A TRILEVEL DEFENDER-ATTACKER PROBLEM WITH PARTIAL INTERDICTION

by

İlknur Çoğal

B.S, Management Engineering, İstanbul Technical University, 1999

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

2013

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis supervisor Prof. Necati Aras for his invaluable guidance and my thesis co-supervisor Assoc. Prof. Deniz Aksen for their continuous guidance during the preparation of this thesis.

I would also like to thank to Prof. Kuban Altınel, Prof. Taner Bilgiç, and Prof. Tülin Aktin for sparing time in examining my thesis and taking part in my thesis jury.

Special thanks to my parents and my brother for their patience and endless support.

ABSTRACT

A TRILEVEL DEFENDER-ATTACKER PROBLEM WITH PARTIAL INTERDICTION

Terrorism is one of the most serious problems of today's world. Thus, the reliability and robustness of network systems with critical infrastructure are crucial. In this thesis, the best location-allocation strategy is sought to serve the customers in a network system in case of intentional disruptions caused by terrorist attacks. For this purpose, a trilevel mixed integer programming model is introduced. In the upper level, the system planner (leader) determines the facility locations, the capacities of the facilities, and the assignment of customers to the facilities. The attacker (follower) makes the choice of the interdiction fractions on the opened facilities in the middle level. In the lower level, the system planner sets the reassignment of the customers considering the capacity reduction of the facilities and decides which customers are served by outsourcing. In the upper level, a tabu search heuristic is used to locate facilities and commercial solver Cplex is employed to determine the capacities of the opened facilities and pre-attack allocations. The best interdiction strategy is found by the Electromagnetism Like Algorithm (EMLA) in the middle level. Finally, the post-attack allocations are found with Cplex. This solution method gives quite satisfactory results.

ÖZET

ÜÇ SEVİYELİ KISMÎ SALDIRI PROBLEMİ

Günümüzde terörizm en önemli problemlerden biri haline gelmiştir. Bu nedenle kritik altyapı sistemlerinin güvenilirliği ve sağlamlığı önem kazanmıştır. Bu çalışmada, üç seviyeli karışık tamsayılı programlama modeli kullanılarak; kasıtlı saldırı riski olan bir ağ sisteminde bulunması gereken tesislerin yerini belirleyebilmek ve hizmet alacak müşterileri bu tesislere atayabilmek için en iyi strateji belirlenmeye çalışılmıştır. Üst seviyede sistem planlayıcısı (lider), tesis yerlerine ve bu tesislerden hizmet alacak müşterilere karar verirken, orta seviyede saldırgan (takipçi), hangi tesislerin hangi oranla vurulacağını belirler. Alt seviyede ise sistem planlayıcısı azalan kapasiteleri göz önünde bulundurarak müşterilerin hizmet alacağı tesisleri belirler. Tesislerin kalan kapasitelerinin yeterli gelmemesi durumunda ise sistem dışından hizmet alımı yoluyla müşteri ihtiyaçlarını karşılar. Çözümde, üst seviyede Tabu arama ile tesis yerleşimi, ticari çözüm paketi Cplex ile müşteri ataması ve orta seviyede, Elektromagnetizm benzeri algoritma ile de saldırganın stratejisi belirlenir. Son olarak alt seviyede, Cplex kullanılarak saldırı sonrası müşteri atamaları yapılarak çözüme ulaşılır. Bu çözüm yöntemi dikkate değer sonuçlar vermiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a_i	Demand of customer i
$avInc$	available increment computed by $\frac{e_{tot} - e_{used}}{e_j}$, $j \in J'$
c_1	Cognitive parameter
c_2	Social parameter
c_d	Shipping cost per demand per unit distance
c_p	Unit cost of outsourcing customer demand (independent of distance)
d_{ij}	Euclidean distance between customer i and facility at site j
e_{tot}	Interdiction budget of the attacker
e_j	Cost of interdicting full capacity of facility at site j
e_{used}	$\sum_{j \in J'} e_j S_j$
f_j	Fixed cost of opening a facility at site j
$generation$	the iteration counter for the EMLA
$generationFactor$	The number of iterations for the EMLA
h	Unit cost of q more capacity acquisition for a facility
I	Set of demand nodes (customers)
$iter$	The iteration counter for the Main Algorithm
J	Set of candidate facility sites (locations)
J'	Set of opened facility sites (locations)
$maxIter$	The maximum number of iterations
$maxNonImpIter$	The maximum number of successive iterations during which the incumbent does not improve
$nonImpIter$	The counter of successive iterations during which the incumbent does not improve
$objBest$	The defender's best objective value found at the end of algorithm
$objBest_{neigh}$	The defender's objective value in the best neighborhood solution
$objVal_{curr}$	Defender's objective value at the current iteration

$objVal_{neigh}$	The defender's objective value in the neighborhood solution
p_{best}	Best remembered position in the population
p_i	Previous best position of particle i
q	Capacity multiplier
Q_j	Level of capacity acquisition at facility site j
r_1	Random number between 0 and 1
r_2	Random number between 0 and 1
S_j	Fraction of the capacity of facility j lost due to interdiction
U_{ij}	The assignment variable before attack
V_{ij}	The assignment variable after attack
v_i	Current velocity of particle i
w	Inertia weight
x_i	Current position of particle i
X_j	Binary variable shows opened facility at site$j$
ϵ	a small number which guarantees that the original inequality is satisfied strictly

LIST OF ACRONYMS/ABBREVIATIONS

BCRIMF-CE	r-interdiction median problem with capacity expansion
BFCLP	bilevel fixed charge location problem
BLPP	a bilevel programming problem
BPFIP	Bilevel partial facility interdiction problem
CPU	Central Processing Unit
EMLA	Electromagnetism Like Algorithm
FIPS	Fully Informed Particle Swarm
FIPSO	Fully Informed Particle Swarm Optimization
LLP	Lower level problem
IMF	Interdiction median problem with fortification
MCPC	Maximal covering problem with precedence constraints
MIP	Mixed Integer Problem
MLP	Middle level problem
PSO	Particle Swarm Optimization
RIC	r-interdiction covering problem
RIM	r-interdiction median problem
RIMF	r-interdiction median problem with fortification
TDAP-PI	Trilevel Defender-Attacker Problem with Partial Interdiction
ULP	Upper level problem
TS	Tabu search
TSH	Tabu Search Algorithm with Hashing

1. INTRODUCTION

Globalization is unquestionably the most important feature of twentieth century and it has changed the world. Now, many researchers deal with how globalization affects terrorism. Beside these academic studies, the World Trade Center attack on September 11 and bombings to the British Bank in 2003 in Turkey and train bombings in 2004 in Spain reveal that terrorist attacks are increasing with globalization and rapid changes in technology. These intelligent attacks embrace a great threat to the world, particularly to the critical infrastructure. Since the terrorists become global actors and benefit from telecommunication and information technology trends, they are able to get the necessary information to cause maximum disruption to network systems.

In this thesis we study a network system which provides a critical service to meet customers' demands. There are two players: a system planner who provides the service and a terrorist who plans to cause the major disruption to the system. They are also called leader and follower, respectively. Our problem can be considered as a game in which the system planner decides on the location of the facilities, their capacities and assignment of customers to them; while the attacker aims at the maximum disruption to the system to maximize the total cost of the service provision. Finally, the system planner modifies the allocation plan in response to the attacker's action. We call our problem Trilevel Defender-Attacker Problem with Partial Interdiction (TDAP-PI).

We formulate TDAP-PI as a trilevel Mixed Integer Programming (MIP) Problem. As trilevel problems are NP-hard [28], we use metaheuristics to develop a solution algorithm. In the upper level, we use a tabu search heuristic to determine the location of the facilities, the capacities of the facilities, and optimally set the assignment of the customers. In the middle level, the attacker determines the interdiction strategy using an Electromagnetism Like Algorithm (EMLA). Finally, in the lower level the system planner optimally computes the post-attack allocation by solving an MIP problem. We also apply Fully Informed Particle Swarm (FIPS) optimization algorithm, a variant of Particle Swarm Optimization (PSO), to solve the middle level problem. Since we

encounter the premature convergence for many instances, we implement the other solution technique EMLA, which is more efficient than FIPS with respect to the solution quality and time.

The major contribution of this study is to integrate location, capacity acquisition, customer assignment decisions of the system planner while taking into account the interdiction decision of the attacker. The partial interdiction concept we use is rarely studied in the literature. This feature increases the applicability of the proposed model. Since our problem cannot be solved with exact methods for large-size instances, there is a need to develop efficient solution methods.

The rest of this thesis is organized as follows. Chapter 2 presents the literature survey about studies on bilevel and trilevel programming and interdiction problems. Chapter 3 explains the problem formulation of TDAP-PI. Chapter 4 explains the solution procedure. Chapter 5 shows the computational results and finally Chapter 6 concludes the thesis.

2. LITERATURE SURVEY

2.1. Interdiction Problems

Deliberate sabotage and attacks are a crucial issue for critical service and supply systems found in both private and public sectors. Therefore, there is a need for analytical models to increase the system robustness. Major disruptions or disorders caused by intentional attacks to the system may be mitigated when networks are designed by considering reliability and resilience issues. Various models have been introduced in the literature with this motivation. The network reliability problems mainly deal with natural disasters. Some examples for reliability problems can be found in [1, 2]. Man-made terrorist attacks formulated as interdiction problems have been first investigated in military applications.

