fortunate to have them.

ABSTRACT

A BILEVEL PARTIAL INTERDICTION PROBLEM WITH CAPACITATED FACILITIES AND DEMAND OUTSOURCING

The bilevel partial interdiction problem with capacitated facilities and demand outsourcing involves a static Stackelberg game between a system planner and a potential attacker. The system planner (defender) is responsible for satisfying the overall demand of customers in an existing service network and aims at minimizing the total demand-weighted transportation cost while serving customers from the capacitated facilities. Simultaneously, he should consider the possible capacity reduction of some facilities in the wake of a destructive attack while the attacker's objective is to cause maximum disruption in the service level. The number of facilities to be attacked cannot be known a priori but heavily depends upon the attacker's interdiction budget. Regarding the *partial interdiction* concept, this defender-attacker relationship is formulated as a bilevel programming model. The attacker takes on the leader role, and forces the system planner, who acts as the follower, to meet customer demands with a higher outsourcing cost. Two different methods are proposed in this study. The first method is a progressive grid search which is impracticable on large-sized problems. The second method is a multi-start revised simplex search heuristic which is based on the Nelder-Mead simplex search method and is developed to overcome the exponential time complexity of the first method. We also develop an exhaustive search to solve all combinations of the full interdiction of the facilities to assess the benefit of partial interdiction from the perspective of the attacker. Our test results indicate that it would be more beneficial to disrupt facility capacities partially rather than totally from the perspective of the attacker.

ÖZET

KAPASİTE KISITLI ÇİFT DÜZEYLİ KISMI SALDIRI PROBLEMİ

Çift düzeyli kısmi saldırı problemi, sistem planlayıcısı(savunan) ile potansiyel saldırgan(terörist) arasındaki statik Stackelberg oyununu içerir. Sistem planlayıcısı mevcut hizmet ağının işletilmesinden sorumludur ve amacı, belli kapasitelere sahip tesislerden müşteri taleplerini sağlarken talep ağırlıklı taşıma maliyetini enküçüklemektir. Bunu yaparken, terörist saldırısı sonucu bazı tesislerin tam kapasiteyle hizmet veremeyeceğini hesaba katmaktadır. Saldırılacak tesis sayısı önceden bilinememekle beraber saldırganın bütçesine bağlıdır. Kısmi saldırı kavramı kullanılarak, problem çift düzeyli programlama modeli olarak formüle edilmiştir. Saldırgan lider oyuncudur ve saldırı sonrası müşteri talebini en yüksek maliyetle sistem planlayıcısına sağlatmaya çalışırken, takipçi oyuncu olan sistem planlayıcısı bunun tam tersini amaçlamaktadır. Problemimizi çözmek için iki metot önermekteyiz. Bunlardan ilki, problem büyüklüğüne oranla üssel bir çözüm süresi gerektiren kafes araması yöntemidir. İkincisi, ilk yöntemin büyük problemlerde uygulanamaması nedeniyle geliştirilen tekrarlı revize edilmiş simpleks araması yöntemidir. Literatürde ilk kez bu çalışmada kısmi saldırı fikri kullanıldığından, bunun tam saldırılara oranla yarar sağlayıp sağlamadığını araştırmak amacıyla tam saldırılı problemleri çözen tam arama yöntemi geliştirilmiştir. Saldırganın bütçesi göz önüne alınarak mevcut tesis kapasitelerinin tam olarak imha edilmesinin kombinasyonlarını değerlendiren bu yöntem, saldırgan açısından kısmen saldırmanın tamamen saldırmaktan daha iyi sonuçlar vereceğini göstermiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a	Size of a simplex
a_i	Demand of customer i
c_i	Cost of shipping customer i's one unit demand for one unit distance
c_p	Cost of outsourcing customer i's one unit demand (independent of distance)
(c_{xi}, c_{yi})	The coordinates of customer i
d_{ij}	Euclidean distance between customer i and facility at site j
e_j	Cost of interdicting full capacity of facility at site j
e_{tot}	Total interdiction budget of the attacker
(f_{xj}, f_{yj})	The coordinates of facility site j
q_j	Capacity of the facility at site j
S_j	Fraction of the facility capacity lost at site j due to an interdiction
s_l	Large step size for Lagrangian multipliers
s_s	Standard step size for Lagrangian multipliers
x_0	Point with the lowest function value f(x_0)
x_e	Expansion point
x_n	Point with the highest function value, f(x_n
$\overline{x}$	The centroid of the simplex with respect to x_0
x_r	Reflection point
x_{ic}	Partial interior contraction point
x_{oc}	Partial exterior contraction point
V_{ij}	Assignment of customer i to facility at site j after the attack
$\Delta_{i,j}$	Possible increment in the ith element of S_j vector
λ_i	Lagrangian multiplier
α	Reflection coefficient
γ	Expansion coefficient

β	Contraction coefficient
δ	Shrink coefficient
φ	Iteration number
υ	Initial step size parameter
τ	Phase Counter
ω	Plateau Counter

LIST OF ACRONYMS/ABBREVIATIONS

BCRIMF-CE	Budget constrained r–interdiction median problem with capacity expansion
BIP	Bilevel interdiction problem
BMILP	Bilevel mixed integer linear program
BP	Bilevel programming
BPIP	Bilevel partial interdiction problem
ES	Exhaustive Search
FRIMT	Fortification r–interdiction median problem with facility recovery time
H	Simplex with $n+1$ vertices
IMF	Interdiction median problem with fortification
K	Multi-start number
LLP	Lower level problem
MCLIP	Maximum covering location interdiction problem
MCLP	Maximal covering location problem
MCPC	Maximal covering problem with precedence constraints
MIP	Mixed integer programming
MILP	Mixed integer linear programming
MS-RSS	Multi-start Revised Simplex Search
NMSS	Nelder-Mead Simplex Search
PGS	Progressive Grid Search
RIC	r-interdiction covering problem
RIM	r-interdiction median problem
RIMF	r-interdiction median problem with fortification
S-RIMF	Stochastic r-interdiction median problem with fortification
SVIs	Supervalid Inequalities
U	Uniform
ULP	Upper level problem

1. INTRODUCTION

Having a history larger than the modern nation-state, terrorist attacks concern both the government and public interest since terrorists are developing new capabilities for attack and improving the efficiency of the existing methods. The flexibility and adaptability of terror throughout the years has contributed to the growth in the number of terrorist attacks. Alterations in the tactics and techniques of the terrorists have been considerable but what is more considerable is the places where terrorism is used. Most of the recent attacks affirm an alarming trend toward more sophisticated, better planned and coordinated strikes so as to engender the utmost harm to public. For instance, The 2003 İstanbul bombings were realized by suicide bombers at the British Consulate and headquarter of a British bank, leaving 57 people dead and 700 wounded. With the aim of maximizing the impact of bombs, the bombers waited for the traffic lights to turn red. The most appalling example is the 9/11 which were a series of intentional attacks by al-Qaeda to the World Trade Center buildings and the Pentagon resulting in 3000 mortalities where the overwhelming majority of these was civilians from various nations.

The aggressive use of modern technology for information management, communication, and intelligence has increased the attractiveness of soft targets for terrorist attacks. Attacks to telecommunication towers in Afghanistan (2008), to an ambulance station in North Ireland (1999), and to electric grid (Motto et al., 2005) are significant examples where the terrorists' intent is to troublesome service level rather than to kill people.

In this thesis, we focus on a service network which is considered a soft target and study the *b*ilevel *p*artial *i*nterdiction *p*roblem for the planning of critical facilities referred to the BPIP. The problem corresponds to a static Stackelberg game between a system planner referred to as defender and a potential attacker. The aim of the former is to provide critical service to customers residing at a number of demand nodes under the man-made attacks, whereas that of the latter is to cause the maximum possible disruption to this service system. Since these facilities are apt to interdiction attempts of the attacker, the defender has to consider the attacks while deciding on the allocation of the resources to minimize the worst-case disruption cost that can possibly be inflicted by the attacker.

One of the differentiating features of this study is that it is the first attempt for the partial interdiction of the facilities in the leader-follower game, and this difficult problem is not addressed before in the literature. Notably lacking in the past model development is partial disruption on the facilities. In some cases, it is quite possible that the facilities may operate at a reduced level compared to their defined level of operation unless they are completely damaged. Most of the past works cannot be directly applied in this type of problem since they focus only on full interdiction, in other words, if a facility is attacked, any demand node in the system cannot be served from this facility. The aim of this paper is to propose a new model for the partial interdiction model for capacitated facilities and a solution methodology to guide planners and policy makers.

The problem is modeled as a bilevel mixed integer programming formulation and two solution algorithms are developed. One of them is a progressive grid search which explores the optimal solution of the problems through grid points. The other one is based on a direct search methods, the Nelder-Mead Simplex search (NMSS). Although it is a very efficient optimization algorithm, we encountered many problems such as bounded spaces, collapse of simplex, large plateaus and premature convergence problems in the implementation of NMSS. We therefore employ a variant of NMSS procedure, which is based on the Revised Simplex Search (RSS)(Humphrey and Wilson, 2000). It consists of a three-phase application of the NMSS method. Additionally, we include some mechanisms that are designed to avoid some of the critical weaknesses of NMSS as mentioned above. The search procedure explores the solution space for finding the attacker's optimal interdiction policy which is used as an input to the second level problem. The second level problem is then solved to optimality by CPLEX 11.0 to obtain the system planner's optimal reaction plan. The results obtained on 84 randomly generated test instances reveal that the proposed solution algorithm works quite well in a wide spectrum of problem sizes. Finally, in an effort to assess the performance of the partial interdiction concept, an exhaustive search for full interdiction problems is developed.

2. LITERATURE SURVEY

2.1. Interdiction Problems for Facility Location

The identification of critical system components due to deliberate sabotage and terrorist attacks is of particular concern in the operation of critical network infrastructures. Critical infrastructure includes certain components of a system whose failure may cause a disorder in the operational and functional capabilities of the system resulting in the deterioration of the performance due to service disruptions or excessive transportation costs. When this failure possibility is considered in the design phase of the network, the effect of such a disorder can be mitigated. Based on this motivation, a variety of systematic and analytical tools to address issues of increasing the post-attack functionality of the network in the presence of malicious actions of an intelligent attacker have recently been developed. The vast majority of these efforts focus on modeling these rational attacks through the use of interdiction models which are rooted in military defense applications. Intentional attacks are different from the natural catastrophic events, labor actions or other similar factors in that a terrorist intelligently attacks the system in order to cause the maximum damage.

In the wake of a destructive attack, many interdiction models that differ in the objectives and underlying network structures have been proposed in the literature. Much research has been dedicated to the design of survivable networks and the reinforcement or fortification of existing networks. Wollmer made the first attempt to model interdiction of supply lines as an optimization model (Wollmer, 1964). Since then, numerous papers dealing with the interdiction of transportation networks have appeared. The majority of the past work is founded on the network optimization theory and generally seeks interdicting arcs with the intention of minimizing the network flow capacity (McMaster and Mustin, 1970; Ghare et al., 1971; Wood, 1993) or maximizing the shortest path between a specified origin and destination (Fulkerson and Harding, 1977; Golden, 1978; Israeli and Wood, 2002). Supply or emergency facility interdiction in a service network has recently been modeled. As opposed to the arc interdiction models, facilities are potential targets for attackers in facility interdiction models. Since the center of attention in the proposed model is the facility interdictions, the literature corresponding to this type of interdiction models is reviewed.

The first published study of supply or emergency facility interdiction in a service network belongs to (Church et al., 2004). Given p existing facilities serving the customers, the authors develop two facility interdiction models from the attacker's viewpoint called the r-interdiction median model (RIM) and the r-interdiction covering model (RIC), which are based on the p-median problem and on the maximal covering problem, respectively. The aim of RIM is to determine a subset of r facilities to be destroyed among the set of p supply or emergency response facilities whose removal will fulfill the attacker's aim the most. On the other hand, the objective of RIC is to choose a subset of r facilities among the set of p existing facilities, which if attacked will cause the greatest reduction in covered customer demand. As can be seen, RIM and RIC identify the most critical facilities in the network. These two single-level models are solved with a commercial MILP solver.

Church and Scaparra extend the interdiction models of (Church et al., 2004) to include the option of fortification against interdiction in (Church and Scaparra, 2007). In fact, the reason for the fortification is provide a less costly alternative to the system planners as they can allocate limited resources among possible mitigation investments to protect the critical infrastructure from distruption. By doing so, they may offer the continuity of service provision in the existence of deliberate attacks. Adding the fortification option results in a mixed-integer linear programming problem called the interdiction median problem with fortification (IMF). IMF identifies the best fortification plan by fortifying a subset of q facilities among the set of p facilities in a service system with n demand nodes. The defender's objective is to minimize the total demand satisfaction cost expressed as the demand-weighted shortest distance between the non-interdicted facilities and customers, while the attacker's objective is the reverse of that of the leader, i.e., interdicting r facilities. Due to the number of all possible ways of losing r out of p facilities, the size of IMF therefore grows rapidly as p and r increase resulting in long computation times. Real-world systems involving a large number of vulnerable facilities make it impracticable to apply.

With the purpose of handling larger-sized instances of IMF, the authors develop two different solution approaches for the resulting r-interdiction median problem with fortification (RIMF) in two subsequent works. The first approach presented in (Scaparra and Church, 2008) is based on a reformulation of the problem as a maximal covering model with precedence constraints (MCPC). They devise a solution technique for the reformulated problem which yields upper and lower bounds. The dimension of the model is reduced using a linear interpolation search procedure that exploits the properties of the coverage function. The resulting model can then be solved easily to optimality by CPLEX. Since the solution time is again sensitive to r, the method requires the enumeration of all patterns. The second approach provided in (Scaparra and Church, 2008) is a tree search algorithm that takes advantage of the bilevel formulation of the problem. In this study, an implicit enumeration algorithm to solve the bilevel integer programming (BIP) formulation of the RIMF is proposed for the first time. RIMF is described within a game theoretic framework as a leader-follower or Stackelberg game and formulated as a bilevel programming problem where the fortification problem (defender's problem) is the ULP and the interdiction problem (the attacker's problem) is the LLP which is RIC. Apart from large r values, the tree search algorithm surpasses MCPC solutions.

A recent working paper by Scaparra solves the stochastic version of RIMF referred to as S-RIMF. The authors address a maximum coverage type formulation for the S-RIMF (Scaparra et al., 2008). This formulation copes with a random number of losses and captures the uncertainty in the scope of man-made attacks. By using monotonically increasing and decreasing probability distributions, S-RIMF aims at minimizing the expected cost expressed as the probability weighted sum of the costs associated with the worst-case interdiction patterns for every feasible value of r.

Aksen et al. propose a budget constrained extension of the RMIF. Their new model, called as the budget constrained r–interdiction median problem with capacity expansion (BCRIMF-CE), adapts a BP formulation of RMIF and uses a budget constraint instead of a predetermined number of facilities to be protected (Aksen et al., 2010). The defender has budget for fortification, and protects any number of facilities as long as the budget allows. They also incorporate the capacity expansion at a unit cost that incurs at some facilities due to the reassignment of customers to these facilities following the interdiction of the attacker. The inclusion of the capacity expansion costs is added to both the defender and attacker's objective function. The authors solve the resulting bilevel programming model through an implicit enumeration algorithm applied on a binary tree similar to the solution methodology described in (Scaparra and Church, 2008) to evaluate the attacker's response.

To our knowledge, the first published study considering the triple problem of location, protection, and interdiction upon a median-type service network is due to (Aksen et al., 2009). This study elaborates on the facility location decision extension of the BCRIMF-CE of (Aksen et al., 2010). It is termed as the bilevel fixed charged location problem (BCFLP). To solve the resulting problem, the authors propose and compare three different methods. The first method is an optimal exhaustive search algorithm with exponential time complexity. The second one is a two-phase tabu search heuristic developed to overcome the first method's impracticality on large size problem instances. Finally, the third one is a sequential solution method in which the defender's location and protection decisions are separated.

The literature mentioned above consists of a small number of interdiction models that mainly concentrate principally on uncapacitated facility interdiction and protection although many real-life systems operate with a capacity limitation. The first attempt for handling capacitated facilities is due to (Scaparra and Church, 2010). They propose a model to optimize a limited amount of protection budget to reduce the adverse impact of a possible disaster via advanced protection of a set of existing capacitated facilities. The complexity of the resulting problem is increased by adding capacity constraints within a protection-interdiction model. Therefore, it is formulated as tri-level optimization model. By dualization of the lower problem, the model is reduced from a tri-level model to a bi-level model which can be handled by implicit enumeration algorithm developed in (Scaparra and Church, 2008).

The most recent extension to interdiction models is due to (Losada et al., 2011). This paper combines the concepts of the facility recovery time and the possibility of multiple disruptions over time. The problem is named as the fortification r-interdiction median problem with facility recovery time (FRIMT), and formulated via a bilevel mixed integer linear program (BMILP) for the fortification of an uncapacitated facility network. The goal of the problem is to harden or protect the critical components fully and to accelerate their recovery times. To solve the resulting problem, the authors propose three assorted decomposition techniques. These techniques not only solve medium to large size problem instances of FRIMT but also provide general a solution methodology for the interdiction models with some suitable alterations.

2.2. Bilevel Programming Problems

As can be seen, many interdiction models that have been developed to solve the most destructive case or to determine the best protection policy of a system with limited resources are formulated via bilevel programming (BP) formulation in the literature. BP problems are a special case of the multilevel optimization with two levels in which a subset of the variables in the upper level problems is constrained to be a solution of the lower level optimization problem. Interdiction problems are often described as a two person game (Stackelberg, 1952) where one level of the problem involves the system planner or defender's decisions about constructing his or her system by considering the effect of other player's (the follower) reactions. The final outcome is the best one for the leader.

The general BP problem is of the form:

$$\min_{\mathbf{x}} \; F(\mathbf{x},\mathbf{y}) \tag{2.1}$$

s.t.

$$G(\mathbf{x},\mathbf{y}) \leq 0 \tag{2.2}$$

$$H(\mathbf{x},\mathbf{y}) = 0 \tag{2.3}$$

where $\mathbf{x}$ is a given vector and $F(\mathbf{x}, \mathbf{y})$ is defined as the optimal objective value of the following problem:

$$\min_{\mathbf{y}} \; f(\mathbf{x}, \mathbf{y}) \tag{2.4}$$

s.t.

$$g(\mathbf{x}, \mathbf{y}) \leq 0 \tag{2.5}$$

$$h(\mathbf{x}, \mathbf{y}) = 0 \tag{2.6}$$

$$x_i \in \Re \qquad i = 1, ..., n, \tag{2.7}$$

$$y_j \in \Re \qquad j = 1, ..., n \tag{2.8}$$

where $\mathbf{x}$ is a vector of the upper level problem variables, and $\mathbf{y}$ is a vector of the lower level problem variables. $F(\mathbf{x}, \mathbf{y})$ is the upper level objective function, $H(\mathbf{x}, \mathbf{y})$ and $G(\mathbf{x}, \mathbf{y})$ are the upper level equality and inequality constraints, respectively. $f(\mathbf{x}, \mathbf{y})$ is the lower level objective function, $h(\mathbf{x}, \mathbf{y})$ and $g(\mathbf{x}, \mathbf{y})$ are the lower level equality and inequality constraints, respectively.

Unlike two person zero-sum games, these are sequantial games and the solution cannot be found via game theoretic solution procedures. The second player's problem called the lower level problem (LLP) takes part in the set of constraints of the leader's problem called the upper level problem (ULP). The ULP variables are fixed to exemplify a leader's feasible solution with the intent of obtaining the follower's optimal reaction scheme. In other words, when the ULP variables are assigned to fixed values, the optimal values of the LLP variables of the attacker are found. The final incumbent solution of the bilevel problem is the one giving the best ULP objective value among the set of feasible pairs of leader's action and the follower's optimal reaction.