The first interdiction model within the context of network optimization is studied by Wollmer [3]. The author tries to determine the sensitivity of a transportation system when some of its roads are closed. This model aims at minimizing the maximal flow capacity from source to sink. Smith et al. [4] consider a problem of building or fortifying a network to defend against enemy attacks and formulate the problem as a trilevel, two player game. In the upper level, the network designer builds a network to maximize profit before interdiction while the enemy inflicts damage by reducing the capacity of certain arcs to reduce the designer's profit at the middle level. In the lower level, the designer tries to maximize post-interdiction profit with the remaining capacity in the network. McMusters and Mustin [5] consider a similar model by introducing incremental interdiction on arcs. The reader is referred to Ghare et al. [6] to examine the same type of model with complete interdiction on arcs. In addition to arc interdiction, node interdiction is investigated in [7, 8]. The latter allows both full and partial interdictions. Several studies [9–12] try to find the arcs that cause maximum increase in the shortest distance between two specific nodes if they are removed.

An interdiction model within the context of a service network is first studied by Church et al. [13]. The model tries to identify the most critical facilities in a service system. They introduce two models: r-interdiction median problem (RIM) and the r-interdiction covering problem (RIC). These models seek the r facilities among p facilities, which cause the most damage in the service level when lost. RIM aims to determine which set of r facilities is interdicted to maximize demand-weighted total distance while RIC's objective is to determine the r facilities to maximize reduction in the coverage of demand.

Church and Scaparra [14] extend the RIM model by adding fortification decisions and call the resulting problem interdiction median problem with fortification (IMF). The fortification concept is used for protecting or hardening the facilities against attacks. IMF determines q facilities out of p facilities to be fortified so that the least disruption is caused by the attacker when r facilities are interdicted.

In [15], Scaparra and Church reformulate the IMF as a maximal covering problem with precedence constraints (MCPC). With this approach the authors provide upper and lower bounds that are used to reduce the size of the original problem, which is then solved via a general-purpose MIP solver. In the next study of the authors [16], a new solution methodology based on a bilevel formulation of r-interdiction model with fortification (RIMF) has been proposed. In RIMF, the defender determines which facilities are fortified in the upper level while a potential attacker decides which unprotected facilites to attack to cause maximal damage in the system. They present a specialized tree search algorithm and try to solve RIMF problem instances of large size. The stochastic version of RIMF called S-RIMF is studied by Liberatore et al. [17]. The authors add uncertainty which comes from the probability distribution over the number of facilities to be attacked.

RIMF is extended by Aksen et al. [18] in such a way that a budget constraint on the total fortification cost is introduced instead of fixing the number of fortified facilities. In this model referred to as the budget constrained r-interdiction median problem with capacity expansion (BCRIMF-CE), customers' assignment to the facilities may

change after interdiction since the capacity of non-interdicted facilities are increased to serve the demand. Thus, the authors include the capacity expansion cost into their model and solve this bilevel model by an implicit enumeration algorithm applied on a binary tree.

Aksen and Aras [19] combine the BCRIMF-CE model by including facility location decisions of the defender. In this bilevel fixed charge location problem (BFCLP), the defender determines the number and protection status of the facilities to be opened to minimize the total cost of demand satisfaction before and after an attack. The authors propose two methods to solve BFCLP. The first one is a tabu search heuristic and the second one is a sequential solution method. The former seeks the best alternative among the large number of facility location-protection plans. In the latter, the facility location and protection decisions are separated and the defender's fixed charge facility location problem is optimally solved first. Then, an implicit enumeration algorithm is applied on a binary tree to decide on the protected facilities. To the best of our knowledge, this is the first study combining the decisions of the facility location, protection and interdiction in a static Stackelberg game between a leader and a follower.

A recent extension to the interdiction models is studied in Losada *et al.* [20]. They present a bilevel MIP for protecting an uncapacitated median type facility network against worst-case losses by including the concept of facility recovery time on system performance and the possibility of multiple distruptions over time. In that study, the protection of a facility involves not only preventing the facility failure but also decreasing recovery time after interdiction.

The above mentioned works mostly concentrate on uncapacitated problems that are unrealistic when considering the real life problems with limited resources. Scaparra and Church [21] study the mitigation of the impacts of a disaster by protecting one or more capacitated facilities. The authors formulate the problem as a trilevel optimization model and propose a solution appraoach based on a tree search strategy. First, they collapse the two bottom-level problems into a single-level problem using duality techniques and solve the new bilevel problem by the implicit enumeration algorithm

originally proposed in [15].

Our study puts an emphasis on facility interdiction to find the criticality of system components.

2.2. Multilevel Programming Models

Multilevel optimization problems are mathematical programs which include interacting decision making units within a hierarchical structure [22, 23]. Each level makes its decision by knowing the decisions of the upper level player to optimize its benefits. Multilevel optimization is closely related to the economic problem of Stackelberg [24] found in the field of game theory [25]. Bilevel and trilevel programming problems are a special case of multilevel optimization with two and three levels, respectively. Interdiction problems are mostly formulated within a bilevel programming framework, where an upper level problem is constrained by the lower level problem. There are two players: the leader (a system planner) and the follower (a potential attacker). The leader is assumed to anticipate the reactions of the follower and choose its optimal strategy accordingly. The follower also knows about the leader's strategy. The Bilevel Programming Problem (BLPP) is NP-hard as shown in [26, 27]. In [28] the mixed integer linear BLPP has been investigated from an algorithmic point of view using branch and bound techniques. Also Wen and Yang [29] study the same problem with different branch and bound solution methodology. Gümüş and Floudas propose two deterministic global optimization methods to solve mixed integer nonlinear BLPPs [30]. Denegre and Ralphs improve the algorithm given in [28] as including the cutting plane techniques [31].

A BLPP can be stated mathematically as follows [30]:

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

s.t.

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

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

where $\mathbf{y}$, for each value of $\mathbf{x}$, is the solution of the following problem:

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

s.t.

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

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

$$\mathbf{x},\ \mathbf{y} \in \mathbb{R}^n \tag{2.7}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the vectors of upper and lower level problem variables, respectively. $F(\mathbf{x},\mathbf{y})$ and $f(\mathbf{x},\mathbf{y})$ are the objective functions of the upper and lower level problems, respectively. (2.2) and (2.3) are the upper level constraints and (2.5) and (2.6) are the lower level constraints.

3. PROBLEM FORMULATION

Our problem is modeled within the framework of trilevel mixed integer programming. In the upper level problem (ULP), the system planner in the role of the leader determines where to open p service facilities among the potential sites, the capacity of each facility to be opened and assignment of customers to open facilities. The attacker, who acts as the follower, decides which facilities to interdict in order to increase the cost of the system in the middle level problem (MLP). Then, in the lower level problem (LLP) the system planner tries to reallocate the customers to be served by considering the remaining capacity of the facilities available after the interdiction. Partial interdiction of a facility is possible. In addition to this, the system planner and the attacker have perfect information regarding their decisions of locating the facilities, assignment of the customers and interdiction of the facilities. It means that the system planner assigns a customer by considering the available capacity of the facilities after interdiction. If the demand is not met with the remaining capacity because of the disruption caused by the attack, the outsourcing option is used by the system planner. It should be emphasized that the important assumption that the attacker has a limited budget to cause the largest capacity reduction in the system in the MLP. Also, single sourcing is assumed in the model with respect to assigning customers to the facilities, which means that a demand node can be served by only one facility.

After the leader's location, allocation and capacity decisions are made, they become known to the follower, who gives his optimal reaction by deciding which interdiction strategy causes the most distruptive effect on the system and determines the interdiction fractions. Last, the defender reallocates the customers to the facilities knowing which facilities are attacked.

A formal description of our model is given as follows:

Table 3.1. Declaration of index sets and parameters.

Index Sets	Explanations
I	set of customers, I=$\{1,...,n\}$
J	set of candidate facility sites, J=$\{1,...,m\}$
J'	set of opened facilities
Parameters	Explanations
a_i	demand of customer i
c_d	unit shipping cost per unit distance
c_p	unit cost of outsourcing customer demand (independent of distance)
d_{ij}	Euclidean distance between customer i and facility at site j
e_j	cost of interdicting the full capacity of facility at site j
e_{tot}	interdiction budget of the attacker
f_j	fixed cost of opening a facility at site j
h	unit cost of capacity acquisition for a facility
q	capacity multiplier

Decision Variables

$$X_j = \begin{cases} 1 & \text{if a facility is opened at site } j, \\ 0 & \text{otherwise,} \end{cases}$$

$$U_{ij} = \begin{cases} 1 & \text{if customer } i \text{ is assigned to facility at site } j \text{ before the attack,} \\ 0 & \text{otherwise,} \end{cases}$$

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

Q_j = level of capacity acquisition at facility site j (capacity can be acquired in bulk only),

S_j = fraction of the capacity of facility j lost due to interdiction.

The mathematical model of TDAP-PI is given as follows:

$$\min_{\mathbf{X},\mathbf{Q},\mathbf{U}} \sum_{j\in J} f_j X_j + h\sum_{j\in J} Q_j + c_d \sum_{i\in I}\sum_{j\in J} a_i d_{ij} U_{ij} \tag{3.1}$$
$$+ c_d \sum_{i\in I}\sum_{j\in J} a_i d_{ij}(1-U_{ij})V_{ij} + c_p \sum_{i\in I} a_i (1 - \sum_{j\in J} V_{ij})$$

s.t.

$$\sum_{j\in J} U_{ij} = 1 \qquad \forall i \in I \tag{3.2}$$

$$\sum_{i\in I} a_i U_{ij} \le q Q_j \qquad \forall j \in J \tag{3.3}$$

$$\sum_{i\in I} a_i U_{ij} \ge q(Q_j - 1) \qquad \forall j \in J \tag{3.4}$$

$$U_{ij} \le X_j \qquad \forall i \in I, \forall j \in J \tag{3.5}$$

$$Q_j \le \left(\frac{\sum_{i\in I} a_i}{q} + 1 \right) X_j \qquad \forall j \in J \tag{3.6}$$

$$X_j, U_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J \tag{3.7}$$

$$Q_j \in \mathbb{Z}^+ \qquad \forall j \in J \tag{3.8}$$

$$\max_{\mathbf{S}} Z_{\text{att}}(\mathbf{S}) = c_d \sum_{i\in I}\sum_{j\in J'} a_i d_{ij}(1-U_{ij})V_{ij} + c_p \sum_{i\in I} a_i (1 - \sum_{j\in J'} V_{ij}) \tag{3.9}$$

s.t.

$$\sum_{j\in J'} S_j e_j \le e_{tot} \tag{3.10}$$

$$0 \le S_j \le 1, \qquad \forall j \in J' \tag{3.11}$$

$Z_{def}(\mathbf{S})$ is defined as the optimal objective value of the following integer problem:

$$\min_{\mathbf{V}} Z_{att}(\mathbf{S}) = c_d \sum_{i\in I}\sum_{j\in J'} a_i d_{ij}(1-U_{ij})V_{ij} + c_p \sum_{i\in I} a_i(1-\sum_{j\in J'} V_{ij}) \tag{3.12}$$

s.t.

$$\sum_{i\in I} a_i V_{ij} \leq (1-S_j)qQ_j \qquad \forall j \in J' \tag{3.13}$$

$$\sum_{j\in J'} V_{ij} \leq 1 \qquad \forall i \in I \tag{3.14}$$

$$V_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J' \tag{3.15}$$

In the above TDAP-PI formulation, (3.1)–(3.8) represent the ULP, (3.9)–(3.11) represent the MLP, and finally (3.12)–(3.15) correspond to the LLP.

Objective function (3.1) shows the total cost of the system planner including pre-attack and post-attack costs. The first component of this objective is the fixed cost of opening a facilty. The second component is the fixed cost of capacity acquisition when opening a facility, and the third component is the total shipment cost to serve the customers. The fourth and fifth components are related the post-attack costs. The multiplication of V_{ij} with $(1-U_{ij})$ in the fourth component ensures that the system planner will incur no additional cost for the customers who are served by the same facilities in the post-attack period. Thus, the last component of (3.1) is the total outsourcing cost incurred after the attack. Constraints (3.2) ensure that each customer is assigned to exactly one facility before the interdiction. Constraints (3.3) require that customers cannot be assigned to the facilities if there is no available capacity. This constraint set also assures that the capacity usage does not exceed the available capacity. Constraints (3.4) and constraints (3.6) are for integrality and sufficiency of the capacity variables (Q_j), respectively. Constraints (3.6) also force $Q_j = 0, \forall j \in J$ if the facility is not opened at site j, otherwise they assure that enough capacity is present to meet the total demand of the customers. Constraints (3.5) imply that a customer cannot be served from a facility if it is not opened. Constraint sets in (3.7)

and (3.8) show the binary variables and the integer variables in ULP, respectively.

The objective function of the attacker shown in (3.9) as $Z_{\text{def}}(\mathbf{S})$ is comprised of the fourth and fifth components of (3.1). Constraint (3.10) represents the interdiction budget restriction of the attacker. Constraints in (3.11) are trivial lower and upper bounds on the fractional interdiction variables S_j. Recall that partial interdiction is possible in our model. The wage of the partial interdiction concept in a facility interdiction problem was introduced in [32]. Later, Liberatore et al. [34] implemented partial interdiction in a trilevel facility fortification-interdiction problem. However, the authors optimize a protection plan for large area disruptions which are mostly caused by natural disasters such as earthquakes, storms, floods, fires, and hurricanes.

The defender's objective in the LLP is the same as in (3.9) except for the sense of optimization. S_j variables represent the attacker's decision, which are input parameters for the LLP. Constraints (3.13) state the capacity restriction on the facilities. The right-hand side of this inequality is constant since S_j's are determined at the middle level problem. This constraint set requires that the demand of customers assigned to facility j does not exceed its remaining capacity after the inderdiction. Constraint set (3.14) states that there might be customers who are not assigned to any facility due to the capacity loss caused by the attacker's actions. Finally, (3.15) are post-attack binary assignment variables of the defender.

4. SOLUTION PROCEDURE

In this study we propose a heuristic solution procedure to obtain efficient solutions to our problem, which is modeled as a mixed integer trilevel programming problem including binary variables in the upper and lower levels.

In the upper level of TDAP-PI, a tabu search heuristic, originally proposed in [19], is used to explore the best location-allocation plan with the optimal capacity decision for the facilities. In the middle level, two population-based metaheuristics are employed: Fully Informed Particle Swarm (FIPS) Optimization proposed by Kennedy and Mendes [35] and Electromagnetism Like Algorithm (EMLA) introduced by Birbil and Fang [36]. These methods have become popular as a global optimization solution technique. After the system planner makes location and capacity decisions in the upper level, the attacker makes interdiction decisions in the middle level, while the system planner finds the best allocation plan with the remaining capacity of the facilities in the lower level. The last problem is solved exactly using general purpose MIP solver Cplex. Since interdiction fractions in the middle level cannot be determined optimally, we find lower bounds for the system planner in the upper level. In this chapter the detailed explanation for the solution methods will be given.

4.1. Tabu Search Algorithm with Hashing (TSH)

Tabu search (TS) is a metaheuristic algorithm that can efficiently solve many real-life combinatorial optimization problems. TS algorithm combined with other solution techniques has been used to obtain better solutions to the problems in production planning and scheduling, resource allocation, network design, financial analysis, telecommunications, portfolio planning, supply chain management, forecasting, data mining, biocomputation, molecular design, and many other areas [49].

Important attributes of TS are adaptive memory structure and responsive exploration strategies. Use of memory helps to rationally exploit the search space. This

approach provides learning from past decisions and adjusting the search strategies with flexible memory framework. In the algorithm, intensification and diversification strategies are implemented to guide the search. Intensification strategies are used to explore more thoroughly with identifying a set of elite solutions to incorporate good features into new search [49]. The search process can be shifted to the unexplored areas with restricting to visit same solutions to escape from local optima using diversification strategies of TS algorithm.

4.2. Fully Informed Particle Swarm Optimization

The Particle Swarm Optimization (PSO) algorithm developed by Kennedy and Eberhart [37] is a stochastic and population-based technique. PSO was discovered by simulation of a simplified social model that was inspired by the studies in [38,39] related to the movement of organisms in a bird flock or fish school. PSO is an evolutionary computation technique and does not require complex mathematical operations. Therefore, it is computationally inexpensive. Its main application areas are given in [40] as image and video analysis, design and restructuring of electricity networks and load dispatching, scheduling, design and optimization of communication networks, combinatorial optimization problems, robotics, modeling, sensors and sensor networks, applications in computer graphics and visualization, and security and military applications.

In a PSO algorithm, each individual (particle) represents a potential solution in the population (swarm) and moves in the search space with a velocity. The velocity of the particle is adjusted using its current position and best position as well as the best position of all particles found so far in the population. Each particle is composed of three d-dimensional vectors, where d is dimensionality of the search space [40]. It is known as standard PSO or gbest version of PSO. This name is coming from neighborhood structure or population topology. In these types of models the target particle is affected by the best neighbor in the population and this topology is an example of static topology [40].

The PSO algorithm can be described as follows:

Table 4.1. Declaration of variables and parameters.

$\boldsymbol{x}_i$	current position of particle i
$\boldsymbol{v}_i$	current velocity of particle i
$\boldsymbol{p}_i$	previous best position of particle i
$\boldsymbol{p}_{best}$	best remembered position in the population
w	inertia weight
c_1, c_2	cognitive and social parameters
r_1, r_2	random number between 0 and 1
t_{max}	maximum number of iterations
k	number of particles

(i) Step 1: Initialize parameters and variables:

- Set constants c_1, c_2, t_{max}, and k.
- Initialize random positions ($\boldsymbol{x}_i$) and velocities ($\boldsymbol{v}_i$) for these particles.
- t=1

(ii) Step 2: Do the following if termination criterion is not satisfied:

- Evaluate the fitness value of all particles.
- Compare the fitness value of each particle i with the value of the best particle $\boldsymbol{p}_{\text{best}}$ in the swarm. If current value (with its position $\boldsymbol{x}_i$) is better than $\boldsymbol{p}_{\text{best}}$ and $\boldsymbol{p}_i$, set $\boldsymbol{p}_{\text{best}}$=$\boldsymbol{x}_i$ and $\boldsymbol{p}_i$=$\boldsymbol{x}_i$
- Update all particle velocities and positions according to the following equations:

$$\begin{aligned} \boldsymbol{v}_i(t+1) = &\quad w\boldsymbol{v}_i(t) + c_1 r_1(\boldsymbol{p}_i(t) - \boldsymbol{x}_i(t)) + c_2 r_2(\boldsymbol{p}_{best}(t) - \boldsymbol{x}_i(t)) \\ \boldsymbol{x}_i(t+1) = &\quad \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) \end{aligned}$$

- $t = t + 1$ and go to Step 2.

Although PSO is easy to implement and competitive with the other evolutionary solution techniques, it may give sub-optimal solutions [41] with a premature convergence because of gbest topology. Therefore, many variants of PSO have been sug-

gested in the literature to overcome this undesirable issue. Some have tried to update the velocity formula by adding different parameters while others have proposed hybrid techniques. Shi and Eberhart [42] revise the velocity update formula by adding an inertia weight that reduces the importance of the maximum velocity (V_{max}). In [43], the authors incorporate constriction coefficient to the velocity update formula to restrain particle velocities so as to stay in the search space after an acceptable number of iterations. Noel and Janett [44] and Wang [41] combine a gradient-based algorithm with the PSO algorithm in their studies.