By their nature, bilevel problems can be very difficult to solve as each level includes an objective that may be the exact opposite of the next. Both the levels of BP can be linear, integer, mixed-integer or nonlinear programming problems. When integer variables appear at both levels, it is inevitable that the problem becomes more difficult to solve. As a consequence, methods developed for the solution of the BP have so far addressed a very restricted class of problems in the literature. Moore and Bard propose a branch and bound type of enumerative solution for the solution of purely integer linear BP, namely BIP (Moore and Bard, 1990). Wen and Yang introduce another branch and bound technique for the solution of the mixed-integer BP where only the first level problem has discrete decision variables and the second level problem has continuous decision variables (Wen and Yang, 1990). Dempe develop cutting plane and parametric solution approaches to solve problem in which the second level problem has separable outer variables in its objective function (Dempe, 2002). Gümüş and Floudas suggest two approaches to solve the mixed-integer nonlinear bilevel problem to global optimality (Gümüş and Floudas, 2005).

3. PROBLEM FORMULATION

In this chapter we present our mathematical formulation to the BPIP problem. We acquaint some concepts and methodologies which are followed by the explanation of the mathematical model.

In the partial interdiction problem proposed in this study, two types of decisions are being made by two different agents called the defender and the attacker, whose objective functions cannot be weighted and aggregated into a single objective. The system operator (defender) determines what is left after an attack operates as efficient as possible, while the interdictor (attacker) decides what facilities to disrupt at what fraction. The problem is modeled as a bilevel programming (BP) problem that especially fits this kind of defender-attacker relationships. In the first level problem (i.e., the ULP), the attacker, who is also the leader, tries to choose which facilities to attack in order to reduce their capacities considering his limited budget so as to increase the operational cost of the existing system. An important assumption is that the number of the attacks depends on the limited budget of the attacker since each facility has a different cost of interdiction at full capacity. The attacker has perfect information so that when the allocation of the customer demand is determined by the defender (follower), it is also known by the attacker. In the second level problem (i.e., the LLP), the defender attempts to assign customers to the nearest facilities taking into account the remaining capacities of the facilities. The demand nodes are served by the system planner. Since additional outsourcing cost incurs due to the need of satisfying the overall demand of the customers, the system planner tries to minimize the unsatisfied demand.

It is important to mention that when the facility is interdicted by the attacker, a partial damage may be caused. This means that facilities will not be rendered totally inoperative after an attack, but will continue to provide service with less than their full capacity depending on the degree of interdiction. For instance, after an interdiction 60 percent of the capacity of the facility may continue to serve its customers as the rest of its capacity is damaged by the attacker. Another vital assumption is that a demand node can be served by only one facility. This means that the demand of a customer is either satisfied totally by a facility or its demand must be outsourced from a supplier.

In the model, index i and j represent respectively the ith customer and the jth facility. Parameters and decision variables are given below:

Index Sets:

- $\mathcal{I}$ = set of demand nodes (customers), $\mathcal{I} = \{1,2,...,n\}$
- $\mathcal{J}$ = set of existing facility sites (locations), $\mathcal{J} = \{1,2,...,m\}$

Parameters:

- a_i = demand of customer i
- c_i = cost of shipping customer i's unit demand per unit distance
- d_{ij} = Euclidean distance between customer i and facility j
- c_p = cost of outsourcing customer i's unit demand (independent of distance)
- e_j = cost of interdicting full capacity of facility j
- e_{tot} = total interdiction budget of the attacker
- q_j = capacity of facility j

Decision Variables:

$$V_{ij} = \begin{cases} 1 & \text{if customer } i \text{ is assigned to facility } j \text{ after attack} \\ 0 & \text{otherwise} \end{cases}$$

$$S_j = \text{fraction of facility } j\text{'s capacity lost due to attack}$$

Decision variables S_j of the ULP represent partial interdiction, whereas variables V_{ij} of the LLP indicate the reassignments of customers to facilities after attack.

The mathematical model is given as follows.

$$\max_{\mathbf{S}} \; Z_{\text{def}}(\mathbf{S}) \tag{3.1}$$

s.t.

$$\sum_{j\in\mathcal{J}} e_j S_j \le e_{tot} \tag{3.2}$$

$$0 \le S_j \le 1 \qquad j \in \mathcal{J} \tag{3.3}$$

where $\mathbf{S}$ is a given vector and $Z_{\text{def}}(\mathbf{S})$ is defined as the optimal objective value of the following integer problem:

$$Z_{\text{def}}(\widehat{\mathbf{S}}) = \min_{\mathbf{V}} \; \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_i c_i d_{ij} V_{ij} + c_p \sum_{i\in\mathcal{I}} a_i (1 - \sum_{j\in\mathcal{J}} V_{ij}) \tag{3.4}$$

s.t.

$$\sum_{i\in\mathcal{I}} a_i V_{ij} \le (1 - \widehat{S}_j) q_j \qquad j \in \mathcal{J} \tag{3.5}$$

$$\sum_{j\in\mathcal{J}} V_{ij} \le 1 \qquad i \in \mathcal{I} \tag{3.6}$$

$$V_{ij} \in \{0, 1\} \qquad i \in \mathcal{I}, j \in \mathcal{J} \tag{3.7}$$

In the above formulation, (3.1)–(3.3) correspond to the formulation of the ULP, and (3.4)–(3.7) represent the LLP. Expression (3.1) shows the leader's objective function. Constraint (3.2) is the budget constraint of the attacker for interdictions. Since the attacker has a limited budget for interdicting the facilities, and each facility has its own cost of interdiction, the number of the attacks is not fixed. In other words, the number of interdicted facilities cannot be known a priori, but depends on the attacker's budget and choices for attacking facilities. Constraint set in (3.3) is trivial lower and upper bound constraint on S_j.

The computation of the system cost after interdiction, $Z_{\text{def}}(\mathbf{S})$, requires solving the lower level problem. The S_j values obtained in the ULP serve as input parameters to the LLP. The defender's objective function in (3.4) is the same as the leader's objective function except the sense of optimization, which is maximization for the attacker and minimization for the defender. It consists of two components. The first component is the total shipment cost from facilities to customers, and the second one is the total penalty cost incurred due to unsatisfied demand. Recall that if a customer is not assigned to any facility, its demand must be outsourced. Increasing the outsourcing cost will be beneficial to the attacker as the operating cost of the defender is increased. Constraints (3.5) state the capacity restriction of the system planner. The term on the right-hand side of this inequality represents the remaining capacity after attack and takes constant value as the S_j variables are fixed at the ULP. In brief, constraints (3.5) ensure that if a facility is interdicted and some of its capacity is lost, the demand of customers assigned to this facility cannot exceed its remaining capacity. It is obvious that if there is no interdiction, the full capacity of facility j is used. The constraint set (3.6) maintains that some customers might not be assigned to facilities after attack due to the capacity reduction in the system. Finally, binary constraints on decision variable V_{ij} stated in (3.7) indicate the single-source assignment of customers.

4. SOLUTION PROCEDURES

BPIP contains continuous variables in the ULP and binary ones in the LLP. Hence, it is a mixed-integer bilevel programming problem that belongs to NP-hard problems (Moore and Bard, 1990). It is impossible to apply directly any procedure in the literature to solve BPIP. We therefore need efficient heuristic methods to produce high quality solutions to BPIP. To this end, we devise two search procedures: a progressive grid search and multi-start revised simplex search. Commonly, they consist of a search for the best facility interdiction ratio decisions of the attacker represented by the S_j variables in the ULP. Regarding their own rules, in fact, they traverse a larger portion of the entire solution space so as to yield better solution. The solution to the LLP is obtained through a general purpose IP solver, e.g. CPLEX. Finally, we develop an exhaustive search for full interdiction problems to evaluate the value of the partial interdiction problems.

In this chapter, we first present the details of our progressive grid search which is followed by the explanation of our problem specific simplex-based search procedure. Finally, we conclude this chapter with the working mechanism of the exhaustive search.

4.1. A Progressive Grid Search (PGS)

When an optimization problem has continuous variables like S_j, there are an infinite number of points in the domain, and this severely restricts the possibility of implicit enumeration even on small-sized instances. In this cases, performing a grid search can be a useful approach. Grid search divides the search space into equally positioned of points and evaluates the objective value at each point. If the domain is n-dimensional hyper-cube with length D on each side and the distance between grid points is K on each coordinate, then there are $(\lceil \frac{D}{K} \rceil + 1)^n$ grid points, where $\lceil x \rceil$ is the lowest integer greater than or equal to x. Hence, the number of function evaluations required to obtain the best solution grows exponentially in n.

The error in the approximation of the global optimum can be decreased by lowering the value of K. As a result, the number of grid points will be increased and the quality of the results will be improved. On the other hand, the exponential increase in the number of the function evaluations results in long computational time, especially for high dimensional problems. We, therefore, rectify PGS to avoid this increase with a selective positioning of the grid points, namely, the determination of the support points. Using PGS, the number of actual function evaluations is limited to the number of support points. The computational efficiency of this method is two-fold. Firstly, the number of support points used to find the final point is relatively small. Secondly, the set of support points are nested. This property reduces the number of required function evaluations by reusing the objective values of the support points that are calculated in the previous step. This also provides sampling refinement of grid points by selecting new support points around the best support point calculated so far.

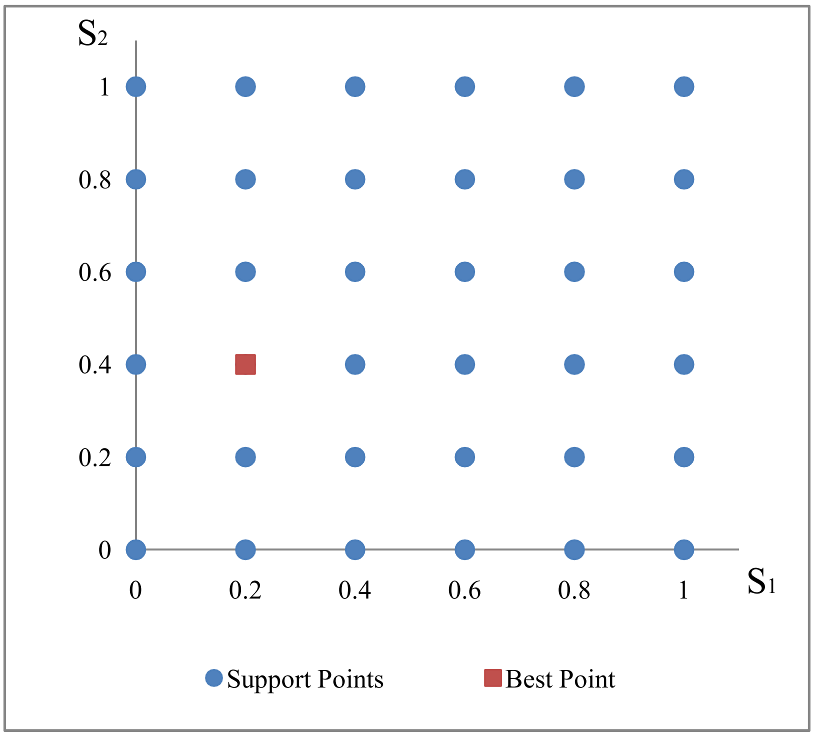

Figure 4.1. The first step of PGS

PGS consists of three steps. In the first step, K is set to 0.2. The search procedure is applied upon the ULP decision variables, i.e., S_j. n is equal to the number of the facilities, i.e. the parameter m, and $D = u_j - l_j = 1$, since $l_j = 0$ (lower bound of the S_j variable) and $u_j = 1$ (upper bound of the S_j variable). Therefore, six grid points are

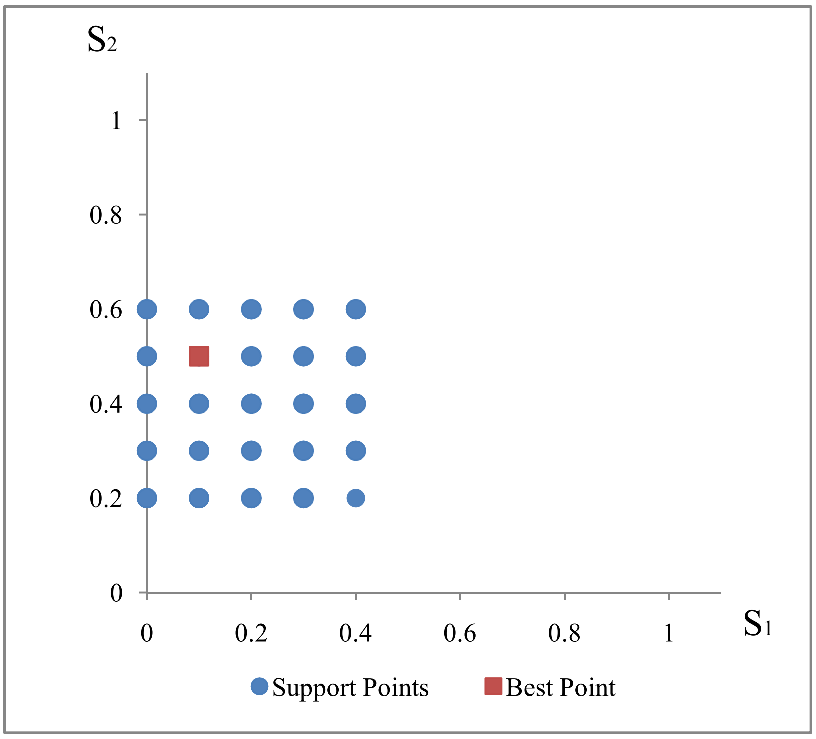

Figure 4.2. The second step of PGS

needed for each dimension, namely, for each facility. The first step of the search results in 6^m grid points to be evaluated. A grid point configuration, called a support point, represent the interdiction scheme of the attacker. They are used as an input to the LLP, and the objective values of each scheme is obtained by solving the resulting IP with CPLEX to optimality. Since we have a maximization problem, the support point with the maximum objective value is the best support point labeled as $\mathbf{S}^*$. It is used as the midpoint of the interval in which new support points are selected to be employed in the next step of the search. The support points of the second step are equally spaced in the new interval with length D'=2K with K = 0.1, and are constructed around the neighborhood of the best support point of the first step. $l_j = S_j^* - 0.2$, and $u_j = S_j^* +$ 0.2 in the second step. The number of the new support points in each dimension is five, so there are 5^m support points in total. After their objective values are evaluated, they are used for the determination of the best support point in the second step. The point with the highest objective value is assigned to $\mathbf{S'}^*$, and is used for the determination of the new support points as in the first step. In the last and third step, K = 0.05, l_j = $Sj'^* - 0.1$, and $u_j = Sj'^* +$ 0.1. The best support point obtained at the end of the evaluations is the solution of PGS.

Let us consider a two-facility problem. $K = 0.2$ and six points are equally spaced in the interval [0, 1] in the first step of PGS. Since $m = 2$, 36 points are obtained over the feasible region. All the combinations of the S_j values are sent to the LLP as an input parameter and their objective values are evaluated by solving them to optimality using CPLEX. Since we have a maximization problem, S_j values related to the best objective value are selected for the next step of PGS. Let us assume that the best support point is selected as (0.2, 0.4) (See Figure 4.1).

It is likely that the global optimum is in the neighborhood of this point. Then, the search procedure is repeated in the new interval with length $2K$ including as midpoints S_1^* and S_2^*. In this step, these new intervals are divided into four subintervals since $K = 0.1$ as in Figure 4.2.

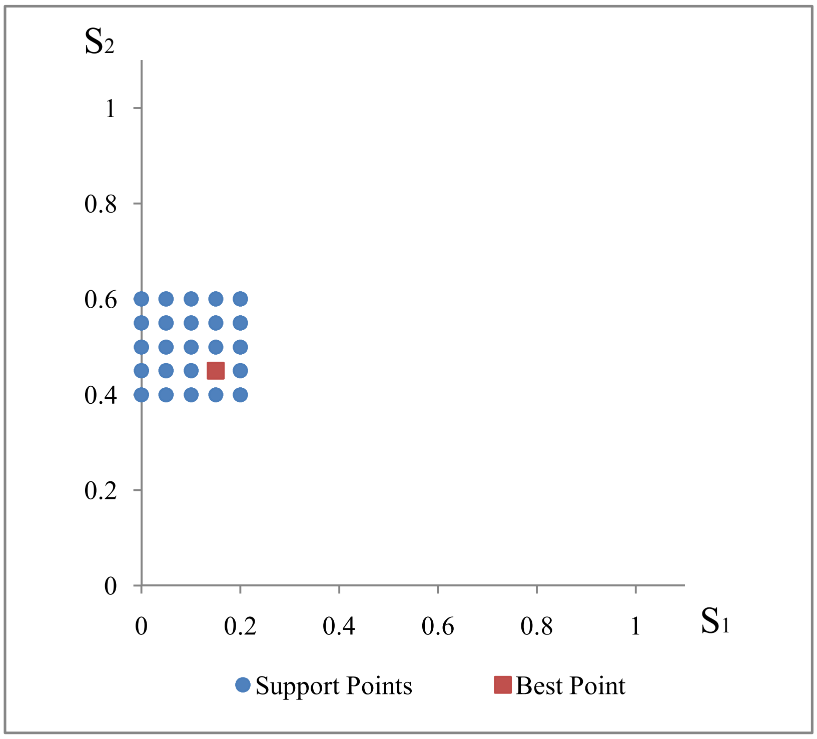

Figure 4.3. The third step of PGS

The new 25 support points obtained are evaluated as in the first step, and the best pair is again selected to start the next and last step of PGS. Let assume that the best support point is (0.1, 0.5). Since we know that the global optimum is likely to be in the interval (0, 0.2) for facility 1, and in the interval (0.4, 0.6) for facility 2, we divide these intervals into four subintervals by choosing $K = 0.05$ as in Figure 4.3.

The new support points are equally spaced of points over the new interval and their objective values are calculated. The best support point obtained at the end of this step is the solution of the PGS algorithm.

Due to the exponential nature, PGS can only solve small sized problems. Therefore, we perform this search on our problem instances in which the number of facilities varies from 4 to 10 as it spends more than 3 days to solve an instance with $m = 10$.

4.2. Multi-Start Revised Simplex Search (MS-RSS)

The use of efficient heuristics to obtain solutions for the BPIP in a reasonable computational time becomes inevitable especially when the problem size increases towards realistic values. Motivated by this fact, we devise a simplex-based search heuristic called Multi-Start Revised Simplex Search (MS-RSS).

Let us consider unconstrained continuous optimization problem:

$$\max f(\mathbf{x}) \tag{4.1}$$

$$\text{s.t.}$$

$$-\infty \leq \mathbf{x} \leq +\infty \tag{4.2}$$

where $\mathbf{x}$ is a vector of continuous variables, and $f(\mathbf{x})$ is an objective function.

It is often desired in many fields such as mathematics, physics, and engineering to solve these types of problems via calculus. As real life problems are quite complex with many variables, the derivatives of their functions are likely to be difficult or impossible to evaluate in a form suitable for calculus. In the case where the gradient of the function is not available, the optimal value of the problem is usually estimated via calculation of its values at several points. Based on this idea, we construct a new method upon the Nelder-Mead Simplex Search (NMSS) which is the most popular direct search method for minimizing unconstrained functions.