Fully Informed Particle Swarm (FIPS) in which social influence comes from the group norm [45] is proposed by Mendes [45] to revise the interaction of the particles which are affected by their own previous best performance and the best particle in their own neighborhood. The particle is affected by all its neighbors. This solution procedure provides all the information available in the neighborhood of the target particle. It is obvious that a particle has almost no power to solve a problem without interaction with the other particles [40]. Mendes [45] showed that FIPS with good parameters outperformed the standard PSO. The velocity update and position equations is changed as follows:

$$\begin{aligned} \boldsymbol{v}_i(t+1) &= w\boldsymbol{v}_i(t) + \frac{1}{k_i}\sum_{n=1}^{k_i} c_n r(\boldsymbol{p}_{in}(t) - \boldsymbol{x}_i(t)) \\ \boldsymbol{x}_i(t+1) &= \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) \end{aligned}$$

where k_i is the number of neighbors of particle i, $\boldsymbol{p}_{in}$ is the position of the nth neighbor of particle i and r is random number between 0 and 1.

4.3. Electromagnetism Like Algorithm

Birbil and Fang [36] propose the EMLA, which is a population-based metaheuristic using an attraction-repulsion mechanism to move the points (particles) to obtain a near-optimal solution. Every point charged with electrons and protons represents a solution. This charge is related to the objective function value and sets the magnitude

of attraction or repulsion of the point. Better solutions are charged more than the others. After calculating the charges of the particles using their objective value, particles need a direction to move towards the better areas of the search space. Thus, forces are evaluated with using the interactions among the charged particles. It is needed to mention that no signs are attached to the charge of a particle. The direction of applied forces on a particle is determined by comparing their objective values of the points. As can be seen in the following formula, a point with a better objective attracts the other one as in the first case and the worse one repels the other points with better objective value as in the second case for a maximization problem.

$$F_i = \sum_{i \neq \text{J}}^{k} \begin{cases} (\boldsymbol{x}_j - \boldsymbol{x}_i) \frac{q_i q_j}{(||\boldsymbol{x}_j - \boldsymbol{x}_i||)^2} & \text{if } f(\boldsymbol{x}_j) > f(\boldsymbol{x}_i) \\ (\boldsymbol{x}_i - \boldsymbol{x}_j) \frac{q_i q_j}{(||\boldsymbol{x}_j - \boldsymbol{x}_i||)^2} & \text{if } f(\boldsymbol{x}_j) \leq f(\boldsymbol{x}_i) \end{cases} , \forall i \tag{4.1}$$

Debels et al. [46] combine scatter search with EMLA to solve resource constrained project scheduling problems. In [47] a hybrid algorithm that combines the EMLA and genetic operators to find the best schedule for a single machine scheduling problem. Naderi et al. [48] propose a solution procedure for a flow shop scheduling problem based on electromagnetism-like algorithm. The authors show that EMLA finds better solutions than Simulated Annealing metodology in almost all cases they consider.

EMLA for a maximization problem is as follows:

(i) Step 1: Initialize parameters and variables:
   (a) Set k as the number of points in the population and $maxIter$ as the maximum number of iterations.
   (b) Randomly initialize points $\boldsymbol{x}_i$, $i = 1, ..., k$.

(ii) Step 2: Total Force calculation:

(a) Evaluate charge of each point as:

$$q_i = \exp\left(- m \frac{f(\boldsymbol{x}_i) - f(\boldsymbol{x}_{best})}{\sum_{j=1}^{k} (f(\boldsymbol{x}_j) - f(\boldsymbol{x}_{best}))} \right) \quad \forall i \tag{4.2}$$

(b) Compute total force (F_i) exerted on point i as follows:

$$F_i = \sum_{i \neq \mathrm{J}}^{k} \begin{cases} (\boldsymbol{x}_j - \boldsymbol{x}_i) \frac{q_i q_j}{(||\boldsymbol{x}_j - \boldsymbol{x}_i||)^2} & \text{if } f(\boldsymbol{x}_j) > f(\boldsymbol{x}_i) \\ (\boldsymbol{x}_i - \boldsymbol{x}_j) \frac{q_i q_j}{(||\boldsymbol{x}_j - \boldsymbol{x}_i||)^2} & \text{if } f(\boldsymbol{x}_j) \leq f(\boldsymbol{x}_i) \end{cases} , \forall i \tag{4.3}$$

(iii) Step 3: Movement along the direction of the force:

(a) Set $\lambda \in U[0,1]$ as a random step length, upper bound $\boldsymbol{u}$ and lower bound $\boldsymbol{l}$ of $\boldsymbol{x}_i$'s

(b) Move particles in the direction $\widetilde{F}_i = \frac{F_i}{||F_i||}$ with the following equation:

$$\boldsymbol{x}_i = \begin{cases} \boldsymbol{x}_i + \lambda \widetilde{F}_i (\boldsymbol{u} - \boldsymbol{x}_i) & \text{if } F_i > 0 \\ \boldsymbol{x}_i + \lambda \widetilde{F}_i (\boldsymbol{x}_i - \boldsymbol{l}) & \text{otherwise} \end{cases} \quad i \in [1, k] \tag{4.4}$$

4.4. Our Solution Procedure

In our trilevel mixed integer problem, we use TS algorithm with hashing (TSH) to locate the facilities, which are fixed to obtain the optimal capacities and allocate the customers to the best possible facilities in the upper level. Then, in the middle level, EMLA is used to obtain the best destructive strategy of the attacker to increase the system cost. In the lower level, the system planner decides the new allocation plan to reduce the disruptive effects of the attack. Key features of our solution procedure are explained in the following subsections.

4.4.1. Neighborhood Structure in TSH

Three types of moves are applied to locate the facilities within TSH. These are 1-Add, 1-Drop, and 1-Swap moves. In every interation of TSH, these moves are exe-

cuted and the best move is chosen before sending the facility location plan to obtain capacities. These moves can be explained as follows:

Table 4.2. Types of Moves.

1-Add	One of the closed facilities is opened.
1-Drop	One of the opened facilities is closed.
1-Swap	One of the opened facilities is replaced by one of the closed facilities.

In our problem, the number of opened facilities may change in every iteration of the algorithm since the number of facilities to be opened is not fixed and determined by the trade-off between the various cost components. Hence the fixed cost of opening a facility is compromised with the pre-attack demand weighted traveling cost. Let p be the number of facilities opened at the current iteration of the solution process. Therefore $m - p$ is the number of closed facilities since m is the number of potential location sites. The number of 1-Add, 1-Drop, and 1-Swap moves to be executed is given as $m - p$, $p - 1$, and $p(m - p)$, respectively. If we apply p 1-Drop moves, we create an infeasible solution as all facilities are closed.

4.4.2. Hash Strategy in TSH

Throughout the iteration of TSH we apply hashing strategy that has been proposed in [19] to prevent visiting the same solutions called cycling. This can be done with creating a tabu list that is able to record the recently visited solution or the attributes of the solutions. The number of iterations to store the visited solutions is one of the most important parameters for efficient use of short term memory of TS algorithm. In our solution algorithm hash values for the solutions are computed and stored sequentially in a list. After applying one of three moves (1-Add, 1-Drop, 1-Swap) we calculate the hash value of the newly created solution and search the hash list whether it is included there. We do not continue with this move to send to Cplex solver if it is found in the list. This kind of hash list is effective with respect to memory usage when compared to keeping track of the attributes which are mostly not numerical. Hash

values are calculated as follows:

$$\text{Hash}(\sigma) = \sum_{j=1}^{m} \begin{cases} 2^{m-j} & \text{if facility } j \text{ is opened} \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

where j is the facility index.

In the hash list, the objective value and the move type of each solution are added besides the hash value of the solution. This allows us to recall and compare the objective value of the candidate solution with the best neighboring solution without computation if the hash value is found in the list. If not found, the objective value is computed and its hash value, objective value, and move type are added to the list in the right position without perturbing the ascending order of hash values that is very important for the search method. When searching the hash value in the list the algorithm starts from both head and tail of the list and if not found, the list is shrunk when removing current head and tail and continue with new head and tail of the smaller list until finding the searched hash or reaching the middle of the list from both sides.

4.4.3. Budget Adjustment Procedure in EMLA

In the middle level the attacker's interdiction strategy is incorporated. The total interdiction budget of the attacker may not be fully utilized when EMLA is applied to reach better solutions in the search space. This means $e_{tot} > \sum_{j \in J'} e_j S_j$ or $e_{tot} < \sum_{j \in J'} e_j S_j$ may hold true. Thus we should apply an upscaling or downscaling procedure to make e_{tot} exactly equal to the cost of facility destruction. After finding S_j variables in the current iteration, we need to check whether there is a budget overutilization. In that case, we decrease the values of S_j variables iteratively starting from the facility that has the highest cost for interdicting its full capacity. If there exists a budget underutilization, the algoritm increases the values of S_j variables starting from the facility with the lowest interdiction cost. The pseudocode can be found in Figure 4.1 with the additional notation as follows:

e_{used}	$\sum_{j \in J'} e_j S_j$
$avInc$	available increment computed by $\frac{e_{tot} - e_{used}}{e_j}$, $j \in J'$
ϵ	a small number which guarantees that the original inequality is satisfied strictly

4.4.4. Pseudocode of Our Solution Algorithm

The important features of our trilevel solution procedure are explained in the previous subsections. The solution algorithm can be summarized by partitioning it into sub-algorithms, which are given in Figures 4.3, 4.4, 4.5, and **??** with the following additional notation:

$iter$	the iteration counter for the main algorithm
$maxIter$	the maximum number of iterations
$nonImpIter$	the counter of successive iterations during which the incumbent does not improve
$maxNonImpIter$	the maximum number of successive iterations during which the incumbent does not improve
$generation$	the iteration counter for EMLA
$generationFactor$	the maximum number of iterations for EMLA
$popSize$	the population size for EMLA
$objVal_{curr}$	defender's objective value at the current iteration
$objBest_{neigh}$	the defender's objective value of the best neighborhood solution
$objVal_{neigh}$	the defender's objective value in the neighborhood solution
$objBest$	the defender's best objective value found by the algorithm.

Our proposed algorithm is trilevel mixed integer problem as explained before. At the beginning of the algorithm, the locations are chosen from the potential facility sites with one of the moves mentioned in Section 4.4.1. After determining the locations of the facilities to be opened, we find the capacities of these facilities and allocate the

```
1: for each particle do
2:   if (e_used − e_tot) < 0 then
3:     Sort e_j as ascending order
4:     for each facility j begining one with the smallest e_j do
5:       while e_used − e_tot < −ϵ and e_used − e_tot > ϵ do
6:         Compute avInc;
7:         if S_j + avInc ≥ 1 then
8:           Set S_j = 1
9:         else
10:          Add avInc to S_j
11:        end if
12:      end while
13:    end for
14:  else if (e_used − e_tot > 0) then
15:    for each facility j begining one with the biggest e_j do
16:      while e_used − e_tot < −ϵ and e_used − e_tot > ϵ do
17:        Compute avInc= (e_tot-e_used)/e_j, j ∈ J′;
18:        if S_j + avInc ≤ 0 then
19:          Set S_j = 0
20:        else
21:          Add avInc to S_j
22:        end if
23:      end while
24:    end for
25:  end if
26: end for
```

Figure 4.1. Pseudocode of the Budget Adjustment Procedure.

customers to them using Cplex 12.1 by the following model:

$$\min_{\mathbf{Q},\mathbf{U}} \quad \sum_{j\in J'} f_j + h\sum_{j\in J'} Q_j + c_d \sum_{i\in I}\sum_{j\in J'} a_i d_{ij} U_{ij} \tag{4.6}$$

s.t.

$$\sum_{j\in J'} U_{ij} = 1 \qquad \forall i \in I \tag{4.7}$$

$$\sum_{i\in I} a_i U_{ij} \leq q Q_j \qquad \forall j \in J' \tag{4.8}$$

$$\sum_{i\in I} a_i U_{ij} \geq q(Q_j - 1) \qquad \forall j \in J' \tag{4.9}$$

$$U_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J' \tag{4.10}$$

$$Q_j \in \mathbb{Z}^+ \qquad \forall j \in J' \tag{4.11}$$

The explanation of the above model is given in Chapter 3.

Also this step computes the defender cost including the cost of opening facilities and serving customers. By fixing the facilities, their capacities, and customer allocations before the attack, we find the best interdiction strategy for the attacker by implementing EMLA. Once the S_j values are calculated, we allocate the customers to the facilities and calculate the attacker objective value by solving the following model:

$$\min_{\mathbf{V}} \quad c_d \sum_{i\in I}\sum_{j\in J'} a_i d_{ij}(1-U_{ij})V_{ij} + c_p \sum_{i\in I} a_i(1-\sum_{j\in J'} V_{ij}) \tag{4.12}$$

s.t.

$$\sum_{i\in I} a_i V_{ij} \leq (1-S_j)qQ_j \qquad \forall j \in J' \tag{4.13}$$

$$\sum_{j\in J'} V_{ij} \leq 1 \qquad \forall i \in I \tag{4.14}$$

$$V_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J' \tag{4.15}$$

The explanation of this model is given in Chapter 3.

These allocations can be called after-attack allocations. Finally, the system cost is computed as adding the cost of the upper level defender cost and the attacker cost. This cost and the related findings are recorded as the result of one move for the current iteration. After calculating the objective values of all the possible moves for this iteration, the best move is determined and recorded in the hash list, if it was not found before, according to the hash strategy explained in Section 4.4.2.

After performing a predetermined number of iterations, the best facility location set, allocations of customers to the best locations and the best interdiction strategy are reported.

The flowchart of the solution algoritm is provided in Figure 4.2.

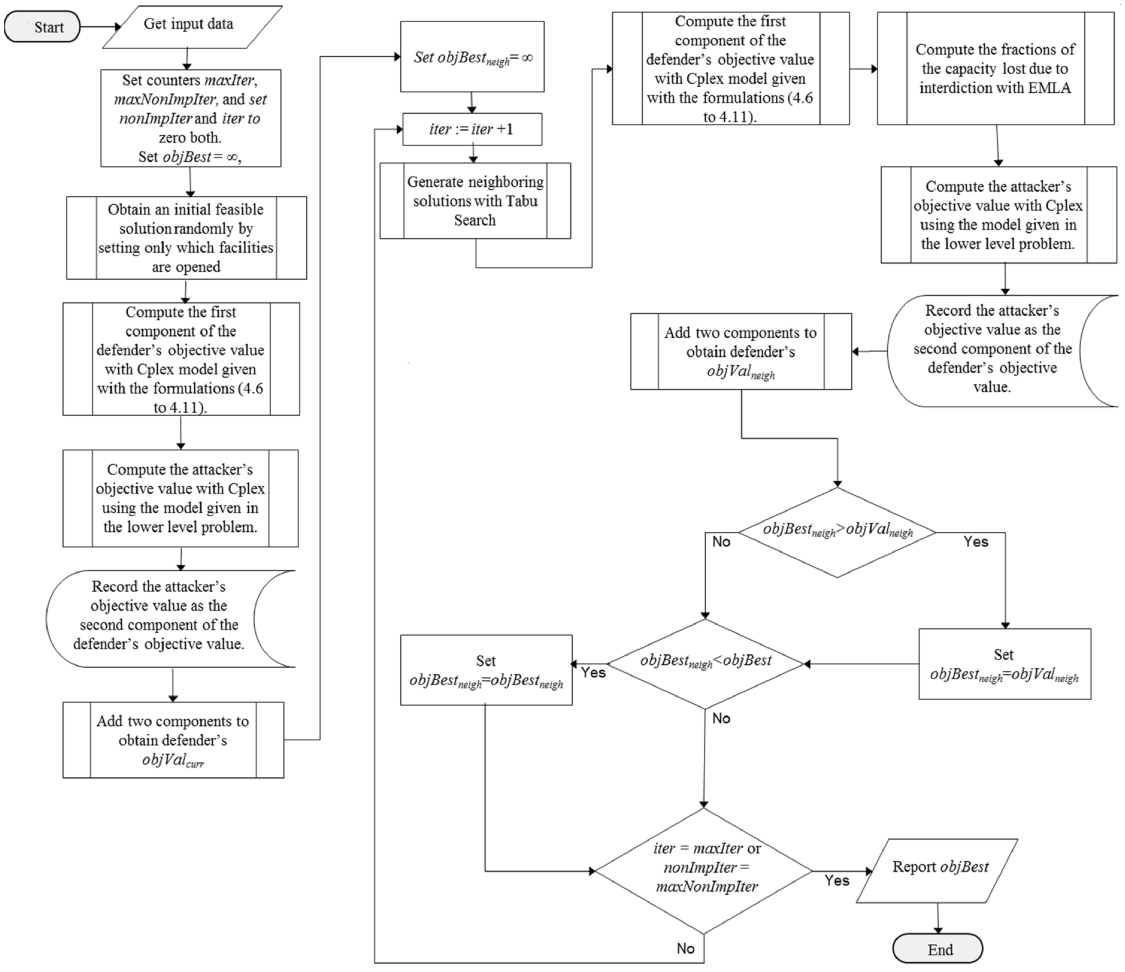

Figure 4.2. The flowchart of the solution algorithm for TDAP-PI.

1: **Initialization**
2: Set counters $maxIter$, $maxNonImpIter$, $nonImpIter$ to 0, $iter$ to 0;
3: Obtain an initial feasible solution randomly by setting only which facilities are opened;
4: Compute $objVal_{curr}$ with using **Sub-Algorithm 1** and current hash value;
5: Add current hash value and $objVal_{curr}$ to the hashlist, and increment $iter$;
6: **while** $iter < maxIter$ and $nonImpIter < maxNonImpIter$ **do**
7: **Tabu Search**
8: **end while**
9: **return** $objBest$.

Figure 4.3. Main Algorithm.

1: Call the Cplex model given with (4.6) to (4.11) to find opening facility capacities and assignment variables U_{ij} before attack;
2: Record the objective value from Cplex as the first component of defender's objective value;
3: Compute attacker's objective value with using **Sub-Algorithm 2** and record this as the second component of defender's objective value;
4: Add two components to obtain defender's $objVal_{curr}$;
5: **return** defender's $objVal_{curr}$ to send to **Main Algorithm**

Figure 4.4. Sub-Algorithm 1.

```
1: Tabu Search
2: Set objBest = ∞, objBest_neigh = ∞;
3: for each move type do
4:   for each feasible neighborhood solution do
5:     Compute hash value ;
6:     if hash value is in hashlist as current then
7:       bypass it and continue with the next neighboring solution;
8:     else if hash value is in hashlist as neighbor then
9:       retrieve objVal_neigh;
10:    else
11:      Compute objVal_neigh with using Sub-Algorithm 1;
12:      Record new hash value and objVal as neighboring solution;
13:    end if
14:    if objVal_neigh < objBest_neigh then
15:      Set objBest_neigh = objVal_neigh;
16:      if objBest_neigh < objBest then
17:        Set objBest = objBest_neigh and record the incumbent has improved;
18:      end if
19:    end if
20:  end for
21: end for
22: if the incumbent has improved then
23:   nonImpIter = 0;
24: else
25:   Increment nonImpIter;
26: end if
27: if nonImpIter = maxNonImpIter then
28:   break;
29: end if
30: Increment iter;
```

Figure 4.5. Tabu Search.