The NMSS, also known as amoeba, generates a sequence of changing simplices, which are modified so that the simplex adapts itself to the local landscape. The name amoeba stems from the oozing behavior of the simplex as it traverses the landscape. Three basic construction principles are defined to determine a new point of the simplex: expansion, reflection, and contraction. Each of these principles is designed to help the simplex expand itself along the direction of improvement and contract itself where improvement is not found.

In fact, the simplex search method of Spendley, Hext and Himsworth is the first method for non-linear simplex search. It relies on the comparison of function values at the $n+1$ vertices x_i of a simplex (where n is the dimension of the problem), and has the minimal number of rules for simplex manipulations between iterations (Spendley et al., 1962). This simplex search method is modified by Nelder and Mead by adding expansion and contraction operations (Nelder and Mead, 1965). The underlying idea of these modifications is rooted in the following observation: the performance of the search procedure is improved when the simplex is allowed to adjust its shape with the curvature of the function.

NMSS has been used in a wide variety of context, especially in chemistry and chemical engineering. Despite its success, NMSS and other simplex methods have not received much attention within operations research community because of the ad-hoc nature for long period. There are few studies which describe the theoretical properties and problems that simplex methods cannot solve. The first theoretical study about NMSS is provided in (Gurson, 2000). In this study, simplex-based direct search methods are analyzed and the small ambiguities in these algorithms are resolved. In recent years, theoretical analysis has lead to modifications in the NMSS algorithm. In (Wolff, 2004), NMSS is used for finding global minima of non-linear optimization problems with simple bounds and non-linear constraints. Luersen et al. introduce a set of restart options and three convergence criteria to detect whether the bounded NMSS procedure is failed (Luersen et al., 2004). The phase concept is added into the NMSS procedure by (Zhao et al., 2009). In this article, NMSS is altered such that the search is restarted by constructing an initial simplex around the solution obtained in the previous phase to avoid being trapped in local minima. Significant factors that affect the overall efficiency of the search procedure are identified as the size of the initial simplex and the adjustment of the shrinkage coefficient.

4.2.1. Basic Principles of the Nelder-Mead Simplex Search

In this section, a step-by-step description of the NMSS is provided since our proposed algorithm MS-RSS is constructed upon the rules of this method. It is worth noting that the original NMSS is devised for minimization problems and can be modified for maximization problems with a little effort. As BPIP is a maximization problem, we explain the basic principles of the NMSS procedure with respect to the maximization types of problems by exemplifying with a two-dimensional NMSS.

NMSS starts with a non-degenerate simplex of $n+1$ vertices, which are selected randomly in $\Re^n$. By a non-degenerate simplex, we mean a set of $n+1$ vertices in $\Re^n$ with the property that the set of simplex edges adjacent to any given vertex spans $\Re^n$.

At the beginning of an NMSS iteration, vertices are labeled as $\mathbf{x}_0$, $\mathbf{x}_1$, ..., $\mathbf{x}_n$ such that $f\ (\mathbf{x}_0) \leq f(\mathbf{x}_1) \leq ... \leq f(\mathbf{x}_n)$, where $\mathbf{x}_0$ denotes the point with the lowest objective value $f(\mathbf{x}_0)$ among the all vertices, whereas $\mathbf{x}_n$ refers to the point that has the highest objective value $f(\mathbf{x}_n)$.

In the maximization problem, it is desirable to replace $\mathbf{x}_0$ with a point whose objective value is higher than $f(\mathbf{x}_0)$ in order to reach an optimum. As a result, three construction principles are defined to determine a new point of the simplex. Each of these principles is designed to help the simplex better follow the gradient of the function. The intention of NMSS algorithm is to pull the simplex toward the region of interests in the solution space by expanding it along the direction of improvement and by contracting it in the opposite directions when improvement is not achieved.

Let $H \subset \Re^n$ be a simplex with $n+1$ vertices. The centroid $\overline{\mathbf{x}}$ of all vertices in the simplex with respect to $\mathbf{x}_0$ is calculated using the formula:

$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n+1} x_i$

The first construction principle is the reflection of the corner $\mathbf{x}_0$ at $\overline{\mathbf{x}}$. $\mathbf{x}_0$ is reflected through $\overline{\mathbf{x}}$ to obtain a new point $\mathbf{x}_r$. This step always provides another non-degenerate simplex consisting of n of the original vertices and $\mathbf{x}_0$'s reflection point. The simplex will be flipped towards the region of interest, thereby locating a valid local maximizer. $\mathbf{x}_r$ is determined from:

$\mathbf{x}_r = \overline{\mathbf{x}} + \alpha(\overline{\mathbf{x}} - \mathbf{x}_0)$ with reflection coefficient $0 \leq \alpha \leq 1$.

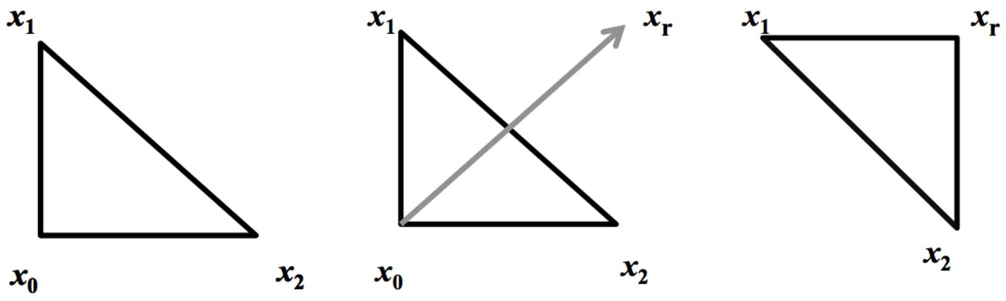

Figure 4.4. Reflection step of NMSS

The visualization of the reflection step for a two-dimensional NMSS can be seen in Figure 4.4. Vertex $\mathbf{x}_0$ has the lowest objective value (left), therefore it must be reflected through $\overline{\mathbf{x}}$, which is the centroid of $\mathbf{x}_1$ and $\mathbf{x}_2$, to a new point $\mathbf{x}_r$ (center). The new non-degenerate simplex including $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_r$ is then generated (right).

The second type of simplex alteration is the expansion of the corner $\mathbf{x}_0$ in the direction of $\mathbf{x}_r - \overline{\mathbf{x}}$. The new point $\mathbf{x}_e$ is determined from $\mathbf{x}_e = \mathbf{x}_r + \gamma(\mathbf{x}_r - \overline{\mathbf{x}})$ with the expansion constant γ>1. The purpose of the expansion step is simple. When the reflection point has a higher objective value than $f(\mathbf{x}_0)$, it could be beneficial to continue sampling the function in the same direction. Therefore, simplex is expanded outward from $\overline{\mathbf{x}}$ in the direction of plausible function increase (See Figure 4.5).

In some cases, the reflection and expansion steps do not improve upon the current highest objective value, resulting the third construction principle: contraction. The simplex is contracted in one of the three ways. The first way is the partial interior contraction of $\mathbf{x}_0$ in the direction of $\mathbf{x}_0 - \overline{\mathbf{x}}$. $\mathbf{x}_0$ is more likely to be closer to maximum

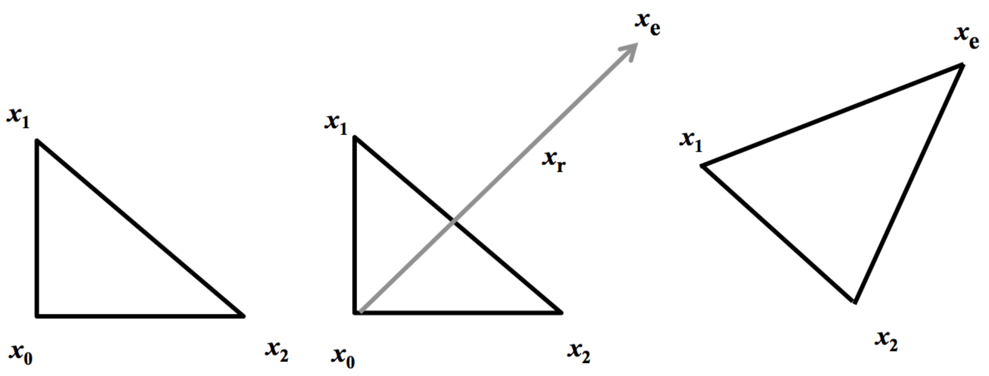

Figure 4.5. Expansion step of NMSS

point if the objective value of $\mathbf{x}_0$ is more than that of the reflection point $\mathbf{x}_r$. The simplex is then contracted from $\overline{\mathbf{x}}$ toward $\mathbf{x}_0$ as illustrated in Figure 4.6. The new contraction point $\mathbf{x}_{ic}$ is calculated from $\mathbf{x}_{ic} = \overline{\mathbf{x}} + \beta(\mathbf{x}_0 - \overline{\mathbf{x}})$ with the contraction constant $0 \leq \beta \leq 1$.

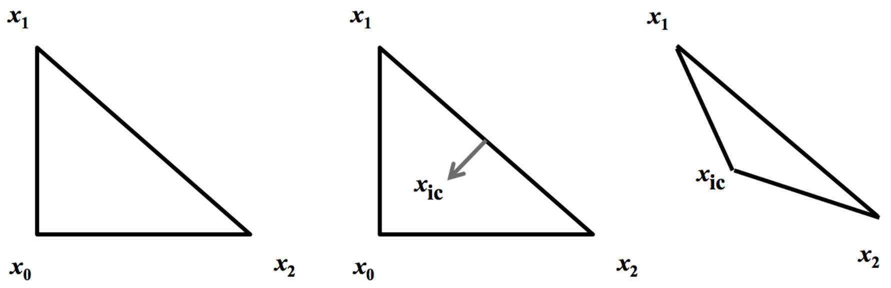

Figure 4.6. Partial interior contraction step of NMSS

If this is not the case and $\mathbf{x}_r$ has the objective value more than $\mathbf{x}_0$ has, a partial exterior contraction is applied. In this type of contraction step, the simplex is contracted from $\overline{\mathbf{x}}$ toward $\mathbf{x}_r$ (See Figure 4.7). The new contraction point $\mathbf{x}_{oc}$ is determined from $\mathbf{x}_{oc} = \overline{\mathbf{x}} + \beta(\widehat{\mathbf{x}} - \overline{\mathbf{x}})$ with the contraction constant $0 \leq \beta \leq 1$.

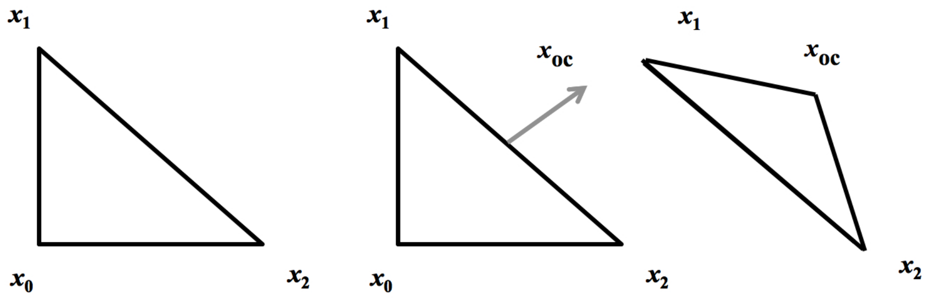

Figure 4.7. Partial exterior contraction step of NMSS

Finally, as a third way, if no improvement has been found at the end of the two contraction steps mentioned above, total contraction to $\mathbf{x}_n$ is applied. The simplex is shrunk toward $\mathbf{x}_n$, namely, the point with highest function value $f(\mathbf{x}_n)$. As can be seen in Figure 4.8, all the points in the simplex will be replaced by $\mathbf{x}_i$' $= \delta(\mathbf{x}_n - \mathbf{x}_i)$

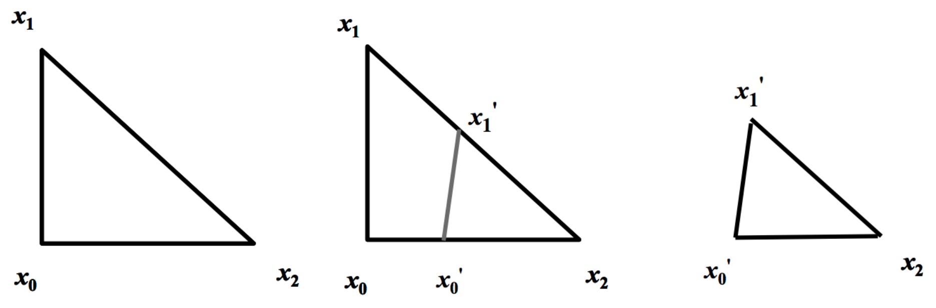

Figure 4.8. Shrinkage step of NMSS

4.2.2. Problems with NMSS and RSS

Having capacitated facilities and continuous variables rather than integer variables, we encountered many problems in the application of the original NMSS during our preliminary experiments. Although some modifications in the literature have improved the performance of the original NMSS method to some extent, we are faced with many obstacles such as being trapped in local optima, reaching a plateau, and simplex initialization problems that have served as major impediments or barriers to progress in this research. Therefore, we have to make a lot of modifications in the existing procedure and also develop some algorithms. In this section, these common problems in simplex-based search algorithm are addressed.

4.2.2.1. Bounded Spaces. The basic simplex algorithm assumes that vertices of the simplex have inputs that are valid for $(-\infty, +\infty)$. Therefore, the new vertex obtained by the move operations may take any value in the search space. On the other hand, in our problem, S_j variables are bounded. In order to deal with this problem, some straightforward solutions are developed in the literature. One of them is the assignment of a large negative (positive) value to the objective function of the point that violates the boundary condition in the maximization (minimization) problem (Brooks, 2000). Simplex search is then forced to immediate contraction as the new point is worse than the current worst. After a number of contractions, the simplex becomes small enough for reflections and cannot replace its points with the ones outside the boundary. As a result, it is a useful solution although it may take more iterations to find optima lie on or near the boundary. However, applying this strategy may cause creating an infinite plateau extending beyond the boundary and pulling the simplex towards this plateau. As a result, simplex can be away from the region including the optimum.

Another strategy is to protect boundary-violating points on bounds (Wolff, 2004). The reflection, expansion and contraction moves of the simplex search algorithm can be modified so that the points taking value beyond the boundary after the moves are placed as a new point on the boundary. This strategy avoids pulling the simplex towards the plateau. The danger with this approach is the collapse of the simplex along some dimension, causing unreachable parts of the search space.

In this study, a new strategy is developed as a third way for dealing with bounded space problem. It is the combination of projecting the points beyond the boundary on bounds and using the adaptive linear penalty function method. The move operations of the NMSS are used for searching the solutions of the ULP whose decision variables (i.e., S_j) are bounded between 0 and 1. Considering the new points obtained after an MS-RSS iteration, if a boundary-violating point has a value less than zero, then it is projected on the lower bound taking value 0, whereas when that point takes a value larger than one, then the related Lagrangian multiplier is updated causing a penalty cost affecting the objective value adversely. Using the penalty method for the upper bound violations rather than the projection methods provides a superior solution for BPIP as it preserves the new simplex against collapsing. The bounded N-M offers just the projection of the boundary-violating point on the boundary (Wolff, 2004). Our preliminary experiments showed that applying only this strategy, in many circumstances, may result in the loss of the dimension in the simplex. In other words, the collapse of the simplex along one or more dimension is very likely as explained in the following.

4.2.2.2. Collapsing Simplex. Another problem that can arise while using a simplex-based search algorithm is the case when all the points in the simplex have the same value in the same dimension. All reflections and contractions will occur along the line or hyper-plane formed by these points forcing the rest of the search space to be unreachable.

For example, let us assume that a simplex search is performed on a two dimensional problem. Therefore, three points with two components are needed for the construction of the simplex. Consider that the second components of these points have the same value. All contractions and reflections will occur along the line formed by these points in the current simplex.

To avoid this problem, two possible strategies can be embedded in the simplex search algorithm. The first strategy is to modify the algorithm such that when the collapse of the simplex is detected, reflection or contraction is applied with the second-worst point rather than the first one. In fact, the detection of the collapsing simplex, i.e, collinearity check, is also another problem. All the vertices in the simplex have to be checked after each reflection and contraction to decide whether any points are collinear. This is an expensive operation especially when a search space is high dimensional. Comparing the new point obtained as a result of reflection or contraction operations to each point in the current simplex is a simple method requiring less operation and time.

The second strategy is to select a new point randomly when all points have the same value in the same dimension. Since the randomly assigned point will have a different value for the problematic dimension, all contractions and reflections will not occur along the line or the hyperplane formed by these points. This solution requires less effort as there is no need for checking the collinearity.

4.2.2.3. Premature Convergence.

Another problem that simplex-based search algorithms are susceptible to is the premature termination at a local optimum. If a vertex of the simplex reaches at a local optimum in the solution space, the algorithm will try a reflection. This action will not help remove the worst point in the current simplex, and then, contraction along one dimension will be attempted. This operation will also not be able to change the worst point, the simplex will shrink along multiple dimensions repeatedly, decreasing the size of the simplex. Checking whether the norm of the vector between the point with the highest objective value and the other points in the current simplex are within a given precision reveals that all the points are within the basis of attraction of an optimum and that the simplex is converged to that local optimum rather than global optimum.

In order to improve the chance of obtaining a better solution, random restart technique, commonly used within hill-climbing algorithms, can be incorporated into NMSS procedure. To guard against premature convergence, the most effective solution is to step away from the current termination point, restart the search procedure with a new larger simplex, cache the value once an optimum is found, and compare these resulting alternative termination points. As the number of restarting grows, the solution quality is increased along with the computational time rise.

4.2.2.4. Plateau Problem. Getting caught in a plateau is another problem that weakens the efficiency of the simplex-based search algorithms. Consider the situation in which all $n+1$ vertices have the same objective value f. When these points are reflected, it is possible that the new point has the same objective value f implying that each of these points maps to a point on the plateau. The new point will be generated from these points at the next iteration as the simplex search algorithm has no memory beyond the current simplex points. Consequently, the simplex might visit the same points repetitively. It is obvious that after some reflection operations, the new points may lead away from the plateau. However, when inappropriate points are initially chosen for the operations of the algorithm, plateau problem can still take place.

Counting the reflections with no improvements is suggested as a conservative approach to cope with the plateau problem (Brooks, 2000). A counter can be added to the basic algorithm. As soon as the objective values of the $n+1$ vertices are evaluated and they have the same objective value f after the reflection step, the counter is adjusted. Nonetheless, our preliminary experiments reveal that the occurrence of the plateau problem is not restricted to the reflection step. To this end, we expand this check for any move of the search. If the new point obtained after the moves of the simplex search produce a different value than that of the n+1 vertices have, the counter is set to zero. Otherwise, it is incremented. Once the counter reaches n+2, a plateau problem is detected. If this is the case, the search obviously ends up with the objective value f. Hence, selecting a new set of random simplex points to restart the algorithm may be a very competent policy.

4.2.2.5. Cycling. Cycling can occur in the reflection step. Consider a two-dimensional simplex search in which three vertices are needed for a simplex. Suppose that $\mathbf{x}_0$ has the lowest objective value $f(\mathbf{x}_0)$, and the reflection step is chosen to generate a new simplex point x_3 at iteration φ. The resulting simplex of the iteration φ, which is also the initial simplex of the next iteration, consists of $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$. At iteration φ+1, in some cases, it is quite possible that $\mathbf{x}_3$ can have the lowest objective value and that the reflection step might be chosen. After the reflection of $\mathbf{x}_3$, the simplex at iteration φ is reached again since the new point of the simplex is $\mathbf{x}_0$. As a result, the current simplex will be reflected again to $\mathbf{x}_3$ at iteration φ+2 again, causing a cycling of the simplices.

For the sake of preventing the search from going back to the same solution, the point reflected is not allowed to reflect at the next two iterations. Hence, the second worst point rather than the worst point is used for the reflection move.

4.2.3. Significant Factors Affecting the Performance of the NMSS algorithm

In this section, two significant factors that affect the performance are described. The initial simplex size and the shrinkage coefficient δ are identified as significant factors that reduce the drawbacks of the N-M procedure (Humphrey and Wilson, 2000).

<u>4.2.3.1. Size the Initial Simplex.</u> In a simplex search algorithm, starting with a larger initial simplex generally gives better results. Although simplex can be far from the optimum, when the procedure is started with a larger simplex, more ground of the search space will be covered as a result of each operation of the NMSS algorithm and any errant contraction or shrinkages will have less severe impact on reaching the true optimum. In contrast, if the procedure is started with a smaller simplex, the algorithm has to iterate many times when the initial simplex is started far from the true optimum. The procedure would become more susceptible to any errant contractions or shrinkages. When the simplex is near the optimum, the progress of the procedure towards the optimum would be slowed from errant operations significantly. Considering the facts mentioned above, taking the initial step size parameter υ larger might improve the overall performance of the procedure.

<u>4.2.3.2. The Shrinkage Coefficient.</u> By performing the shrinkage operation, each edge of the simplex is rescaled by shrink coefficient δ. Since $0 \leq \delta \leq 1$, shrinkages reduce the size of the simplex remarkably, and protecting against errant shrinkages might result in better performance of the NMSS method. Humphrey and Wilson have been observed that a protection against premature convergence can be obtained by increasing the value of the shrinkage coefficient linearly when a restarting procedure is performed (Humphrey and Wilson, 2000).

During the earlier simplex search iterations, the current simplex is usually far from the global optimum. Any errant shrinkage will have a less severe effect on the procedure. If the procedure detects that a shrinkage operation must be performed, the smaller value of δ allows this operation to be more successful in pulling the simplex towards a locally convex neighborhood of the optimum. After several errant shrinkages during the earlier search iterations, however, simplex becomes too small to make effective progress towards the optimum. When restarting procedure is applied in the search procedure, the formation of the new initial simplex around the solution obtained in the previous phase may result in a higher likelihood of performing errant shrinkages since the new simplex is usually constructed in a subregion of the region of interest. In the later phases of the search, it is therefore advantageous to increase the value of the shrinkage coefficient δ linearly to avoid significant reduction in the size of the simplex.

4.2.4. Adaptive Linear Penalty

The original NMSS is developed for unconstrained and unbounded optimization problems. Real-life problems, however, might contain many constraints and bounded variables. The penalty method can be an effective approach for handling constraints and bounded variables for the modifications of NMSS.

Penalty method adds penalty terms to the objective function. Every time a constraint is violated or a bounded variable takes value beyond its upper bound, the objective function is penalized. As a result, solutions will not leave the feasible domain.

Let us consider the problem

$$\max \ f(\mathbf{x}) \tag{4.3}$$

s.t.

$$g_i(\mathbf{x}) \leq 0, \qquad i = 1, 2, ..., m \tag{4.4}$$

$$h_i(\mathbf{x}) = 0, \qquad i = m+1, m+2, ..., n \tag{4.5}$$

$$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \tag{4.6}$$

where the vectors $\mathbf{l}$ and $\mathbf{u}$ represent lower and upper bounds on variables, g_i and h_i the inequality and equality constraints, respectively.

This problem can be written in an unconstrained penalized form as

$$\max f(\mathbf{x}) + \sum_{i \in \mathcal{I}} \lambda_i \ P_i(\mathbf{x})$$

where P denotes relative penalty cost and λ penalty parameter.

If the ith constraint is violated, relative penalty cost $P_i(\mathbf{x})$ will be less than zero. Otherwise, it will take a value of zero meaning that no penalty has occurred for the related constraint.

Appropriate values of penalty parameters λ_i need to be estimated. They are usually initialized as zero. When the related constraints of the new changed simplex point x_{new} are violated, the penalty parameters of these constraints are updated using the formula:

$$\lambda_i^{new}{=}\lambda_i{+}s{\times}P_i(\mathbf{x}),\ i = 1{,}2{,}...{,}m$$

where s is a positive step size. Since s and $P_i(x)$ are strictly positive, $\lambda_i^{new} \geq \lambda_i \geq 0$.

Considering the unlimited growth of the penalty costs, the constraints $g(\mathbf{x}) \leq 0$ and $h(\mathbf{x}) = 0$ are less violated for growing number of iteration.

Step size s may influence the performance of the overall procedure. If a small step size is chosen, the penalty parameters will not be increased fast enough and the optimum will be found in a region with strongly violated constraints. It also takes many iterations. On the contrary, a large step size s may let NMSS fail in generating a well designed simplex. Consequently, a series which will be a constant small value and a large one for certain number of iterations can be used for step size.

Let s_s denotes the standard step size and s_l a large step size with $0{<}s_s{<}s_l$. The step size of the penalty parameter is then determined from

$$s_\varphi \quad = \quad \begin{cases} s_l & \text{if } \varphi_s \text{ and } \varphi \text{ mod } \mid \varphi_s \mid = 0 \\ s_s & \text{otherwise} \end{cases}$$

where φ refers to the iteration number and φ_s a fixed number of iterations in which small step sized will be used while updating the multipliers.

4.2.5. Multi-Start Revised Simplex Search Procedure

As mentioned before, NMSS can fail to converge or may converge to a non-optimal solution. A multi-start strategy in which randomly determined points used as starting points of our single simplex-based search can lead to substantial improvements in the efficiency. It should be noted that the success of this strategy heavily relies upon the solution quality of single starts. Many users consider that a random restart technique, commonly used within hill-climbing algorithms, is the easiest fix to improve the chance of obtaining better solution as it guards against premature convergence. Our proposed algorithm is actually based on the Revised Simplex Search (RSS) of Humphrey and Wilson (2000). RSS is developed for unconstrained and unbounded problems and is probably the most efficient modification of NMSS. It consists of three-phase application of the NMSS method where: i) the search is restarted such that the ending values for the one phase is the starting values for the next phase, ii) the size of the initial simplex is decreased geometrically while the shrinkage coefficient is reduced progressively to provide adequate protection against premature convergence, iii) the final estimated optimum is taken over all phases of the search procedure.

Due to capacitated facilities and continuous variables, the original RSS algorithm cannot be applied directly to the BPIP problem. It is therefore modified by including some additional modifications. In MS-RSS, this modified RSS is started K times from different randomly selected points. The exact reason for the multi-start strategy is the observation that although the modified RSS procedure is started three times, the final estimates of the global values are not well enough in most of the cases. Therefore, the modified RSS is restarted K times. The best solution of each start is cached once the incumbent of the three phases is found. These values are compared to each other in order to find the optimal solution of the overall procedure, in other words, the final estimated optimum is the best of the ending values of K RSS runs.

The proposed procedure is given in the following.

4.2.5.1. Initial Simplex Generation. One of the weaknesses of direct search methods is excessive sensitivity to starting values and premature termination at a local optimum. Due to the high possibility of being trapped in a local optimum, it is of great importance to initialize the first simplex of MS-RSS from a good starting point to improve the performance of the solution procedure.

An efficient action is to generate initial simplex from the different part of the search space at each restart. With this purpose, we devise two initial vertex generation methods which are based upon the start number of MS-RSS. We let $\mathbf{S}_j$ denote the vertex of these generation methods for j =1,. . . ,m+1 where $\mathbf{S}_j$ is the m dimensional vector with elements $s_{i,j}$ for $i = 1, 2, \ldots, m$. Note that, the interdiction fraction $\mathbf{S}_j$ corresponds to $\mathbf{x}_i$ which is used in definition of NMSS method. Similarly, m, the number of the existing facilities, is used in our problem instead of n in NMSS definition. The working mechanisms of the initial simplex generation methods are as follows.

In the first and the second start of MS-RSS, a deterministic initial simplex vertex generation is used. It is worth noting that $\mathbf{S}_j$ is the m dimensional unit vector with elements $s_{i,j}$ which are one in the jth component and zero elsewhere for $i = 1, 2, \ldots, m$. $\mathbf{S}_j$'s are used as an input parameter of the LLP, and then, the objective functions are calculated by solving the LLP with CPLEX, and sorted in non-decreasing order. This generation method lies on the logic that an intelligent interdictor would first prefer to hit the facility the loss of which costs the maximum increase in the objective value, and then, would proceed with the remaining facilities which causes the maximum harm as long as his budget allows.

When the starting number is more than two, the initial vertex is generated via double randomization scheme. Both the facility to be attacked and the ratio of the interdiction are determined randomly. The aim of double randomization is to provide intensification and diversification of the search space. Obtaining the index i randomly, elements $s_{i,1}$ are also determined randomly for i = 1,2,. . . ,m to construct the initial vertex S_1. If the budget underutilization is detected, $s_{i,1}$'s are increased one by one with $\Delta_{i,j}$ which is determined via formula: $\Delta_{i,j} = \min\{ 1 - \mathbf{s}_{i,j} , \frac{e_{tot} - e_{used}}{e_j} \}$ where

e_{used} is the current budget used to interdict facilities before the current $s_{i,j}$ increment. After increasing $s_{i,j}$ values with $\Delta_{i,j}$'s, their new respective objective function values are sorted in non-decreasing order. Starting from the $s_{i,j}$ values with the maximum increase in the objective value toward that with less increase, $\Delta_{i,j}$'s are added until no budget is left.

At the end of deterministic and double randomization vertex generation methods, the initial vertex $\mathbf{S}_1$ is generated. It is customary to specify a starting point in $\Re^n$ that is taken as one of the initial simplex vertices in the lack of information about the function. The remaining vertices of the simplex are then produced with respect to $\mathbf{S}_1$ in one of the two ways: Haftka and Gürdal's method and the standard unit vector method explained in Humphrey and Wilson, 2000.

When the start number is an odd number, Haftka and Gürdal's method is used by initializing a simplex of size a at $\mathbf{S}_1$ based on rule

$$\mathbf{S}_j = \mathbf{S}_1 + p\ \mathbf{e}_i + \sum_{k \in \mathcal{I}} q\ \mathbf{e}_k$$

where $\mathbf{e}_i$ are the unit base vectors and

$$p = \tfrac{a}{n\sqrt{2}}(\sqrt{n+1} + n - 1)$$

$q = \tfrac{a}{n\sqrt{2}}(\sqrt{n+1} - 1)$ where $a = 1$ in our problem. Applying these formulae, $\mathbf{S}_j$ may have some components larger than one which violates the upper bound of the ULP variables resulting in an infeasible solution. Since any component of the vertex j cannot exceed one, we normalize its components by dividing them with its largest component. This act guarantees a boundary feasibility, but not budget feasibility. Therefore, after the normalization step, if budget is exceeded, we scale the components of the vertex j by dividing them by the scalar product of unit interdiction costs and interdiction ratios of facilities to obtain budget feasible solution.

For even start numbers, the standard unit vector method that generates the initial simplex by perturbing the starting point by a specified step v along m coordinates is used as explained in (Humphrey and Wilson, 2000). In terms of the initial vertex $\mathbf{S}_1$, the remaining vertices of the initial simplex are given by $\mathbf{S}_j = \mathbf{S}_1 + v\mathbf{e}_i$ where v is 1 in our problem.

As this formulation would end up with boundary violating components, the same scaling methodology used in Haftka and Gürdal's method is applied. By applying these two simplex generation methods in MS-RSS, we obtain K different initial simplices. Each simplex includes m+1 vertices each of which consists of m elements. As soon as an initial vertex is generated, MS-RSS continues with the single MS-RSS run which is explained in the following paragraphs.

4.2.5.2. A Single MS-RSS Run. It includes three phases indexed by the phase counter τ. Within each phase, a new simplex is generated via the reflection, expansion and contraction operations of the N-M procedure as described in Section 4.2.1 in each additional iteration φ.

In the initialization step of the modified RSS, phase counter τ and iteration counter φ are set to 1. As recommended by Nelder and Mead (Nelder and Mead, 1965); the reflection, expansion, contraction and shrinkage coefficients in the NMSS iterations should satisfy $\alpha > 0$, $\beta > 1$, $0 < \gamma < 1$, and $0 < \delta < 1$, respectively. Chooses for these values are $\alpha = 1$, $\beta = 2$, $\gamma = 0.5$, and $\delta = 0.5$ in our procedure as they are universal values. Considering the number of the restart, a deterministic or double randomization method is used to yield the initial vertex of the starting simplex. It is noteworthy that the formula used in the determination of the remaining vertices of the initial simplex depends on the number of the restart. Once the initial simplex is generated, the centroid of the vertices is calculated as in Section 4.2.1.

$\mathbf{S}_j \equiv [s_{1,j}, s_{2,j} \ldots, s_{m,j}]$ denote the jth vertex of the current simplex.

At the beginning of any iteration, the objective values of m+1 points of the current simplex are calculated and sorted in non-increasing order. In this step, these values are checked for the plateau problem which let the search method fail in visiting the same points repeatedly. When they are the same for all vertices of the simplex, the plateau counter ω is incremented by one. ω is only set to zero in case the new point obtained as the result of search operations yields a different objective value from that of m vertices have. After ω reaches $m+2$ which indicates that no improvements are obtained during the last $m+2$ iterations, the simplex is detected to be caught in a plateau. Hence, the new phase is started (as described below). On the other hand, the iteration number φ is increased by one and a suitable move operation of the original NMSS is performed.

RSS is developed for unconstrained and unbounded optimization problem. On contrary, BPIP contains budget constraint and bounded variables like S_j's. The combination of the two approaches is used to adapt the attacker's budget constraint and bounded variables. Considering the new points obtained after simplex search moves, if a boundary-violating point has a value less than the lower bound of the variable, then it is projected on the lower bound. When that point takes value more than the upper bound and/or the budget constraint is violated, the adaptive penalty method is applied and the related Lagrangian multiplier is updated causing a penalty cost affecting the objective value of the problem adversely.

Cycling check is applied after the reflection move operations. If a point is reflected, it is not allowed to be reflected in the next two iterations. The second worst point is reflected in the case the worst point cannot be reflected.

4.2.5.3. Termination Criterion. To complete a simplex-based search algorithm, termination criteria should be defined. Since multi-phase and multi-start policies are applied, we use three termination criteria for the current phase, and one for a single RSS run, and finally one for the overall procedure. After each reflection, expansion, and contraction, the volume of the simplex is controlled as

$$\max_{0\leq i\leq n} \|\mathbf{S}_i - \mathbf{S}_{\max}\| \leq \begin{cases} \varepsilon_1 \|\mathbf{S}_{\max}\|, \; if \; \|\mathbf{S}_{\max}\| \neq 0 \\ \varepsilon_2, otherwise \end{cases}$$

where ε_1 and ε_2 are user-specified tolerances. The maximum number of the iterations φ_{max} is also introduced to determine the termination of the current phase as the second usual criterion. The third and the last one is the detection of a plateau problem. In the case that these termination conditions are not satisfied, φ is incremented and one iteration of single RSS run is performed. Otherwise, the best point of the last iteration is recorded as the termination point of the current phase ($\mathbf{S}^*$), phase counter τ is incremented. It is expected that the attacker depletes his or her all budget in the optimal solution, however, this is not always the case in the preliminary solution of the experiments. After the termination of each phase $\mathbf{S}^*$ values are increased as

$$\mathbf{S}^* = \mathbf{S}^* + \min\{\ 1\text{-s}_{i,j}\ ,\ \frac{e_{tot}-e_{used}}{e_j}\ \}$$

These values are sorted in non-decreasing order. The same scaling methodology used in Haftka and Gürdal's method is applied. After the termination of a phase, a new phase is started following the settings described below.

At the beginning of a new phase, the iteration number φ, the plateau counter ω and penalty parameters λ_i's are set to zero. The search is restarted by initializing a new and larger simplex around the best solution found in the previous iteration. To manage this, the first vertex of the initial simplex is selected as the best solution of the previous phase. In addition, the initial step size υ for the current phase is decreased geometrically using the formula $\upsilon_\tau = \frac{1}{2}\upsilon_{(\tau-1)}$. The shrink coefficient is also increased linearly as $\delta_\tau = \delta_{(\tau-1)} + 0.2$ and the other vertices of the initial simplex are constructed with initial vertex generation methods mentioned above regarding the restart number of MS-RSS. Finally, the phase counter τ is incremented by one.

For the termination of a single RSS run, τ is taken into consideration. If $\tau > 3$, the final estimate $\mathbf{S}^*$ of the global optimum is computed as the best ending values for the three phases. MS-RSS is terminated when the start number reaches the predetermined multi-start number. The best objective value is obtained among the best ending values for the three phases of all RSS.

4.3. An Exhaustive Search for Full Interdiction Problem

Up to here, we focus on the partial interdiction. Research on facility interdiction models in the literature concentrate on full facility interdiction. Since this thesis is the first attempt for partial interdiction concept, the question about the benefit of the partial interdiction model upon full interdiction models may arise. Hence, we develop an exhaustive search and solution validation method called ES so as to assess the value of the partial interdiction from the perspective of the attacker. Performing ES upon the same BPIP instances, we can evaluate the value of partial interdiction.

Exhaustive search is a trivial but very general problem-solving technique that consists of systematically enumerating all possible candidate solutions and checking whether each candidate satisfies the constraints of the problem. This search is simple to implement, and will always find a solution if it exists. However, its cost is proportional to the number of candidate solutions, which, in many practical problems, tends to grow very quickly as the size of the problem increases. It investigates in all site combinations to interdict full capacities of p facilities among m existing ones. For each combination, all budget-feasible subsets of interdiction scenarios are checked. Then, our BP problem is reduced into single level integer optimization problem which can be solved to optimality with commercial solvers.

Consider a vector $\mathbf{S}$ with m elements. Suppose that all elements in this vector is set to zero at the initialization. Then, we have to check $\binom{m}{p}$ combinations for $p =$ 1,2,...,m where p denotes the number of the facilities to be totally interdicted, namely, the number of 1's in $\mathbf{S}$ vector. The budget constraint of the attacker's problem is checked for each combination. If the budget allows, the objective value of the defender's problem is obtained by solving the LLP with CPLEX. Reviewing all possible combinations, the optimal solution is selected as the interdiction scenarios with the maximum objective function value.

The number of function evaluations required to reach a result grows exponentially in the number of the facilities m, and this strictly limits its applicability on large-sized instances. We observed that ES works in acceptable time limits when $m \leq 15$. However, it needs enormous CPU times when m is large. For example, it spends almost 20 hours to solve a high budget instance with $m = 15$. The computation time exceeds 40 hours when m is raised to 16.

5. COMPUTATIONAL RESULTS

5.1. Random Problem Generation

The performance of the proposed solution methodologies are tested on 84 BPIP instances which differ in customer and facility configurations along with parameters m and e_{tot}. These instances are produced by using the data generation scheme of (Aksen et al., 2010). The number of customers is set to $10m$ where m indicates the number of the facilities in the existing system in each instance. Customers are uniformly distributed over a circular area centered at the origin (0,0) as in (Aksen et al., 2010), but with the radius R equal to 500 units. Apart from a side length of a square L which is set to 1000 units in our instances, candidate facility sites are dispersed uniformly on (m+1) equidistant horizontal and vertical lines centered at the origin (0,0) hypothetically dice a square. The corresponding vertical and horizontal coordinates on the plane are calculated through the functions given in Table 5.1. All coordinates are then rounded to the nearest integer. The visualization of customers and facility sites on the xy-plane for $m = 5$ and $m = 15$ are shown in Appendix A.