1: Set $popSize = 20$, $generationFactor = 50$, and $generation = 0$;
2: **for** each particle **do**
3: Initialize S_j variables randomly;
4: **end for**
5: Ensure attacker's budget utilization by the algorithm in Figure 4.1;
6: **for** each particle **do**
7: Compute initial objective value of the attacker with Cplex calls by the model given with the formulations 4.12 to 4.15 and set them as the attacker's current objective values;
8: **end for**
9: Set *best* as finding the maximum of the attacker's current objective values;
10: **while** $generation < generationFactor - 1$ **do**
11: Increment generation;
12: **for** each particles **do**
13: Compute charges as in Equation 4.2;
14: **end for**
15: **for** each particles **do**
16: Compute forces as in Equation 4.3;
17: **end for**
18: **for** each particle **do**
19: Move particles as computing $\widetilde{F}_i = \frac{F_i}{||F_i||}$ with the equation 4.4;
20: Ensure attacker's budget utilization by the algorithm in Figure 4.1;
21: **end for**
22: **for** each particles **do**
23: Compute attacker's objective function value with Cplex calls by the model given with the formulations 4.12 to 4.15;
24: **end for**
25: Set and update the attacker's best objective value as finding the maximum of the objective function values;
26: **end while**
27: **return** the attacker's best objective value to send to **Sub-Algorithm 1**

Figure 4.6. Sub-Algorithm 2

5. COMPUTATIONAL RESULTS

5.1. Problem Instance Generation

The experimental analysis of the proposed TS algorithm for TDAP-PI is conducted on 24 test instances by varying the number of potential facility sites, the number of customers, and the fixed cost of opening a facility. These instances are generated by using the data generation scheme in [18] and [32]. The number of potential facility sites (m) is between 4 and 15 while the number of customers is ten times the number of potential facility sites. The customers' coordinates are uniformly distributed over a circular area centered at the origin (0,0) with a radius of 500 units as in [32]. They are rounded to the neraest integer. The demand of customers changes between 5 and 100 in multiples of 5. Two fixed cost levels are used for opening a facility at site j: low and high. The levels of fixed costs have been decied after conducting several tests on the instances when the number of potential facility sites is 5, 10, and 15. They are presented in Tables 5.1 and 5.2 for the high fixed cost levels and low fixed cost levels, respectively. The fixed costs have been generated randomly between the upper and lower values in these tables. The random problem generation scheme is given in Table 5.3.

Table 5.1. High Fixed Cost Levels.

# Potential Facility Sites	4	5	6	7	8	9	10	11	12	13	14	15
Lower value (000s)	100	100	100	120	120	120	140	140	140	150	150	150
Upper value (000s)	120	120	120	140	140	140	160	160	160	170	170	170

Before running the algorithm, we have conducted some experiments to compare EMLA, FIPS, and MSS (Multi-start Revised Simplex Search) proposed in [50]. For this study, 72 BPFIP (bilevel partial facility interdiction problem) instances are used, which vary in the number of customers and facility configurations taken from the study in [50].

Table 5.2. Low Fixed Cost Levels.

# Potential Facility Sites	4	5	6	7	8	9	10	11	12	13	14	15
Lower value (000s)	40	40	40	50	50	50	60	60	60	70	70	70
Upper value (000s)	50	50	50	60	60	60	70	70	70	80	80	80

The algorithms are coded in C# environment of Microsoft Visual Studio 2008 using Cplex 12.1 Concert Technology. The run results are obtained on a workstation having one Intel Xeon W3690 3.46 GHz Hexa-Core processor and 24 GB DDR3 ECC memory.

Table 5.3. Random Problem Generation.

Parameters	Values
Number of facilities (m)	4,5,...,15
Number of customers (n)	40,50,...,150
Customer i's demand (a_i)	5,10,...,100
Unit shipment cost (c_d)	0.1
Outsourcing cost (c_p)	100
Unit capacity acquisition cost (h)	2500
Capacity multiplier (q)	250
Full interdiction cost of a facility j (e_j)	15000,16000,...,30000
Interdiction budget (e_{tot})	$\eta \sum_{j \in J} e_j$
η	0.2

5.2. Results

5.2.1. Comparison Results for MSS, FIPS, and EMLA

The results of EMLA and FIPS, which we have employed to obtain better results than the previous study in [32] for the middle and lower levels of our problem, are compared to MSS since our model includes the model explained in [32].

Multi-Start Revised Simplex Search (MSS) algorithm is based on the Revised Simplex Search (RSS) of Humphrey and Wilson [33]. RSS is developed for unconstrained and unbounded problems and the author declares that it is probably the most efficient modification of Nelder-Mead Simplex Search (NMSS) [32]. In [32], this modified RSS is started K times from different randomly selected points, and hence the optimum solution is the best of these K RSS runs.

5.2.1.1. Comparing MSS and EMLA. The results, which are obtained with six runs with six different random seeds for EMLA, are compared with the MSS results given in [50].

The results are presented in Tables 5.4 and 5.5 for the instances with low attacker budget, and in Tables 5.6 and 5.7 for the instances with high attacker budget. In these tables Diff columns refer to the objective values and CPU times between MSS and EMLA, which are computed using the formulae $100(\mathrm{Z}_{\mathrm{att}}^{\mathrm{EMLA}} - \mathrm{Z}_{\mathrm{att}}^{\mathrm{MSS}})/\mathrm{Z}_{\mathrm{att}}^{\mathrm{MSS}}$ and $100(\mathrm{CPU}^{\mathrm{EMLA}} - \mathrm{CPU}^{\mathrm{MSS}})/\mathrm{CPU}^{\mathrm{MSS}}$, respectively.

The results show that there is a minor increase in the average value of the best objective values found by EMLA while a significant improvement can be observed in the CPU times. Although the MSS methodology is faster than EMLA for very small instances, it requires more CPU time to solve the large sized problem instances in comparison with EMLA. The time improvement is significant for the low budget instances. There is also an improvement in the objective values found by EMLA in

60% of low budget instances and 70% of high budget instances, and in 65% of the instances on average.

Table 5.4. Comparison Results for MSS and EMLA on small sized test instances with low attacker's budget.

	MSS		EMLA		Diff(%)	
Instance	Z_{att}	CPU(s)	Z_{att}	CPU(s)	Z_{att}	CPU(s)
4-1	133,455	60	133,917	135	0.35	125
4-2	95,897	68	95,838	113	-0.06	66
4-3	95,760	68	96,028	121	0.28	78
5-1	151,537	156	150,318	186	-0.8	19
5-2	105,359	104	105,752	87	0.37	-16
5-3	122,576	290	122,991	305	0.34	5
6-1	141,597	225	143,844	192	1.59	-15
6-2	123,886	413	124,450	397	0.46	-4
6-3	132,308	306	133,092	200	0.59	-35
7-1	172,613	740	172,146	264	-0.27	-64
7-2	172,353	641	172,744	383	0.23	-40
7-3	161,550	1425	161,460	117	-0.06	-92
8-1	190,145	1968	191,728	328	0.83	-83
8-2	187,248	587	184,705	408	-1.36	-30
8-3	196,436	3216	195,301	786	-0.58	-76
9-1	220,231	3104	221,244	513	0.46	-83
9-2	191,439	5594	190,128	858	-0.68	-85
9-3	221,309	1723	225,542	825	1.91	-52
10-1	236,310	6993	233,035	882	-1.39	-87
10-2	232,791	3922	232,813	1031	0.01	-74
10-3	241,720	4385	242,903	1957	0.49	-55
Averages	167,930	1714	168,094	480	0.13	-28

5.2.1.2. Comparing FIPS and EMLA. The results for FIPS and EMLA are obtained with six runs with six different random seeds and we report them in Tables 5.8 and 5.9 for the instances with low attacker budget and Tables 5.10 and 5.11 with high attacker budget. In these tables Diff columns refer to in the objective values and CPU times between FIPS and EMLA, which are computed using the formulae $100(\mathrm{Z}_{\text{att}}^{\text{EMLA}} -$

Table 5.5. Comparison Results for MSS and EMLA on large-sized test instances with low attacker's budget.

	MSS		EMLA		Diff(%)	
Instance	Z_{att}	CPU(s)	Z_{att}	CPU(s)	Z_{att}	CPU(s)
11-1	244,826	10,920	250,619	1610	2.37	-85
11-2	242,505	8997	244,565	2463	0.85	-73
11-3	244,814	9116	246,424	2236	0.66	-75
12-1	253,023	9174	253,608	1609	0.23	-82
12-2	277,464	5601	273,862	1735	-1.30	-69
12-3	232,252	5256	234,342	997	0.90	-81
13-1	291,288	5666	289,027	1970	-0.78	-65
13-2	283,359	36,702	276,145	5397	-2.55	-85
13-3	310,922	34,866	313,907	8365	0.96	-76
14-1	288,379	17,073	298,102	19,103	3.37	12
14-2	291,356	13,583	290,424	3655	-0.32	-73
14-3	302,989	25,042	298,806	18,868	-1.38	-25
15-1	329,193	12,666	334,945	2876	1.75	-77
15-2	352,567	43,341	352,916	13,923	0.10	-68
15-3	351,240	28,357	342,920	7335	-2.37	-74
Averages	286,412	17,757	286,707	6143	0.17	-67

Table 5.6. Comparison Results for MSS and EMLA on small-sized test instances with high attacker's budget.

	MSS		EMLA		Diff(%)	
Instance	Z_{att}	CPU(s)	Z_{att}	CPU(s)	Z_{att}	CPU(s)
4-1	168,174	29	168,125	48	-0.03	66
4-2	140,902	29	141,087	66	0.13	128
4-3	146,357	33	146,357	78	0	136
5-1	213,249	67	213,249	111	0	66
5-2	167,760	48	167,623	61	-0.08	27
5-3	183,783	91	185,008	86	0.67	-5
6-1	201,362	90	200,989	79	-0.19	-12
6-2	202,636	131	202,662	178	0.01	36
6-3	205,554	105	205,883	107	0.16	2
7-1	251,692	164	254,451	128	1.10	-22
7-2	248,815	191	247,618	178	-0.48	-7
7-3	248,187	426	250,233	225	0.82	-47
8-1	286,688	235	289,201	193	0.88	-18
8-2	276,822	352	277,62	217	0.29	-38
8-3	284,171	337	286,023	196	0.65	-42
9-1	333,990	400	334,724	215	0.22	-46
9-2	296,831	540	304,872	285	2.71	-47
9-3	325,006	548	326,859	218	0.57	-60
10-1	360,567	1105	359,117	298	-0.4	-73
10-2	348,330	859	350,695	280	0.68	-67
10-3	357,653	470	358,354	286	0.2	-39
Averages	249,930	298	250,988	168	0.38	-3

Table 5.7. Comparison Results for MSS and EMLA on large-sized test instances with high attacker's budget.