Our test bed consists of two types of problem sets. As the literature indicates that simplex-search type procedures tend to perform well for $m \leq 10$ (Nelder and Mead, 1965; Barton and Ivey, 1996), we took $m \leq 10$ as the "small-sized" test problems and $11 \leq m \leq 17$ as the "large-sized" test problems. We use two budget levels in both problem sizes. A low interdiction budget is equal to the 30 percent of the total cost required for completely interdicting the facilities while a high interdiction budget is 60 percent of the same cost. Three random BPIP instances each with a low and a high interdiction budget is generated for each of the m values. Thus, we obtain 42 BPIP instances for the small-sized test problems with $4 \leq m \leq 10$, and 42 BPIP instances for the large-sized test problems with $11 \leq m \leq 17$.

The random problem generation template employed in the computational study is described in Table 5.1. where U(0,1) stands for a uniform random number between

0 and 1, $DU[lb,ub]$ typifies a random integer number between a lower bound lb and an upper bound ub. $(c_{xi},\ c_{yi})$ and $(f_{xj},\ f_{yj})$ designate the coordinates of customer i and facility j, respectively, and distance $d(i,\ j)$ returns the Euclidean distance between these two. Customer locations and the corresponding demand values are determined independent of each other. The customer demand in both budget level types is determined randomly from the set {5, 10, 15,...,100}. Penalty cost induced from outsourcing one unit customer demand is set to 100. Unit shipment costs are the same for each customer and fixed to 0.1 TL in all instances.

It is noteworthy that we generate some additional variations of these 84 problem instances by altering some of the parameter such as attacker's budget e_{tot}, capacities (q_j's) and the number of facilities (m) in the following sections.

Table 5.1. Random problem generation table

Parameters	Values
Number of facilities (m)	4,5,...,17
Number of customers (n)	40,50,...,170
Demand of customer i (a_i)	5,10,...,100
Unit shipment cost (c_i)	0.1
Outsource cost (c_p)	100
Capacity of facility j (q_j)	400,420,...,800
Full interdiction cost of facility j (e_j)	15000,16000,...,30000
Attacker's budget (e_{tot})	$\eta \times \sum_{j \in J} e_j$
η	0.3,0.6

5.2. Computational Environment

The source codes of both the MS-RSS and PGS algorithms are written in C# language and compiled through Microsoft Visual Studio 2005. CPLEX Callable Library 11.0 is called via C# environment for the solution of the defender's LLP. CPU times have been measured on a workstation equipped with two Intel Xeon X5460 3.16 GHz Quad-Core processors and 16 GB RAM. One core of the processor in the workstation is reserved for each test.

We present our experiments in the following order. First, we provide the overall results of 84 problem instances to the MS-RSS algorithm and that of 42 problem instances to the PGS algorithms. Then, we perform some experiments on the computational time and the attacker's objective value (i.e., the operational cost to the system planner in the existence of the malicious attacks). Finally, we give some computational and managerial insights into the BPIP problem by analyzing it from the different aspects of the overall results of experiments.

5.3. Results of the BPIP Problem Instances

In this section, we test the solution algorithms on our instances. The PGS algorithm can solve only the small-sized test problems. Since the number of combinations grows exponentially, the size of the instances with $m > 10$ are too large to be solved by PGS. Therefore, we apply the MS-RSS algorithm on 84 instances and the PGS algorithm on 42 instances, which are generated via the settings in Table 5.1.

The objective values and the CPU times obtained by MS-RSS and PGS for each small-sized problem instance are provided in Table 5.2 and Table 5.4. In these tables, "OV Gap (%)" refers to the gap between the objective values attained by MS-RSS and PGS, which is computed by the formula $100 \times (\mathrm{OV}_{\mathrm{MS-RSS}} - \mathrm{OV}_{\mathrm{PGS}})/\mathrm{OV}_{\mathrm{PGS}}$. Also, "CPU Gap (%)" denotes the computational time gap between MS-RSS and PGS, which is calculated as $100 \times (\mathrm{CPU}_{\mathrm{MS-RSS}} - \mathrm{CPU}_{\mathrm{PGS}})/\mathrm{CPU}_{\mathrm{PGS}}$. The detailed MS-RSS results to the large-sized instances can be found in Appendix B.

The average results to the small-sized problem instances with high and with low budget levels can be found in Table 5.3 and 5.5. An instance group represents three random instances of the given the number of facility sites (m). The average of objective values and CPU times are presented for each instance group as well as the average gap between the solution of MS-RSS and the solution of PGS.

Table 5.2. The results of MS-RSS and PGS to small-sized problem instances based on high level budget with $K = 6$

Instance No	e_{tot} (TL)	OV_{MS-RSS} (TL)	CPU_{MS-RSS} (sec)	OV_{PGS} (TL)	CPU_{PGS} (sec)	OV Gap (%)	CPU Gap (%)
4-1	46,800	168,174	29	167,380	4	0.47	625.00
4-2	57,600	140,902	29	139,637	3	0.91	866.67
4-3	43,800	146,357	33	144,944	5	0.97	560.00
5-1	73,800	213,249	67	211,799	12	0.68	458.33
5-2	69,000	167,760	48	166,584	14	0.71	242.86
5-3	72,600	183,783	91	183,775	16	0.00	468.75
6-1	87,000	201,362	90	201,060	67	0.15	34.33
6-2	81,600	202,636	131	200,818	93	0.91	40.86
6-3	77,400	205,554	105	203,722	125	0.90	-16.00
7-1	90,600	251,692	164	253,888	480	-0.86	-65.83
7-2	94,200	248,815	191	247,519	604	0.52	-68.38
7-3	96,600	248,187	426	246,119	574	0.84	-25.78
8-1	97,800	286,688	235	287,732	3,953	-0.36	-94.06
8-2	109,800	276,822	352	277,084	2,923	-0.09	-87.96
8-3	106,200	284,171	337	284,966	5,023	-0.28	-93.29
9-1	97,800	333,990	400	334,386	27,415	-0.12	-98.54
9-2	112,800	296,831	540	304,114	33,085	-2.39	-98.37
9-3	110,400	325,006	548	326,818	28,254	-0.55	-98.06
10-1	112,800	360,567	1,105	360,092	221,819	0.13	-99.50
10-2	119,400	348,330	859	349,406	221,395	-0.31	-99.61
10-3	137,400	357,653	470	358,521	227,275	-0.24	-99.79

The MS-RSS algorithm is a simplex-based heuristic algorithm and devised for tackling BPIP for the first time. Since there is no proposed solution algorithm to BPIP in the literature, we can only compare our solution methodology with PGS, which is an exact solution algorithm for given decimal numbers that are corresponding to the interdiction levels (i.e., the S_j values). Considering 42 instances, we conclude that the CPU times of MS-RSS are visibly less than that of PGS. When $m = 9$, MS-

Table 5.3. The average results of MS-RSS and PGS to small-sized problem instances based on high level budget with $K = 6$

Instance Group	$\mathrm{OV}_{\mathrm{MS-RSS}}$ (TL)	$\mathrm{CPU}_{\mathrm{MS-RSS}}$ (sec)	$\mathrm{OV}_{\mathrm{PGS}}$ (TL)	$\mathrm{CPU}_{\mathrm{PGS}}$ (sec)	OV Gap (%)	CPU Gap (%)
4	151,811	30	150,654	4	86.90	0.78
5	188,264	69	187,386	14	78.45	0.46
6	203,184	109	201,867	95	11.84	0.65
7	249,565	260	249,175	553	-147.89	0.16
8	282,560	308	283,261	3,966	-1234.34	-0.25
9	318,609	496	321,773	29,585	-5945.48	-1.04
10	355,517	811	356,006	221,810	-2125.09	-0.14

RSS solves BPIP 8 times faster than PGS. At the time m is increased to 10, this value increases to 50. In fifteen out of 42 instances, the objective values of PGS surpass those of MS-RSS. The maximum gap is recorded as 0.97%. The average increase in the objective values obtained from MS-RSS is 0.44%. The experiment indicates that MS-RSS is an efficient algorithm that provides good near-optimal solutions for BPIP instance within a reasonable amount of time. PGS, however, requires a large amount of time to solve the instances with high m value.

5.3.1. Contribution of Phase Approach in MS-RSS

Our MS-RSS algorithm is actually based on the Revised Simplex Search (RSS) of Humphrey and Wilson which consists of three-phase application of the NMSS method. RSS is constructed upon the fact that the most effective solution is to step away from the current termination point, restart the search procedure with a new larger simplex, cache the value once an optimum is found, and compare these resulting alternative termination points to guard against premature convergence. Much research in the literature proves that it is beneficial to restart the whole method (in the second phase) from the solution obtained in the first phase.

To emphasize the improvement achieved by applying three phases concept in the

Table 5.4. The results of MS-RSS and PGS to small-sized problem instances based on low level budget with $K = 6$

Instance No	e_{tot} (TL)	OV_{MS-RSS} (TL)	CPU_{MS-RSS} (sec)	OV_{PGS} (TL)	CPU_{PGS} (sec)	OV Gap (%)	CPU Gap (%)
4-1	23,400	133,817	69	130,912	3	2.22	2200.00
4-2	28,800	95,897	68	93,591	4	2.46	1600.00
4-3	21,900	95,760	68	92,501	4	3.52	1600.00
5-1	36,900	151,820	148	150,402	15	0.94	886.67
5-2	34,500	105,359	104	104,077	14	1.23	642.86
5-3	36,300	122,576	290	122,075	29	0.41	900.00
6-1	43,500	141,597	225	142,090	71	-0.35	216.90
6-2	40,800	123,886	413	122,557	142	1.08	190.85
6-3	38,700	132,308	306	132,185	98	0.09	212.24
7-1	45,300	173,537	721	172,133	570	0.82	26.49
7-2	47,100	172,353	641	171,586	754	0.45	-14.99
7-3	48,300	161,550	1,425	159,226	1,144	1.46	24.56
8-1	48,900	192,478	2,012	190,492	5,710	1.04	-64.76
8-2	54,900	187,248	587	184,337	2,056	1.58	-71.45
8-3	53,100	196,436	3,216	195,543	6,959	0.46	-53.79
9-1	48,900	220,231	3,104	220,723	37,287	-0.22	-91.68
9-2	56,400	191,439	5,594	189,298	18,806	1.13	-70.25
9-3	55,200	221,309	1,723	223,521	31,017	-0.99	-94.44
10-1	56,400	236,522	5,354	236,703	272,156	-0.08	-98.03
10-2	59,700	232,791	3,922	232,957	296,563	-0.07	-98.68
10-3	68,700	241,720	4,385	242,122	211,694	-0.17	-97.93

MS-RSS algorithm, we compare the performance of a three-phase runs with that of single phase runs. We examine nine instances of our data configuration with $m = 5,\ 10,\ 15$. Table 5.6 illustrates the results of this experimentation for $K = 6$ and high level budgets. $OV_{MS-RSS-1}$ refers to the objective value of MS-RSS with single phase and $OV_{MS-RSS-3}$ is that of MS-RSS with three phases. $CPU_{MS-RSS-1}$ denotes the running times value of the former and $CPU_{MS-RSS-3}$ that of the latter. “OV

Table 5.5. The average results of MS-RSS and PGS to small-sized problem instances based on low level budget with $K = 6$

Instance Group	$\mathrm{OV}_{\mathrm{MS-RSS}}$ (TL)	$\mathrm{CPU}_{\mathrm{MS-RSS}}$ (sec)	$\mathrm{OV}_{\mathrm{PGS}}$ (TL)	$\mathrm{CPU}_{\mathrm{PGS}}$ (sec)	OV Gap (%)	CPU Gap (%)
4	108,491	68	105,668	4	94.63	2.66
5	126,585	181	125,518	19	88.80	0.85
6	132,597	315	132,277	104	67.35	0.27
7	169,147	929	167,648	823	7.68	0.90
8	192,054	1,938	190,124	4,908	-183.48	1.01
9	210,993	3,474	211,181	29,037	-1012.54	-0.03
10	237,011	4,554	237,261	260,138	-7461.52	-0.07

Gap(%)" and "CPU Gap (%)" are the percentage of the gap between the objective function values and the solution times of the algorithm with single phase and our proposed algorithm with three phases, respectively. "OV Gap(%)" is obtained by the formula 100 $\times(\mathrm{OV}_{\mathrm{MS-RSS-1}} - \mathrm{OV}_{\mathrm{MS-RSS-3}})/\mathrm{OV}_{\mathrm{MS-RSS-3}}$ whereas "CPU Gap (%)" is calculated as 100 $\times$ $(\mathrm{CPU}_{\mathrm{MS-RSS-1}} - \mathrm{CPU}_{\mathrm{MS-RSS-3}})/\mathrm{CPU}_{\mathrm{MS-RSS-3}}$.

The maximum gap is found to be 2.23%. It can be seen that the ratios between the attacker's objective values obtained from single phase application and those obtained from three phases application provide concrete evidence that a three-phase approach incorporated into MS-RSS is a key improvement in the solution quality.

When the computing times are taken into consideration, the total CPU times decrease drastically in the case that single phase is applied individually to MS-RSS. In a three-phase application, the search procedure is started from the simplex generated via our initial simplex generation methods in the first phase, and from the solution obtained in the previous phase in both the second and third phases. Therefore, one single RSS run with three phases is similar to three restarts with single phase. For example, when K is 6 for MS-RSS with three phases, we actually restart the search procedure from 18 different points, which are selected as a result of the previous phase, from the solution space. It can be argued that comparing the results of a three-phase

Table 5.6. The results of MS-RSS with single phase and three phases based on high level budget with K = 6

Instance No	$\text{OV}_{\text{MS-RSS-1}}$ (TL)	$\text{CPU}_{\text{MS-RSS-1}}$ (sec)	$\text{OV}_{\text{MS-RSS-3}}$ (TL)	$\text{CPU}_{\text{MS-RSS-3}}$ (sec)	OV Gap (%)	CPU Gap (%)
5-1	208,490	20	213,249	67	-2.23	-70.15
5-2	166,777	16	167,760	48	-0.59	-66.67
5-3	181,342	26	183,783	91	-1.33	-71.43
10-1	353,490	288	360,567	1,105	-1.96	-73.94
10-2	347,231	267	348,330	859	-0.32	-68.92
10-3	357,524	139	357,653	470	-0.04	-70.43
15-1	503,668	613	513,205	1,793	-1.86	-65.81
15-2	514,052	2,525	520,933	7,733	-1.32	-67.35
15-3	511,517	1,368	515,655	3,153	-0.80	-56.61

runs of MS-RSS with K = 6 to the results of MS-RSS with single phase runs with K = 18 is much more rational means of measuring the performance of multi-phase policy. Based on this motivation, we run the following test in order to observe the effects of phase concept.

The results are summarized in Table 5.7. The column $\text{OV}_{\text{MS-RSS-1}}$ reports the objective values of MS-RSS with single phase for K = 18 while the column $\text{CPU}_{\text{MS-RSS-1}}$ lists accordingly the solution times. Similarly, the column $\text{OV}_{\text{MS-RSS-3}}$ represents the objective values of MS-RSS with three phases for K = 6 whereas the column $\text{CPU}_{\text{MS-RSS-3}}$ displays the related solution times. The last two columns of Table 5.7 are "OV Gap (%)" and "CPU Gap (%)". They refer to the gap between the values of the column $\text{OV}_{\text{MS-RSS-1}}$ and those of the column $\text{OV}_{\text{MS-RSS-3}}$, and the computational time gap between the values of column $\text{CPU}_{\text{MS-RSS-1}}$ and the column $\text{CPU}_{\text{MS-RSS-3}}$, respectively. "OV Gap (%)" is computed by the formula $100 \times (\text{OV}_{\text{MS-RSS-1}} - \text{OV}_{\text{MS-RSS-3}})/\text{OV}_{\text{MS-RSS-3}}$. Also, "CPU Gap" is calculated as $100 \times (\text{CPU}_{\text{MS-RSS-1}} - \text{CPU}_{\text{MS-RSS-3}})/\text{CPU}_{\text{MS-RSS-3}}$.

The results demonstrate that the computational times are increased considerably

Table 5.7. The results of MS-RSS with single phase for $K = 18$ and three phases for $K = 6$

Instance No	$\text{OV}_{\text{MS-RSS-1}}$ (TL)	$\text{CPU}_{\text{MS-RSS-1}}$ (sec)	$\text{OV}_{\text{MS-RSS-3}}$ (TL)	$\text{CPU}_{\text{MS-RSS-3}}$ (sec)	OV Gap (%)	CPU Gap (%)
5-1	210,885	59	213,249	67	-1.11	-11.94
5-2	167,474	53	167,760	48	-0.17	10.42
5-3	183,447	87	183,783	91	-0.18	-4.40
10-1	359,891	836	360,567	1,105	-0.19	-24.34
10-2	347,231	685	348,330	859	-0.32	-20.26
10-3	357,524	618	357,653	470	-0.04	31.49
15-1	503,668	1,821	506,205	1,793	-0.50	1.56
15-2	514,052	6,523	520,933	7,733	-1.32	-15.65
15-3	511,517	3,862	515,655	3,153	-0.80	22.49

when K is set to 18 for single phase application. On the other hand, even increasing the number of multi-start, a single phase application cannot outperform a three phase application in any instances.

We conclude here that neither single phase application with $K = 6$ nor that with $K = 18$ provides good solutions to the BPIP problem, and that the phase concept in our proposed solution methodology plays an important role in the solution quality.

5.3.2. Contribution of Plateau Counter in MS-RSS

As we mentioned before, a plateau problem occurs when the objective value of the simplex vertices have the same value. We add plateau counter ω into our MS-RSS algorithm to overcome unnecessary iterations performed until the maximum iteration number is reached.

With an effort to understand the contribution of the plateau check, we eliminate the plateau counter ω from the algorithm by assigning a very high value to the maximum number of iterations performed without an improvement in the objective value.

Table 5.8. The effect of plateau check in MS-RSS

Instance No	$\mathrm{OV}_{\mathrm{without}-\omega}$ (TL)	$\mathrm{CPU}_{\mathrm{without}-\omega}$ (sec)	$\mathrm{OV}_{\mathrm{with}-\omega}$ (TL)	$\mathrm{CPU}_{\mathrm{with}-\omega}$ (sec)
5-1	211,082	100	213,249	67
5-2	167,686	62	167,760	48
5-3	183,783	126	183,783	91
10-1	356,050	1,211	360,567	1,105
10-2	348,330	864	348,330	859
10-3	357,653	478	357,653	470
15-1	501,704	1,827	513,205	1,793
15-2	520,933	7,681	520,933	7,733
15-3	515,655	3,177	515,655	3,153

Using these settings, we take three problem instances for each of $m = 5$, 10, 15. K is set to 6 and the high level budget is chosen.

The results of this analysis for each problem instances are demonstrated in Table 5.8, where $\mathrm{OV}_{\mathrm{without}-\omega}$ is the objective value of MS-RSS without plateau counter, and $\mathrm{OV}_{\mathrm{with}-\omega}$ that with plateau counter. $\mathrm{CPU}_{\mathrm{without}-\omega}$ and $\mathrm{CPU}_{\mathrm{with}-\omega}$ represent the solution time of the former and the latter, respectively.

In all but one of the problems, MS-RSS without plateau check spends more time than that with plateau check and results with no improvement in the objective function value of the instances. This behavior shows that when the objective values of the vertices are the same, MS-RSS without plateau check fail to stop the current phase and/or a single RSS run until the maximum number of iteration is satisfied. Since the algorithm is trapped in the plateau, it cannot traverse the solution space, which has an adverse effect on the solution quality.

As can be seen in Table 5.8, only the CPU time of instance 15-2 to MS-RSS without check is less than that to our MS-RSS. It is obvious that if the plateau check is omitted, the search procedure will continue iterate with reflection steps without an improvement in the objective value but with the change in the vertices of the simplex. Reaching the maximum iteration number, the vertex obtained from the last reflection step will be selected as a starting vertex of the initial simplex in the new phase. In other words, a single RSS run of MS-RSS without plateau check will start from a different point where the other algorithm starts, and there is a possibility of finding different objective values. By chance, instance 15-2 is terminated with a higher value when the check is not appointed. If the maximum number of the iterations of the phases are changed, this value has a possibility of decreasing adversely.

Considering the facts mentioned above, incorporating plateau check into the MS-RSS algorithm is a significant factor that especially gains importance in the large-sized problems as each iteration requires much more computation time compared to the small-sized ones.

5.3.3. Contribution of Cycling Check in MS-RSS

Recall that after a sequence of reflection operations, MS-RSS may fail to replace the worst vertex in the current simplex, which brings the search back to the simplex from which this sequence started. To prevent this phenomenon, we adjust our procedure to reflect the second worst point rather than the point reflected in the previous iteration.

To highlight the impacts of the cycling check upon MS-RSS performance, we allow the algorithm to reflect the worst point. Then, we analyze the degree of the decrease in the computational times along with the increase in the objective function values via comparing the solutions of the instances with $m = 5$, 10, 15 for $K = 6$ and high level budget. In all instances, the results, which are summarized in Appendix C, verify that when the cycling check is not used, the CPU times are increasing 62.08% on the average whereas the objective values are worsening 1.15% on the average. Therefore, it can be concluded that cycling check used in our propose algorithm has a key role on the algorithm performance.

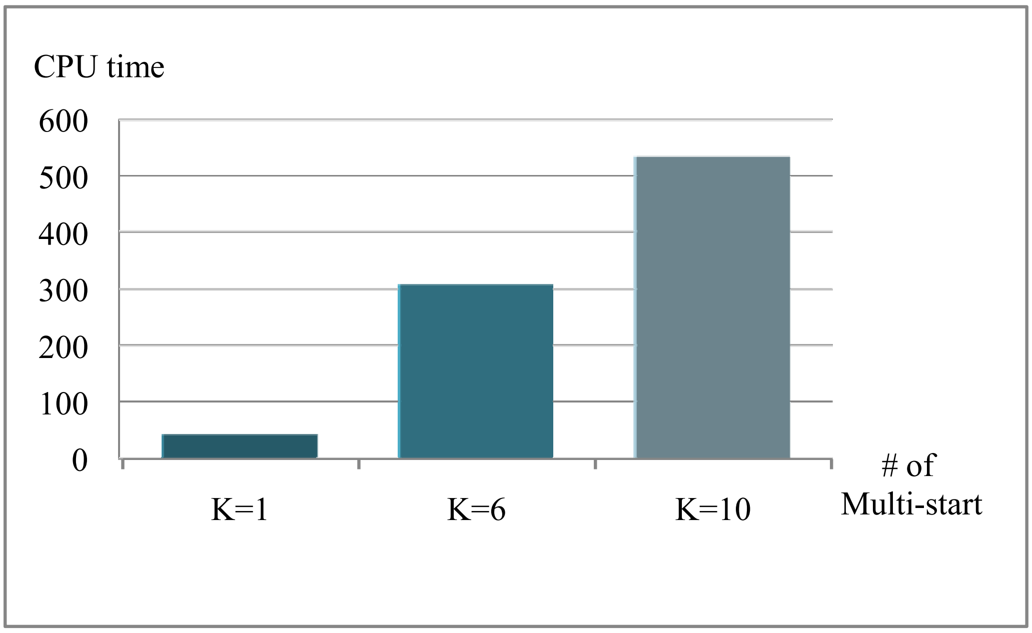

Figure 5.1. The average CPU times to small-sized problems for different number of multi-starts

5.3.4. The Effect of the Number of Multi-starts on MS-RSS

To overcome premature convergence to a non-maximum solution in BPIP, an efficient method is to restart the search procedure as many time as the predetermined number of the multi-start K. This provide us to explore different parts of the solution space. In each restart, MS-RSS is initialized from dissimilar points which are generated via our deterministic and double randomization vertex generation methods.

It is obvious that starting the search from dissimilar points in the search space increases the solution quality but causing the growth in the running time of the algortihm. Therefore, the determination of parameter K is of utmost important from the view of the efficiency of the MS-RSS algorithm. To overcome this challenge, we conduct the following test. We take one instance with high level budget for each number of facilities in our test bed, in other words, m varies from 4 to 17. Then, we restart the MS-RSS algorithm one ($K = 1$), six ($K = 6$), and ten ($K = 10$) times.

Clearly, the total time spend to finish the algorithm is increases as the number of starts increases. The detailed results of this analysis for each problem instance are

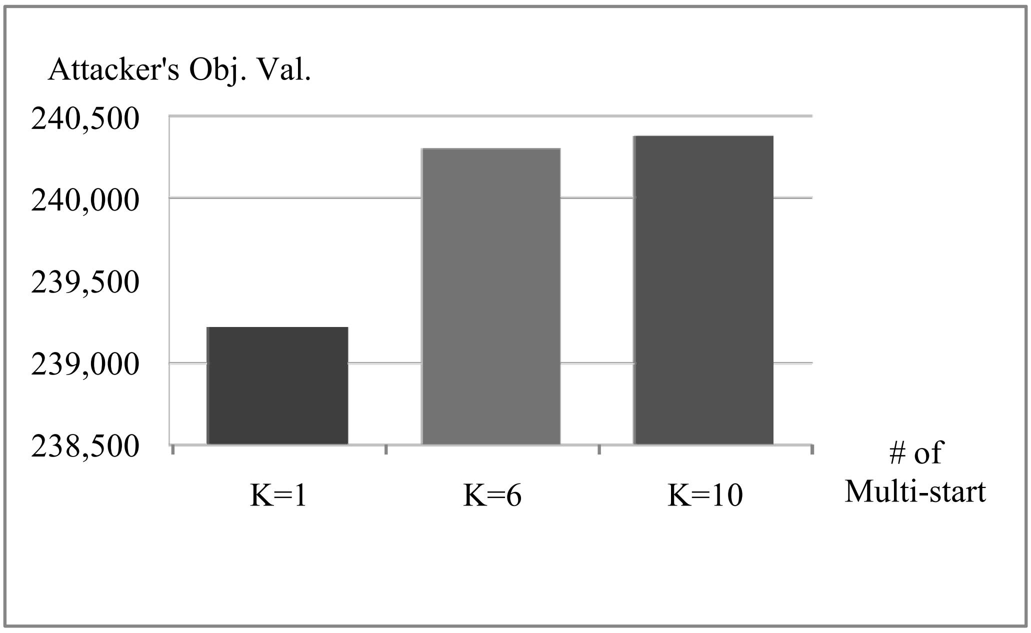

Figure 5.2. The average attacker's objective value to small-sized problems for different number of multi-starts

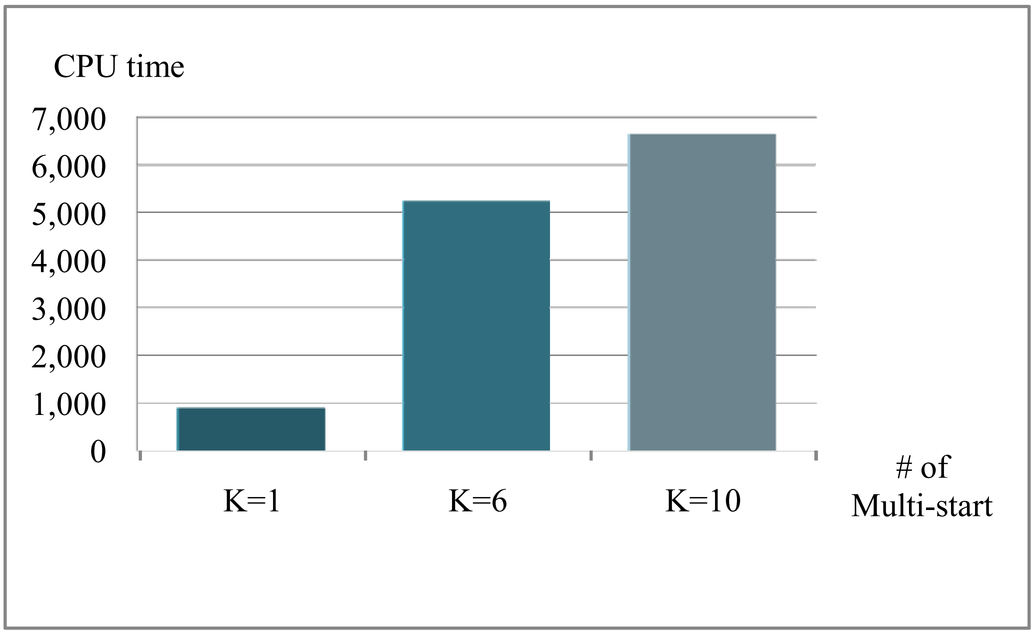

Figure 5.3. The average CPU times to large-sized problems for different number of multi-starts

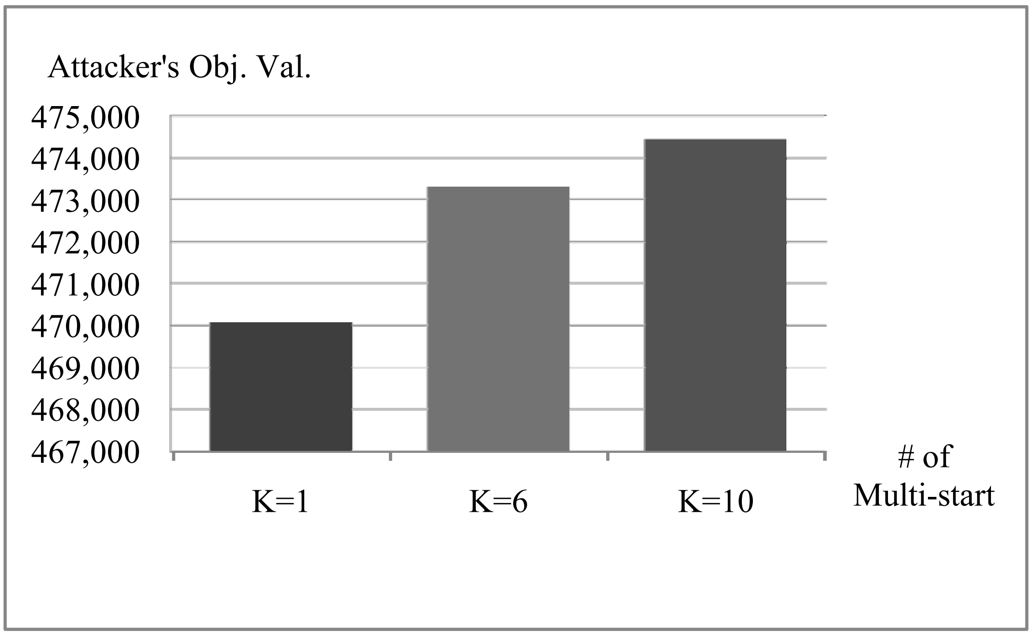

Figure 5.4. The average attacker's objective value to large-sized problems for different number of multi-starts

provided in Appendix D. The summarized illustrations for the average results to 7 small-sized problem instances are given in Figure 5.1 and 5.2 while those to 7 large-sized problem instances are provided in Figure 5.3, and 5.4, respectively. In all figures, the horizontal axis displays the K values. The vertical axis in Figure 5.1 and 5.3 show the average CPU times while that in Figure 5.2 and 5.4 demonstrate the attacker's objective values.

An interesting finding is that the increase in the computational times are not proportional to the number of starts. When K is increased from 1 to 6, the CPU times of the latter are higher about 7.2 times for small-sized problems and 5.8 times for large ones than that of the former. These increases are about 1.73 and 1.26 times after K is raised from 6 to 10. This circumtance is due to the termination criterion of MS-RSS. The time spend to complete a single RSS run cannot be known a priori as it heavily relies on the iteration numbers performed until optimum solution is found. For example, MS-RSS may converge to an optimum solution in 25 iterations or in 300 iterations. To reach a solution, RSS may iterate many times until it reaches maximum iteration number or the simplex has to be small enough.

Considering the increases in the objective values, if K is increased from 1 to 6, the increase is 0.64% on average for all instances. On the other hand, when it is set from 6 to 10, the same objective values are obtained in the nine out of the fourteen instances. Therefore, we end up with a conclusion that it would be adequate to set K to 6 in order to obtain reasonable solution quality.

5.3.5. Partial Interdiction Problems versus Full Interdiction Problems

Recall that our problem permits the attacker to interdict facilities partially. As BPIP was not addressed before in the literature, we devise an exhaustive search method with the aim of evaluating the value of the partial interdiction models. Our exhaustive search algorithm (ES) inspects, indeed, all combinations of the full interdiction of facilities as long as the attacker's budget allows. In this search, if a facility is attacked, it will become totally inoperative so that it could not provide service to its customers.

To highlight the utility of the partial interdiction concept, we perform an experiment so as to compare the results of our two solution algorithms (PGS and MS-RSS) with the ES algorithm. Our test bed consists of 72 BPIP instances in which three problem instances are taken with m = 4, 5,...,15 for two different budget levels. The detailed computational times and the objective values of partial and full interdiction problems are given in Appendix E. The summary of the this experiment for high level budget is presented in Table 5.9 and Table 5.10. In these tables, "OV Gap " denotes the average gap between the objective values of partial interdiction obtained via MS-RSS or PGS and that of full interdiction calculated via ES for each instance group that has the same facility number in the test bed. The improvement by partial interdiction is computed by the formula provided in Section 5.3 for the values obtained from the MS-RSS, PGS and ES algorithms. Additionally, the average of the running times are mentioned for each instance group.

Due to the exponential time increase in the PGS algorithm, the problem instances whose facility number are greater than 10 cannot be solved by PGS and therefore we compare the MS-RSS solutions of large-sized problem instances with their ES solutions.

Table 5.9. Partial interdiction vs. full interdiction (high level budget)

	Partial Interdiction MS-RSS or PGS		Full Interdiction ES		
Instance Group	OV (TL)	CPU (sec)	OV (TL)	CPU (sec)	OV Gap (%)
4	151,811	30	141,101	1	7.71
5	188,264	69	180,797	2	4.08
6	203,184	109	197,205	6	3.10
7	249,565	260	242,503	28	2.93
8	283,261	3,966	274,145	59	3.33
9	321,773	29,585	310,508	219	3.62
10	356,165	149,925	349,477	693	1.92
11	378,681	1,127	375,288	3,682	0.91
12	403,795	1,193	401,755	4,541	0.50
13	449,585	4,110	447,871	46,740	0.40
14	476,630	6,543	475,423	39,623	0.25
15	514,264	4,226	522,947	88,294	-1.66

Comparing the results of partial interdiction with those of full interdiction, it is observed that the CPU times of the latter is smaller than those of the former in small-sized instances while the reverse is true for the large ones because of the exponential increase in the number of combinations to be evaluated. For instance, at the time the number of the existing facilities is 15, the average running times of ES is 21 times higher than that of MS-RSS. On contrary, the benefit of the partial interdiction lessens gradually as the number of facilities enlarges. The maximum gap between the partial and the full interdiction problems is recorded as 15.41%. The results displays that our two solution algorithms have lower objective value only in eight out of 72 instances. In fact, in five out of these seven instances, m is equal to 15. This explains the situation that when the dimension of the simplex, that is to say, the parameter m, is high, the simplex-search based procedures tend to perform poorly. The literature signifies that they can perform well when the dimension is less than or equal to 10 (Nelder and Mead, 1965; Barton and Ivey, 1996; Humphrey and Wilson 2000). The high level

Table 5.10. Partial interdiction vs. full interdiction (low level budget)

	Partial Interdiction MS-RSS or PGS		Full Interdiction ES		
Instance Group	OV (TL)	CPU (sec)	OV (TL)	CPU (sec)	OV Gap (%)
4	108,491	68	101,007	1	7.52
5	126,585	181	114,040	1	11.41
6	132,761	263	125,016	3	6.82
7	169,147	929	166,411	14	1.66
8	192,054	1,938	180,216	24	6.67
9	211,894	24,633	206,130	101	2.88
10	237,261	260,058	230,223	329	3.06
11	244,048	9,678	240,314	2,107	1.57
12	254,246	6,677	250,849	1,964	1.30
13	295,190	25,745	291,039	31,991	1.46
14	294,241	18,566	293,250	14,269	0.34
15	344,333	28,121	345,653	51,335	-0.38

dimensionality is defined as 18 for their unconstrained test problems (Humphrey and Wilson, 2000).

When the results of PGS and ES are taken into consideration, the computational time of the latter is less than the former in all instances even though both algorithms are exponential time algorithms. On the other hand, with respect to their objective values, PGS outperforms ES in all instances. The attacker's objective value calculated via PGS is higher 2.82% on average.

To summarize, both the objective values of the MS-RSS and the PGS algorithms are higher than those of the ES algorithm. This provides evidence that choosing an interdiction strategy in which the facilities are attacked at some fraction rather than policy where the facilities are rendered totally is worthwhile from the view of both our actors.

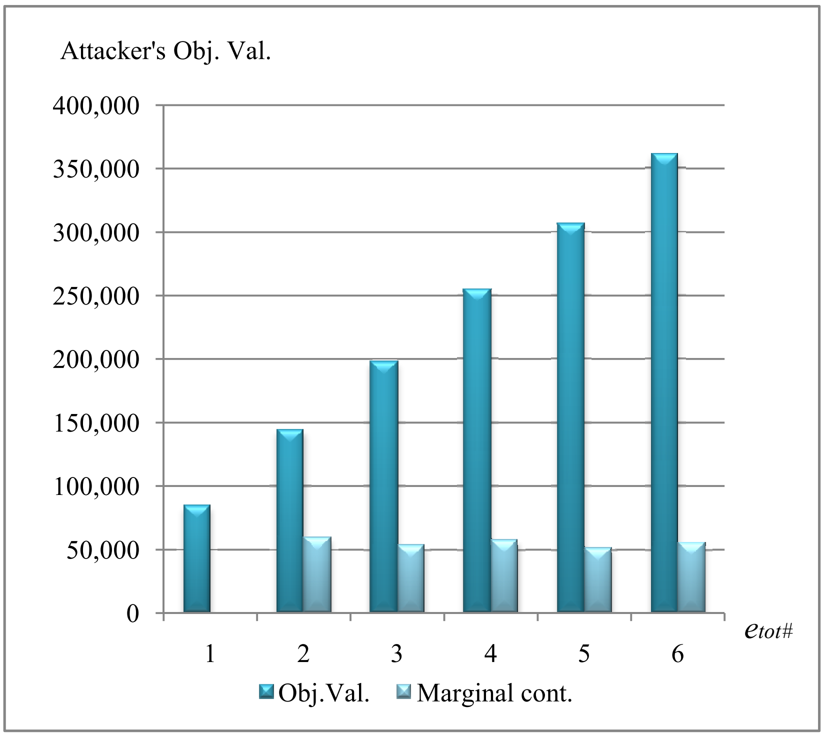

Figure 5.5. The objective values and the marginal contributions of instance 7-1

5.3.6. The Effect of the Budget in Attacker's Strategies

With the aim of understanding the value of interdiction budget from the attacker's viewpoint, we conduct analysis on the parameter e_{tot}. We obtain six different budget levels which are increased by 20% from 0 to the amount required for interdicting all facilities in the network completely. Instance 7-1 with m=7 and 14-1 with m=14 are investigated for the sake of drawing a conclusion about the behavior of the attacker.