	MSS		EMLA		Diff(%)	
Instance	Z_{att}	CPU(s)	Z_{att}	CPU(s)	Z_{att}	CPU(s)
11-1	380,498	999	384,409	289	1.03	-71
11-2	375,081	1272	375,626	331	0.15	-74
11-3	380,463	1109	381,844	476	0.36	-57
12-1	403,593	1629	402,481	419	-0.28	-74
12-2	418,656	890	419,456	332	0.19	-63
12-3	388,935	1061	391,904	360	0.76	-66
13-1	450,275	1535	447,510	581	-0.61	-62
13-2	443,106	6918	441,948	660	-0.26	-90
13-3	455,373	3877	457,571	992	0.48	-74
14-1	478,885	5214	482,701	1053	0.8	-80
14-2	475,530	2709	474,404	535	-0.24	-80
14-3	475,474	11,705	472,105	1271	-0.71	-89
15-1	513,205	1793	512,197	438	-0.2	-76
15-2	520,933	7733	534,211	998	2.55	-87
15-3	515,655	3153	518,366	866	0.53	-73
Averages	445,044	3440	446,449	640	0.30	-74

$Z_{att}^{FIPS})/Z_{att}^{FIPS}$ and $100(CPU^{EMLA} - CPU^{FIPS})/CPU^{FIPS}$, respectively.

The FIPS results are not satisfactory in comparison with the EMLA results in terms of both CPU time and the best objective value found. That is why we choose EMLA to implement in the middle level of TDAP-PI instead of FIPS optimization or MSS algorithm.

Table 5.8. Comparison Results for FIPS and EMLA on small-sized test instances with low attacker's budget.

	FIPS		EMLA		Diff(%)	
Instance	Z_{att}	CPU(s)	Z_{att}	CPU(s)	Z_{att}	CPU(s)
4-1	133,785	277	133,917	135	0.1	-51
4-2	95,838	195	95,838	113	0	-42
4-3	95,157	155	96,028	121	0.92	-22
5-1	149,227	299	150,318	186	0.73	-38
5-2	105,304	203	105,752	87	0.43	-57
5-3	121,367	504	122,991	305	1.34	-39
6-1	142,311	495	143,844	192	1.08	-61
6-2	123,309	778	124,450	397	0.93	-49
6-3	132,952	476	133,092	200	0.11	-58
7-1	168,667	717	172,146	264	2.06	-63
7-2	167,435	1016	172,744	383	3.17	-62
7-3	158,021	1394	161,460	117	2.18	-92
8-1	190,738	1162	191,728	328	0.52	-72
8-2	183,184	856	184,705	408	0.83	-52
8-3	192,506	1417	195,301	786	1.45	-45
9-1	219,836	2257	221,244	513	0.64	-77
9-2	187,910	2194	190,128	858	1.18	-61
9-3	221,527	1756	225,542	825	1.81	-53
10-1	229,608	2373	233,035	882	1.49	-63
10-2	229,632	2693	232,813	1031	1.39	-62
10-3	242,370	3129	242,903	1957	0.22	-37
Averages	166,223	1159	168,094	480	1.07	-55

Table 5.9. Comparison Results for FIPS and EMLA on large-sized test instances with low attacker's budget.

	FIPS		EMLA		Diff(%)	
Instance	Z_{att}	CPU(s)	Z_{att}	CPU(s)	Z_{att}	CPU(s)
11-1	247,434	3330	250,619	1610	1.29	-52
11-2	242,236	6186	244,565	2463	0.96	-60
11-3	245,194	5231	246,424	2236	0.5	-57
12-1	251,511	4456	253,608	1609	0.83	-64
12-2	270,332	3551	273,862	1735	1.31	-51
12-3	230,936	3473	234,342	997	1.47	-71
13-1	284,677	4666	289,027	1970	1.53	-58
13-2	274,093	12,277	276,145	5397	0.75	-56
13-3	306,728	13,246	313,907	8365	2.34	-37
14-1	297,759	44,952	298,102	19,103	0.12	-58
14-2	284,855	6188	290,424	3655	1.96	-41
14-3	296,393	32,600	298,806	18,868	0.81	-42
15-1	331,672	6008	334,945	2876	0.99	-52
15-2	352,518	29,359	352,916	13,923	0.11	-53
15-3	339,386	14,062	342,920	7335	1.04	-48
Averages	283,715	12639	286,707	6143	1.07	-53

Table 5.10. Comparison Results for FIPS and EMLA on small-sized test instances with high attacker's budget.

	FIPS		EMLA		Diff(%)	
Instance	Z_{att}	CPU(s)	Z_{att}	CPU(s)	Z_{att}	CPU(s)
4-1	168,299	96	168,125	48	-0.1	-50
4-2	140,852	90	141,087	66	0.17	-27
4-3	144,631	83	146,357	78	1.19	-6
5-1	210,444	119	213,249	111	1.33	-7
5-2	167,364	108	167,623	61	0.15	-44
5-3	184,115	153	185,008	86	0.49	-44
6-1	200,842	150	200,989	79	0.07	-47
6-2	201,650	211	202,662	178	0.5	-16
6-3	205,468	321	205,883	107	0.2	-67
7-1	252,942	242	254,451	128	0.6	-47
7-2	244,619	357	247,618	178	1.23	-50
7-3	244,936	383	250,233	225	2.16	-41
8-1	285,481	346	289,201	193	1.3	-44
8-2	273,967	348	277,620	217	1.33	-38
8-3	282,070	342	286,023	196	1.4	-43
9-1	331,816	472	334,724	215	0.88	-54
9-2	299,499	419	304,872	285	1.79	-32
9-3	324,353	418	326,859	218	0.77	-48
10-1	354,193	671	359,117	298	1.39	-56
10-2	345,376	572	350,695	280	1.54	-51
10-3	356,262	625	358,354	286	0.59	-54
Averages	248,532	311	250,988	168	0.9	-41

Table 5.11. Comparison Results for FIPS and EMLA on large-sized test instances with high attacker's budget.

	FIPS		EMLA		Diff(%)	
Instance	Z_{att}	CPU(s)	Z_{att}	CPU(s)	Z_{att}	CPU(s)
11-1	382,734	768	384,409	289	0.44	-62
11-2	373,552	1087	375,626	331	0.56	-70
11-3	377,128	693	381,844	476	1.25	-31
12-1	399,018	1088	402,481	419	0.87	-61
12-2	413,990	814	419,456	332	1.32	-59
12-3	388,919	748	391,904	360	0.77	-52
13-1	439,206	1306	447,510	581	1.89	-56
13-2	438,468	1622	441,948	660	0.79	-59
13-3	451,389	1795	457,571	992	1.37	-45
14-1	477,640	3331	482,701	1053	1.06	-68
14-2	468,399	1346	474,404	535	1.28	-60
14-3	465,629	2721	472,105	1271	1.39	-53
15-1	508,109	1351	512,197	438	0.8	-68
15-2	528,370	3426	534,211	998	1.11	-71
15-3	517,065	1855	518,366	866	0.25	-53
Averages	441,974	1597	446,449	640	1.01	-58

5.2.2. Results obtained for TDAP-PI Instances

In this study we test our solution algorithm for 24 test instances. The detailed results of the experiments are given in the appendix in Tables A.1 through A.4. The results are given in Table 5.12 for the case of high fixed cost level for opening a facility, and in Table 5.13 for the case of low fixed cost level.

In our solution algorithm, the number of 1-Swap moves is computed by multiplying the ratio of 1-Swap moves with the formula $p(m-p)$, which is the total number of 1-Swap moves. A decrease in the number of 1-Swap moves, which can be obtained by decreasing the ratio of 1-Swap, decreases the neighborhood size. Although smaller neighborhood size has a negative effect on the performance of the results, this strategy may also yield a better result in a shorter amount of time because it will visit more neighbors in the search area. This would be possible, if the limited area, provided by the ratio, contained the good results or the best result. Thus, solution quality will improve with the acceptable CPU times. This can be facilitate with testing more ratio values of 1-Swap moves. This kind of testing provides randomness of restricted search area. However, running with more ratio alternatives will be expensive with respect to the amount of time spent to reach desirable results. On the other hand if the area is large enough with using bigger ratio alternatives, we would not be able to reach the good quality result with the number of iterations in which we used for the restricted area. It is also not willing to happen.

The instance column in Table 5.12 contains the number of potential facilities, the number of customers, the level of the fixed cost for opening a facility, and the attacker's budget. The defender's objective value (OFV), the CPU time, and the number of opened facilities are provided for 1-Swap ratios of 1/10 and 1/1. In the table, Diff columns refer to the difference between the objective values and CPU times of the ratio of 1/10 and 1/1 computed using the formulae $100(\text{OFV}^{1}-\text{OFV}^{0.1})/\text{OFV}^{0.1}$ and $100(\text{CPU}^{1}-\text{CPU}^{0.1})/\text{CPU}^{0.1}$, respectively.