Evidently, the final objective values increase as the budget levels increase. We call the increase in the objective value due to addition of new budget resources as the marginal contribution of these resources. The results of this analysis for different e_{tot} values are shown in Appendix F. In Figure 5.5 and Figure 5.6, the objective values for each e_{tot} are displayed for instance 7-1 and 14-1. It can be seen that the marginal contributions are close to each other in both instances. This circumstances can be enlightened such that e_{tot}'s are computed from the certain percentage of the full interdiction cost of the whole system. That is, we tolerate attacker to cause damage

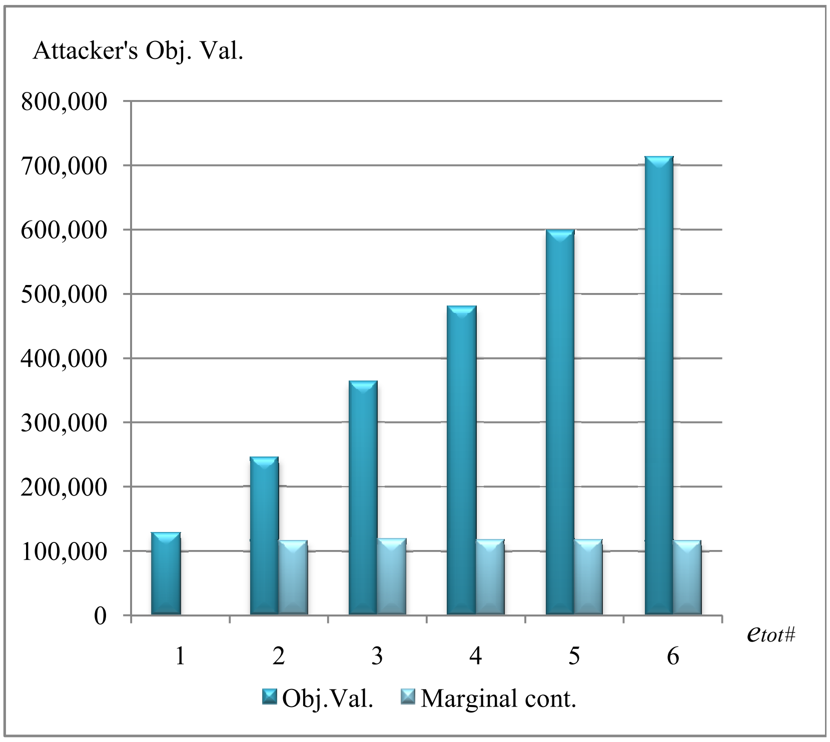

Figure 5.6. The objective values and the marginal contributions of instance 14-1

in the system with an assured level, as he would deplete his all budget in the optimal interdiction scenarios. For that reason, examining carefully how the S_j values alternate at the time e_{tot} is amplified can give us an evidence of the attacker's interdiction actions.

The S_j values of instance 7-1 for different budget levels are reviewed in Table 5.11. The visualization of instance 7-1 in which customer nodes and the facility sites are plotted in a two-dimensional Cartesian space can be seen in Figure 5.7.

When the budget is increased from 0 to 30,200, the interdictor prefers to interdict facility 6 with the highest fraction of his budget as this facility is the nearest one to the customer crowded areas, furthermore, depletes the remaining budget on other facilities. After e_{tot} is enlarged to 60,400; most disruptive fractions are due to facility 6 and facility 5. As can be seen in Figure 5.7, facility 5 is the nearest facility to facility 6 and the second closest one to the centralization of the customers. The attacker increases the disruption levels of facility 3 and facility 2 as the budget increases to 90,600. The results indicates that the S_j values of facility 7 are the smallest ones in each budget

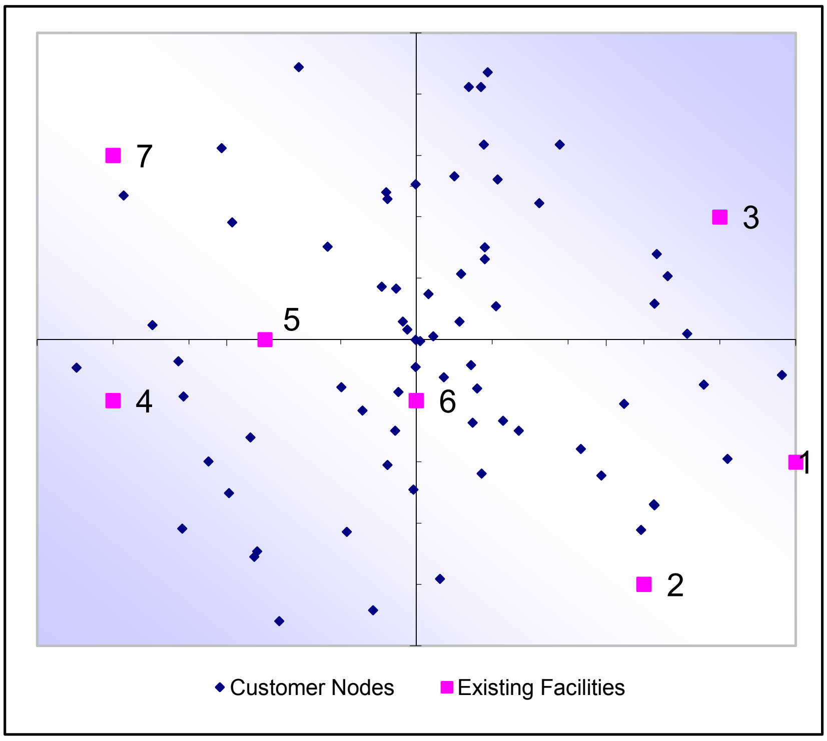

Figure 5.7. The visualization of the instance 7-1

Table 5.11. The facility interdiction fractions of Problem instance 7-1 for different interdiction budget

FN[1]	e_{tot1}= 0	e_{tot2}= 30,200	e_{tot3}= 60,400	e_{tot4}= 90,600	e_{tot5}= 120,800	e_{tot6}= 151,000
1	0%	2.07%	3.12%	50.97%	89.96%	100.00%
2	0%	6.04%	49.20%	95.33%	86.42%	100.00%
3	0%	3.45%	10.43%	98.03%	63.34%	100.00%
4	0%	3.23%	17.63%	2.02%	96.03%	100.00%
5	0%	42.47%	99.43%	97.33%	91.72%	100.00%
6	0%	97.32%	91.94%	93.43%	95.13%	100.00%
7	0%	0.16%	7.58%	3.39%	45.66%	100.00%

level since the average distance from customers to this facility is the highest of all facilities. It is trivial that when e_{tot} is greater than or equal to 151,000 all facilities are rendered wholly.

[1]Facility Number

At the end of these analysis, it can be concluded that the attacker follows the policy that he starts an interdiction from the facility that has the smallest average distance to the customers towards the ones that are distant from the customer concentration by reducing the degree of harmful actions as long as his budget permits.

6. CONCLUSION

In this thesis, we studied the bilevel partial interdiction problem with capacitated facilities and demand outsourcing (BPIP) for the planning of critical facilities. The problem is actually a static Stackelberg game between two actors: a system planner and an intelligent attacker. The system planer takes the role of the system defender who is responsible for servicing to customers residing at a number of demand nodes under the interdictor's attacks. On contrary, the attacker, who acts as a leader in BPIP, attempts to cause the maximum possible disruption in the service level. Therefore, the defender has to take the attacks into consideration while serving customer in order to minimize the worst-case disruption cost that can possibly be inflicted by the attacker. We relax the common assumption made in interdiction problems that a facility is rendered completely if it is attacked. That is, the facilities will persist to provide service at a lower level which will depend on the extent of the interdiction.

To our knowledge, this study is the first attempt for the partial interdiction of the facilities in the same leader-follower game. The problem is modeled as a bilevel mixed integer programming formulation, where the leader is the interdictor and the follower is the system planner. Two solution methods are proposed for BPIP. First, we develop a progressive grid search (PGS) in which the search is applied on the predetermined values, i.e., support points, within the solution space corresponding to the attacker, while the CPLEX is employed so as to optimize the follower's problem. Secondly, we devise a simplex-based search algorithm which is called as MS-RSS. Our proposed algorithm is a variant of the Revised Simplex Search (RSS) of Humphrey and Wilson (2000) which includes three phase application of the NM method. As we have encountered many obstacles that the simplex-based algorithms tend to, we alter RSS by additionally including some inner algorithms such as plateau and cycling checks, multi-start applications, and so forth. They are designed to tackle with some of the critical weaknesses of NM. In the MS-RSS algorithm, performing initial vertex generation methods, the initial point used in the construction of the initial simplex of single RSS run is found. The ULP, namely, the attacker's interdiction policy problem, is solved with the help of the search procedure. In each iteration, vertices of the simplex, which correspond to the S_j variables of the ULP, are generated via move operations of the search procedure. These values are then used as an input for the second level problem which is solved to optimality by CPLEX 11.0.

The performance of the PGS and MS-RSS procedures are tested on 42 and 84 randomly generated test instances, respectively. The small-sized instances are solved by both methods. The results of the instances point out that the MS-RSS algorithm works quite well in a wide spectrum of various problem sizes and yields high quality solutions to BPIP problem. The solution times of PGS increases exponentially with the number of the existing facilities while that of MS-RSS depends on the number of starts and the budget level of the attacker.

We perform computational experiments to find out the strengths and the sensitivities of the algorithms that affect the computational performance. Phase concept, plateau checks, cycling checks, and the number of the multi-starts are identified as important factors that play critical role on the solution performance of the MS-RSS algorithm.

We develop an exhaustive search algorithm (ES) to solve our problem instances for impartial interdictions with the aim of assessing the benefit of following a strategy in which the facility capacities are reduced at some fraction and are not diminished completely from the view of the attacker. We solve our small-sized instances with PGS, MS-RSS, and ES and our large-sized instances with MS-RSS and ES. Then, our two proposed optimization procedures are compared against ES with respect to the running times and the objective values. On the basis of the comparisons, we conclude that interdicting to facilities partially is a rational act for the attacker as it causes a higher operational cost to the system planner.

Another key finding of this study is that regardless of the number of the facilities and their capacities as well as the budget level, the attacker prefers to reduce the capacity of the facilities, as much as possible, that are close the customer crowded areas. Beginning from the facility that has the smallest average distance to customers, he proceeds with the ones that are away from the customers by reducing the extent of the attack.

In the lights of these findings, we sincerely believe that the further effort of research on the bilevel partial interdiction problems will provide additional insights about the defensive endeavors in the network design to the system planners. Based on this motivation, the promising area for the future work might be the extension of the problem studied in this thesis that considers partial protection for a better allocation of limited protection resources among the opened facilities. In other word, the protection resources are allocated by the system planner in such a way that only the critical capacity of the facilities is protected.

APPENDIX A: CUSTOMER AND FACILITY CONFIGURATIONS

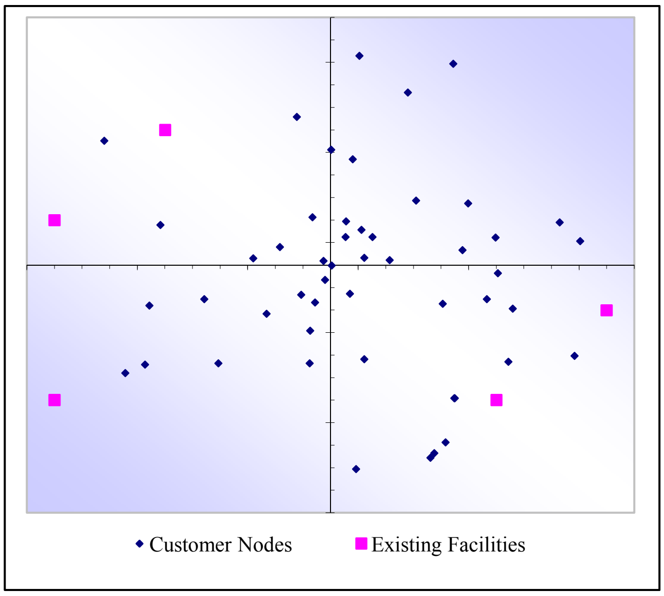

Figure A.1. Customer and facility configuration of instance 5-1

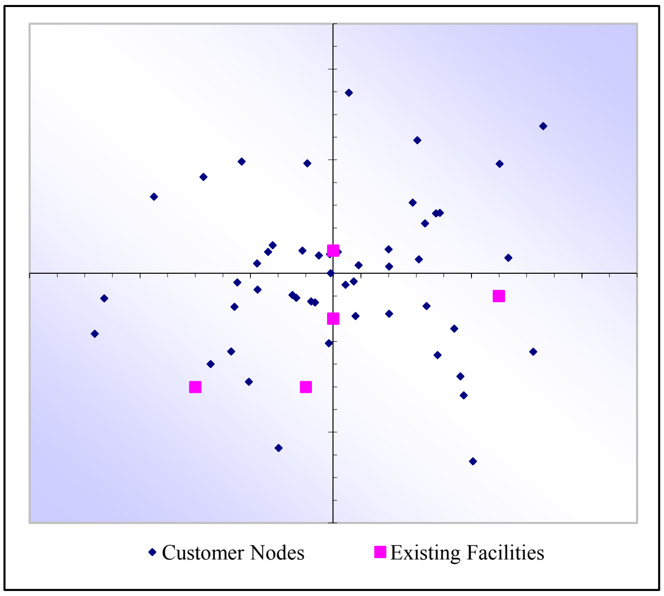

Figure A.2. Customer and facility configuration of instance 5-2

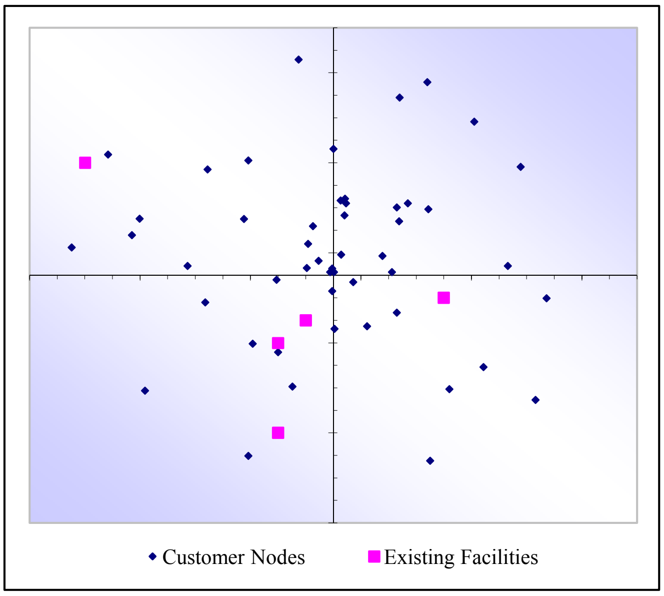

Figure A.3. Customer and facility configuration of instance 5-3

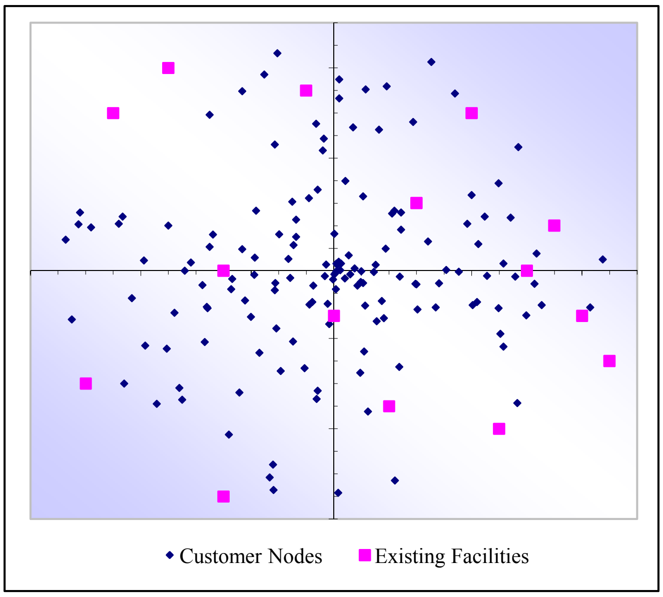

Figure A.4. Customer and facility configuration of instance 15-1

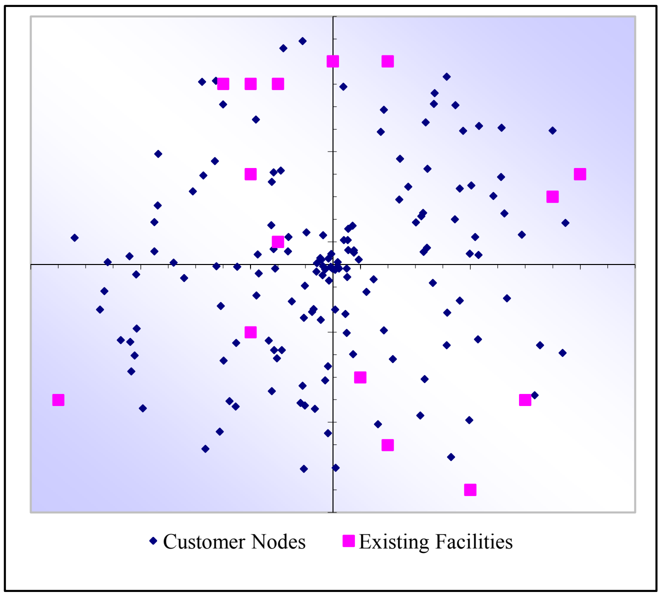

Figure A.5. Customer and facility configuration of instance 15-2

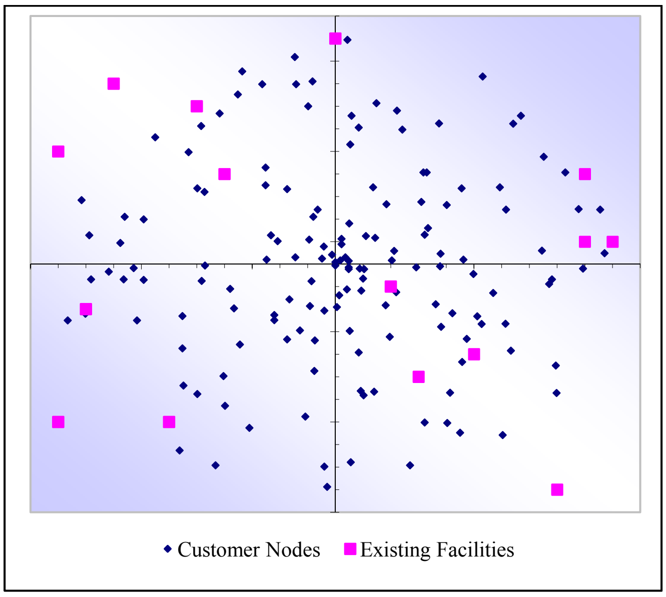

Figure A.6. Customer and facility configuration of instance 15-3

APPENDIX B: OVERALL RESULTS TO MS-RSS AND PGS

Table B.1. MS-RSS solutions to large-sized problem instances based on high level budget with $K = 6$

Instance No	e_{tot} (TL)	OV_{MS-RSS}[1] (TL)	CPU_{MS-RSS} [2] (sec)
11-1	136,800	380,498	999
11-2	139,200	375,081	1,272
11-3	150,600	380,463	1,109
12-1	159,600	403,593	1,629
12-2	151,200	418,656	890
12-3	165,000	388,935	1,061
13-1	165,000	450,275	1,535
13-2	164,400	443,106	6,918
13-3	173,400	455,373	3,877
14-1	180,600	478,885	5,214
14-2	184,800	475,530	2,709
14-3	181,800	475,474	11,705
15-1	199,200	513,205	1,793
15-2	195,000	520,933	7,733
15-3	204,000	515,655	3,153
16-1	210,000	513,711	4,085
16-2	213,000	525,476	10,625
16-3	216,000	540,552	8,321
17-1	226,800	552,209	4,821
17-2	229,800	554,290	6,513
17-3	229,800	582,223	7,610

[1]Attacker's objective value calculated via MS-RSS

[2]The solution time of MS-RSS

Table B.2. MS-RSS solutions to large-sized problem instances based on low level budget with $K = 6$

Instance No	e_{tot} (TL)	$OV_{\text{MS-RSS}}$[1] (TL)	$CPU_{\text{MS-RSS}}$ [2] (sec)
11-1	68,400	244,826	10,920
11-2	69,600	242,505	8,997
11-3	75,300	244,814	9,116
12-1	79,800	253,023	9,174
12-2	75,600	277,464	5,601
12-3	82,500	232,252	5,256
13-1	82,500	291,288	5,666
13-2	82,200	283,359	36,702
13-3	86,700	310,922	34,866
14-1	86,400	288,379	17,073
14-2	92,400	291,356	13,583
14-3	90,900	302,989	25,042
15-1	99,600	329,193	12,666
15-2	97,500	352,567	43,341
15-3	102,000	351,240	28,357
16-1	105,000	303,939	20,680
16-2	106,500	315,863	32,583
16-3	108,000	354,373	98,176
17-1	113,400	339,718	52,329
17-2	114,900	344,157	73,914
17-3	114,900	381,281	58,124

[1]Attacker's objective value calculated via MS-RSS

[2]The solution time of MS-RSS

APPENDIX C: EFFECT OF CYCLING CHECK IN MS-RSS

Table C.1. The results of MS-RSS and MS-RSS without cycling check to instances based on high level budget with $K = 6$

Instance No	$\text{OV}_{\text{no-cycling}}$[1] (TL)	$\text{CPU}_{\text{no-cycling}}$[2] (sec)	$\text{OV}_{\text{with-cycling}}$[3] (TL)	$\text{CPU}_{\text{with-cycling}}$[4] (sec)	OV Gap (%)	CPU Gap (%)
5-1	208,219	109	213,249	67	2.36	-62.69
5-2	166,765	91	167,760	48	0.59	-89.58
5-3	180,521	101	183,783	91	1.77	-10.99
10-1	351,133	1,896	360,567	1,105	2.62	-71.58
10-2	347,724	1,379	348,330	859	0.17	-60.54
10-3	355,242	782	357,653	470	0.67	-66.38
15-1	503,668	2,118	513,205	1,793	1.18	-18.13
15-2	515,576	9,768	520,933	7,733	1.03	-26.32
15-3	512,642	7,961	515,655	3,153	0.58	-152.49

[1] Attacker's objective value calculated via MS-RSS without cycling check

[2] The solution time of MS-RSS without cycling check

[3] Attacker's objective value calculated via MS-RSS

[4] The solution time of MS-RSS

APPENDIX D: EFFECT OF THE MULTI-START NUMBER IN MS-RSS

Table D.1. MS-RSS solutions problem instances based on high level budget with $K = 1$ and $K = 6$

Instance No	$\mathrm{OV}_{\mathrm{K=1}}$[1] (TL)	$\mathrm{CPU}_{\mathrm{K=1}}$[2] (sec)	$\mathrm{OV}_{\mathrm{K=6}}$[3] (TL)	$\mathrm{CPU}_{\mathrm{K=6}}$[4] (sec)	OV Gap (%)	CPU Gap (%)
4-2	138,445	7	140,902	29	1.74	75.86
5-2	164,511	7	167,760	48	1.94	85.42
6-2	202,609	18	202,636	131	0.01	86.26
7-2	248,512	24	248,815	191	0.12	87.43
8-2	276,822	45	276,822	352	0.00	87.22
9-2	295,546	40	296,831	540	0.43	92.59
10-2	348,076	157	348,330	859	0.07	81.72
11-2	373,776	194	375,081	1,272	0.35	84.75
12-2	415,226	128	418,656	890	0.82	85.62
13-2	442,249	1,259	443,106	6,918	0.19	81.80
14-2	473,180	749	475,530	2,709	0.49	72.35
15-2	516,896	1,432	520,933	7,733	0.77	81.48
16-2	520,560	1,654	525,476	10,625	0.94	84.43
17-2	548,529	876	554,290	6,513	1.04	86.55

[1]Attacker's objective value calculated via MS-RSS for single restart

[2]The solution time of MS-RSS with single restart

[3]Attacker's objective value calculated via MS-RSS for $K = 6$

[4]The solution time of MS-RSS with $K = 6$

Table D.2. MS-RSS solutions to problem instances based on high level budget with $K = 6$ and $K = 10$

Instance No	$OV_{K=6}$[1] (TL)	$CPU_{K=6}$[2] (sec)	$OV_{K=10}$[3] (TL)	$CPU_{K=10}$[4] (sec)	OV Gap (%)	CPU Gap (%)
4-2	140,902	29	141,240	47	0.24	62.07
5-2	167,760	48	167,953	82	0.11	70.83
6-2	202,636	131	202,636	214	0.00	63.36
7-2	248,815	191	248,815	412	0.00	115.71
8-2	276,822	352	276,822	537	0.00	52.56
9-2	296,831	540	296,831	1,110	0.00	105.56
10-2	348,330	859	348,330	1,337	0.00	55.65
11-2	375,081	1,272	378,769	2,098	0.97	64.94
12-2	418,656	890	418,656	1,459	0.00	63.93
13-2	443,106	6,918	443,106	8,784	0.00	26.97
14-2	475,530	2,709	475,530	3,698	0.00	36.51
15-2	520,933	7,733	526,452	10,799	1.05	39.65
16-2	525,476	10,625	525,476	11,713	0.00	10.24
17-2	554,290	6,513	560,009	7,846	1.02	20.47

[1]Attacker's objective value calculated via MS-RSS for $K = 6$

[2]The solution time of MS-RSS with $K = 6$

[3]Attacker's objective value calculated via MS-RSS for $K = 10$

[4]The solution time of MS-RSS with $K = 10$

APPENDIX E: PARTIAL INTERDICTION vs. FULL INTERDICTION

Table E.1. Partial interdiction versus full interdiction based on high level budget with $K = 6$

		Partial Interdiction MS-RSS or PGS		Full Interdiction ES			
Instance	#EF[1]	OV[2]	CPU[3]	OV[4]	CPU [5]	OV Gap	CPU Gap
No		(TL)	(sec)	(TL)	(sec)	(%)	(%)
4-1	4	168,174	29	154,864	0	8.59	0
4-2	4	140,902	29	126,343	1	11.52	2800.00
4-3	4	146,357	33	142,095	1	3.00	3200.00
5-1	5	213,249	67	200,510	2	6.35	3250.00
5-2	5	167,760	48	159,977	2	4.87	2300.00
5-3	5	183,783	91	181,904	2	1.03	4450.00
6-1	6	201,362	90	198,816	4	1.28	2150.00
6-2	6	202,636	131	189,731	8	6.80	1537.50
6-3	6	205,554	105	203,067	7	1.22	1400.00
7-1	7	251,692	164	246,842	16	1.96	925.00
7-2	7	248,815	191	243,036	19	2.38	905.26
7-3	7	248,187	426	237,630	50	4.44	752.00
8-1	8	287,732[6]	3,953	273,713	55	5.12	7087.27
8-2	8	277,084[6]	2,923	270,847	53	2.30	5415.09
8-3	8	284,966[6]	5,023	277,874	69	2.55	7179.71
9-1	9	334,386[6]	27,415	310,419	211	7.72	12892.89
9-2	9	304,114[6]	33,085	300,805	222	1.10	14803.15

[1]The number of the facilities in the system
[2]Attacker's partial interdiction objective value calculated via MS-RSS or PGS
[3]The solution time of MS-RSS or PGS with the related m value
[4]Attacker's full interdiction objective value calculated via ES
[5]The solution time of ES with the related m value
[6]The solution obtained from PGS

Table E.1. Partial interdiction versus full interdiction based on high level budget with $K = 6$ (Contd.)

		Partial Interdiction		Full Interdiction			
		MS-RSS or PGS		ES			
Instance	#EF[1]	OV[2]	CPU[3]	OV[4]	CPU [5]	OV Gap	CPU Gap
No		(TL)	(sec)	(TL)	(sec)	(%)	(%)
9-3	9	326,818[6]	28,254	320,299	224	2.04	12513.39
10-1	10	360,567	1,105	356,161	665	1.24	66.17
10-2	10	349,406[6]	221,395	343,840	885	1.62	24916.38
10-3	10	358,521[6]	227,275	348,431	529	2.90	42863.14
11-1	11	380,498	999	378,304	2,136	0.58	-53.23
11-2	11	375,081	1,272	374,132	4,678	0.25	-72.81
11-3	11	380,463	1,109	373,427	4,231	1.88	-73.79
12-1	12	403,793	1,629	403,762	5,213	0.01	-68.75
12-2	12	418,656	890	412,966	3,716	1.38	-76.05
12-3	12	388,935	1,061	388,538	4,695	0.10	-77.40
13-1	13	450,275	1,535	446,797	9,833	0.78	-84.39
13-2	13	443,106	6,918	438,432	62,147	1.07	-88.87
13-3	13	455,373	3,877	458,384	68,239	-0.66	-94.32
14-1	14	478,885	5,214	477,390	66,026	0.31	-92.10
14-2	14	475,530	2,709	474,687	40,200	0.18	-93.26
14-3	14	475,474	11,705	474,193	12,644	0.27	-7.43
15-1	15	506,205	1,793	514,147	57,793	-1.54	-96.90
15-2	15	520,933	7,733	530,109	82,811	-1.73	-90.66
15-3	15	515,655	3,153	524,585	124,278	-1.70	-97.46

Table E.2. Partial interdiction versus full interdiction based on low level budget with $K = 6$

		Partial Interdiction MS-RSS or PGS		Full Interdiction ES			
Instance	#EF[1]	OV[2]	CPU[3]	OV[4]	CPU [5]	OV Gap	CPU Gap
No		(TL)	(sec)	(TL)	(sec)	(%)	(%)
4-1	4	133,817	69	125,372	0	6.74	0
4-2	4	95,897	68	90,141	1	6.39	6700.00
4-3	4	95,760	68	87,508	1	9.43	6700.00
5-1	5	151,820	148	139,848	1	8.56	14700.00
5-2	5	105,359	104	92,051	1	14.46	10300.00
5-3	5	122,576	290	110,220	1	11.21	28900.00
6-1	6	142,090[6]	71	139,208	2	2.07	3450.00
6-2	6	123,886	413	107,340	5	15.41	8160.00
6-3	6	132,308	306	128,499	3	2.96	10100.00
7-1	7	173,537	721	171,619	7	1.12	10200.00
7-2	7	172,353	641	169,774	9	1.52	7022.22
7-3	7	161,550	1425	157,841	26	2.35	5380.77
8-1	8	192,478	2012	188,557	21	2.08	9480.95
8-2	8	187,248	587	175,145	25	6.91	2248.00
8-3	8	196,436	3216	176,945	27	11.02	11811.11
9-1	9	220,723[6]	37,287	216,747	90	1.83	41330.00
9-2	9	191,439	5594	183,252	108	4.47	5079.63
9-3	9	223,521[6]	31,017	218,391	106	2.35	29161.32
10-1	10	236,703[6]	272,156	233,596	291	1.33	93424.40
10-2	10	232,957[6]	296,563	227,334	458	2.47	64651.75

[1]The number of the facilities in the system

[2]Attacker's partial interdiction objective value calculated via MS-RSS or PGS

[3]The solution time of MS-RSS or PGS with the related m value

[4]Attacker's full interdiction objective value calculated via ES

[5]The solution time of ES with the related m value

[6]The solution obtained from PGS

Table E.2. Partial interdiction versus full interdiction based on low level budget with $K = 6$ (Contd.)

Instance	#EF[1]	Partial Interdiction MS-RSS or PGS		Full Interdiction ES		OV Gap	CPU Gap
		OV[2]	CPU[3]	OV[4]	CPU [5]		
No		(TL)	(sec)	(TL)	(sec)	(%)	(%)
10-3	10	242,122[6]	211,454	229,739	238	5.39	88746.22
11-1	11	244,826	10,920	243,585	900	0.51	1113.33
11-2	11	242,505	8,997	240,670	2,981	0.76	201.81
11-3	11	244,814	9,116	236,688	2,439	3.43	273.76
12-1	12	253,023	9,174	247,633	2,327	2.18	294.24
12-2	12	277,464	5,601	272,005	1,663	2.01	236.80
12-3	12	232,252	5,256	232,908	1,902	-0.28	176.34
13-1	13	291,288	5,666	288,988	2,506	0.80	126.10
13-2	13	283,359	36,702	275,493	38,686	2.86	-5.13
13-3	13	310,922	34,866	308,637	54,780	0.74	-36.35
14-1	14	288,379	17,073	290,485	11,482	-0.72	48.69
14-2	14	291,356	13,583	287,979	20,699	1.17	-34.38
14-3	14	302,989	25,042	301,285	10,626	0.57	135.67
15-1	15	329,193	12,666	331,430	28,593	-0.67	-55.70
15-2	15	352,567	43,341	356,932	63,971	-1.22	-32.25
15-3	15	351,240	28,357	348,596	61,442	0.76	-53.85

APPENDIX F: BUDGET EFFECT IN INTERDICTION STRATEGIES

Table F.1. MS-RSS solutions to instance 7-1 based on different budget levels

Instance No	e_{tot} (TL)	OV[1] (TL)	CPU [2] (sec)
7-1	0	84,596	971
7-1	30,200	143,959	2,222
7-1	60,400	197,449	756
7-1	90,600	255,063	385
7-1	120,800	306,122	208
7-1	151,000	361,500	41

[1]Attacker's objective value calculated via MS-RSS
[2]The solution time of MS-RSS

Table F.2. MS-RSS solutions to instance 14-1 based on different budget levels

Instance No	e_{tot} (TL)	OV[1] (TL)	CPU [2] (sec)
14-1	0	128,441	45,419
14-1	60,800	244,079	50,838
14-1	121,600	362,519	11,809
14-1	182,400	479,841	5,473
14-1	243,200	597,557	2,163
14-1	304,000	713,000	1,162

[1]Attacker's objective value calculated via MS-RSS

[2]The solution time of MS-RSS

REFERENCES

1. Aksen, D., N. Piyade and N. Aras, “The Budget Constrained r-Interdiction Median Problem with Capacity Expansion”, *Central European Journal of Operations Research*, Vol. 18, No. 3, pp. 269-291, 2010.

2. Brooks, C., “The Introduction to Ameoba”, 1997, http://www.cs.usfca.edu / brooks/papers/amoeba.pdf, 10 October 2010.

3. Church, R. and C. ReVelle, “The Maximal Covering Location Problem”, *Papers of the Regional Science Association*, Vol. 18, pp. 101-118, 1974.

4. Church, R. L., M. P. Scaparra, R. S. Middleton, “Identifying Critical Infrastructure: The Median and Covering Facility Interdiction Problems”, *Annals of the Association of American Geographers*, Vol. 94, No. 3, pp. 491-502, 2004.

5. Church, R. L. and M. P. Scaparra, “Protecting Critical Assets: The r-interdiction Median Problem With Fortification”, *Geographical Analysis*, Vol. 39, No. 2, pp. 129-146, 2007.

6. Dempe, S., “Foundations of Bilevel Programming”, *Kluwer Academic Publishers*, Dordrecht, The Netherlands, 2002.

7. Israeli, E. and Wood, R.K., “Shortest-path Network Interdiction”, *Networks*, Vol. 4, pp. 97-111, 2002.

8. Gurson, A. P., “Simplex Search Behavior in Nonlinear Optimization”, 2000, http://www.cs.wm.edu/ va/CS495/gurson.pdf, 10 October 2010.

9. Gm, Z. H. and Floudas C.A., “Global Optimization of Mixed-Integer Bilevel Programming Problems”, *Computational Management Science*, Vol. 2, No. 3, pp. 181-212, 2005.

10. Humphrey, D.G. and Wilson, J.R. , “A Revised Simplex Search Procedure for the Stochastic Simulation Response Surface Optimization”, *Informs Journal on Computing*, Vol. 12, No. 4, Fall 2000.

11. H. von Stackelberg. *The Theory of the Market Economy*, William Hodge & Co., London, UK, 1952.

12. Luersen, M.A. and Riche, R., “Globalized Nelder-Mead Method for Engineering Optimization”, *Computers and Structures*, Vol. 82, pp. 1251-1260, 2004.

13. Moore, J.T. and Bard, J.F., “The Mixed-Integer Linear Bilevel Programming Problem”, *Operations Research*, Vol. 38, No. 5, pp. 911-921, 1990.

14. Motto, A. L., J. M. Arroyo and F. D. Galiana, “MILP for the Analysis of Electric Grid Security under Disruptive Threat” *IEEE Trans. on Power Syst.*, Vol. 20, No. 3, pp. 1357-1365, 2005.

15. O’Hanley, J.R., R. L. Church, and J. K. Gilless, “Locating and Protecting Critical Reserve Sites to Minimize Expected and Worst-Case Losses”, *Biological Conservation*, Vol. 34, pp. 130-141, 2007

16. Radio Free Europe - Radio Liberty: *Afghanistan: Mobile-Phone Towers Are Taliban’s New Target*, 2008, http://www.rferl.org/content/ article/1347757.html, 10 October 2010.

17. Scaparra, M. P. and R. L. Church, “An Exact Solution Approach for The Interdiction Median Problem With Fortification”, *European Journal of Operational Research*, Vol. 189, No. 1, pp. 76-92, August 2008.

18. Scaparra, M. P. and R. L. Church, “A Bilevel Mixed-integer Program for Critical Infrastructure Protection Planning”, *Computers and Operations Research*, Vol. 35, No. 6, pp. 1905-1923, June 2008.

19. Scaparra, M. P., Liberatore, F., and Daskin, M.S., “Analysis of Facility Protection

Strategies against Uncertain Numbers of Attacks: The Stochastic r-Interdiction Median Problem with Fortification", *Kent Business School, University of Kent*, Working Paper No. 176, 2008.

20. Smith, J.C., "Basic interdiction models", *Wiley Encyclopedia of Operations Research and Management Science*, 2010, http://eu.wiley.com/WileyCDA/Section/id-380764.html, 22 May 2010.

21. Spendley, W., Hext, G.R., Himsworth, F.R., "Sequential Application of Simplex Design in Optimization and Evolutionary Operations", *Technometrics*, Vol. 4, No. 4, pp. 441-461, 1962.

22. Walter, F.H., Parker, Jr. L.R., Morgan, S.L., and Deming,S.N. "Sequential Simplex Optimization: a Technique for Improving the Quality and Productivity in Search, Development, and Manufacturing", *CRC Press*, 1991.

23. Wen, U.P., Yang, Y.H., "Algorithms for Solving the Mixed-Integer Two-Level Linear Programming Problem", *Computers and Operations Research*, Vol. 17, No. 2, pp. 133-142, 1990.

24. Wolff, S.,"A Local and Globalized, Constrained and Simple Bounded Nelder-Mead Method", 2004, http://webuser.uni-weimar.de/wolff3/software/BNM-GBNM.pdf, 15 June 2009.

25. Wollmer, R., "Removing Arcs from a Network", *Operations Research*, No. 12, pp. 934-940, 1964.

26. Wood, R. K., "Deterministic Network Interdiction", *Mathematical and Computer Modelling* , Vol. 17, pp. 1-18, 1993.