The comparison results show that the ratio of 1/1 for 1-Swap moves does not

improve the objective function value found. It is true that the better results can be found with increasing the number of iterations in the algorithm with the ratio of 1/1 but it will be unnecessarily time consuming. The CPU times for the instances of (13,130,high,60400), (14,140,high,61400), and (15,150,high,65000) in Table 5.12 are lower for the ratio of 1/1 than the ratio of 1/10. The termination criteria that is the number of successive iterations ($nonImpIter$) when the incumbent does not improve generates these cases since the best objective function value is found in the very first iterations and it does not change during the predetermined number of iterations.

Table 5.12. High Fixed Cost.

	Ratio=1/10			Ratio=1/1			Diff(%)	
Instance	OFV	CPU(s)	# of Opened Fac	OFV	CPU(s)	# of Opened Fac	OFV	CPU(s)
(4,40,high,17000)	326,156	36	1	326,156	34	1	0	-6
(5,50,high,23000)	449,483	85	1	449,483	87	1	0	2
(6,60,high,27600)	474,762	218	2	474,762	238	2	0	9
(7,70,high,35600)	671,136	541	2	671,136	525	2	0	-3
(8,80,high,35200)	669,944	1680	2	669,944	2071	2	0	23
(9,90,high,37800)	789,086	1962	2	789,086	3731	2	0	90
(10,100,high,39800)	931,524	2155	2	905,459	4111	2	-2.8	91
(11,110,high,46200)	1,004,769	6488	2	1,011,166	51,416	2	0.6	692
(12,120,high,58000)	1,151,048	3313	3	1,131,927	39,593	3	-1.7	1095
(13,130,high,60400)	1,248,756	19,880	3	1,248,609	9734	3	0	-51
(14,140,high,61400)	1,303,613	24,785	3	1,303,613	9963	3	0	-60
(15,150,high,65000)	1,427,257	47,618	3	1,427,257	8979	3	0	-81
Averages	870,628	9063	2	867,383	10,874	2	-0.3	150

5.2.2.1. Dealing with the Fixed Cost. As can be seen in Tables 5.12 and 5.13, the number of opened facilities is inversely proportional to the level of fixed cost for opening a facility. We have spent a significant amount of time on adjusting these levels of fixed cost since it directly increases the CPU time when running the algorithm with the very low fixed cost levels.

Table 5.13. Low Fixed Cost.

	Ratio=1/10			Ratio=1/1			Diff(%)	
Instance	OFV	CPU(s)	# of Opened Fac	OFV	CPU(s)	# of Opened Fac	OFV	CPU(s)
(4,40,low,17000)	246,446	35	2	246,446	36	2	0	3
(5,50,low,23000)	268,431	88	2	268,604	89	2	0.1	1
(6,60,low,27600)	345,852	211	3	345,852	234	3	0	11
(7,70,low,35600)	499,014	375	3	498,530	529	3	-0.1	41
(8,80,low,35200)	490,022	2947	3	490,522	1886	3	0.1	-36
(9,90,low,37800)	602,188	2651	3	602,468	7286	3	0	175
(10,100,low,39800)	715,158	3001	3	715,158	4373	3	0	46
(11,110,low,46200)	744,310	4366	4	743,824	36,208	4	-0.1	729
(12,120,low,58000)	850,930	5510	4	851,586	42,735	4	0.1	676
(13,130,low,60400)	923,577	46,142	4	920,170	47,096	4	-0.4	2
(14,140,low,61400)	967,838	39,580	5	967,252	44,208	5	-0.1	12
(15,150,low,65000)	1,118,007	15,497	5	1,121,089	26,095	4	0.3	68
Averages	647,648	10,034	3	647,625	17,565	3	0	144

5.2.2.2. The Effect of the Attacker Budget. The attacker budget (e_{tot}) is computed by the $\eta = 0.2$ of the full interdiction cost of the whole system for an instance. As can be seen in Table 5.14 the attacker capability percentage of being able to destroy, which is calculated by the formula $(\sum_{j \in J} S_j e_j)/e_{tot}$, is extremely high for the case of $\eta = 0.2$. Evidently, this is caused by not being opened all the potential facilities in the system. The results of other η cases in Table 5.14 prove this argument.

Table 5.14. Effect of Attacker Budget.

Instance	$\eta = 0.2$ (%)	$\eta = 0.1$ (%)	$\eta = 0.05$ (%)
(4,40,high,1/10)	73.9	37.0	18.5
(5,50,high,1/10)	82.1	41.1	20.5
(6,60,high,1/10)	48.4	47.6	23.8
(7,70,high,1/10)	62.5	59.3	37.1
(8,80,high,1/10)	69.0	58.7	29.3
(9,90,high,1/10)	87.9	44.0	37.8
(10,100,high,1/10)	94.8	51.0	43.3
(11,110,high,1/10)	94.3	53.7	46.2
(12,120,high,1/10)	75.3	52.7	26.9
(13,130,high,1/10)	75.5	43.8	30.8
(14,140,high,1/10)	71.4	53.9	33.4
(15,150,high,1/10)	95.6	70.7	35.3

5.2.2.3. Comparing Enumeration Results with TDAP-PI Algorithm . We generate all the facility configurations with the number of potential sites 4 to 11. The total number of the combinations is $2^m - 1$ with $m = 4, 5, 6, 7, 8, 9, 10, 11$. We input these facility configurations to EMLA in the middle level of TDAP-PI to find the objective value to evaluate the performance of our solution algorithm. The average CPU times obtained with the complete enumeration algorithm are 16,147 seconds and 16,277 seconds for high fixed cost and low fixed cost instances, respectively while the average CPU times found by TS are 1646 seconds and 1709 seconds for the results of high fixed cost and low fixed cost instances, respectively. The results meet our expectations since complete enumeration explores exhaustive solution space of all facility configurations

in exponential time. These CPU times represent the average times for the instances with the number of potential sites 4 to 11. Also, we do not observe any improvement in the objective values found by complete enumeration in comparison to TS.

6. CONCLUSION

In this thesis we develop a trilevel mixed integer programming formulation to solve a defender-attacker problem with the decisions of facility location, customer assignment, and interdiction. To our knowledge this type of problem has not been studied in the literature. We use metaheuristics to propose a solution algorithm and test it on a network system which is designed to provide critical service to the customers meeting their demands. We do not compare the results of our algorithm with any admitted study in the literature; hence we develop a complete enumeration algortihm to evaluate the quality of the results.

In our model there are two players: a system planner who gives the service and the attacker who tries to cause the major disruption to the system. The system planner decisions are to determine the location of the facilities, capacity of the facilities and preattack assignment of customers in the upper level and the post allocation plan in the lower level after the attacker's interdiction strategy decision in middle level. We use TS to locate the facilites and the commercial solver Cplex to set the assignment of the customers. The attacker's decision is made with using EMLA and the post allocation plan of the system planner is made with Cplex.

We compare two metaheuristics, FIPS and EMLA, to find the best interdiction strategy for the attacker in the MLP. Although FIPS has proven success with easy implementation in the literature, we encounter a premature convergence problem and we could not succeed in getting better results with even parameter adjustment. Thus, we chose to implement EMLA for solving our trilevel problem.

Based on our test results, if the neighborhood size can be efficiently reduced, better results are attainable in a smaller amount of time. Random search makes this possible as we reach the same results with the smaller amount of time.

The problem formulation has enabled us to deal with real world cases by integrating of various decisions and partial interdiction in the same problem. However, we could not compare the results by any values which are obtained with exact solution algorithms, so this problem needs to be solved with such kind of methods. Based on this motivation, the future work focuses on this promising area.

APPENDIX A: Experimental Results

Table A.1. High Fixed Cost with 1-Swap Ratio 1/1.

Instance	Attacker Budget	OFV	CPU(s)	Cplex CPU(s) in Upper L Cplex	Cplex CPU(s) in Lower L Cplex	Opened Facilities
(4,40,high,1/1)	17,000	326,156	34	0	34	1
(5,50,high,1/1)	23,000	449,483	87	0	86	1
(6,60,high,1/1)	27,600	474,762	238	9	228	2
(7,70,high,1/1)	35,600	671,136	525	9	513	2
(8,80,high,1/1)	35,200	669,944	2071	19	2049	2
(9,90,high,1/1)	37,800	789,086	3731	115	3610	2
(10,100,high,1/1)	39,800	905,459	4111	57	4047	2
(11,110,high,1/1)	46,200	1,011,166	51,416	53	51,353	2
(12,120,high,1/1)	58,000	1,131,927	39,593	50	39,534	3
(13,130,high,1/1)	60,400	1,248,609	9734	134	9585	3
(14,140,high,1/1)	61,400	1,303,613	9963	238	9708	3
(15,150,high,1/1)	65,000	1,427,257	8979	208	8754	3

Table A.2. High Fixed Cost with 1-Swap Ratio 1/10.

Instance	Attacker Budget	OFV	CPU(s)	Cplex CPU(s) in Upper L Cplex	Cplex CPU(s) in Lower L Cplex	Opened Facilities
(4,40,high,1/10)	17,000	326,156	36	0	36	1
(5,50,high,1/10)	23,000	449,483	85	1	83	1
(6,60,high,1/10)	27,600	474,762	218	8	209	2
(7,70,high,1/10)	35,600	671,136	541	7	532	2
(8,80,high,1/10)	35,200	669,944	1680	7	1670	2
(9,90,high,1/10)	37,800	789,086	1962	31	1928	2
(10,100,high,1/10)	39,800	931,524	2155	22	2129	2
(11,110,high,1/10)	46,200	1,004,769	6488	29	6453	2
(12,120,high,1/10)	58,000	1,151,048	3313	23	3283	3
(13,130,high,1/10)	60,400	1,248,756	19880	69	19803	3
(14,140,high,1/10)	61,400	1,303,613	24785	201	24572	3
(15,150,high,1/10)	65,000	1,427,257	47618	114	47494	3

Table A.3. Low Fixed Cost with 1-Swap Ratio 1/1.

Instance	Attacker Budget	OFV	CPU(s)	Cplex CPU(s) in Upper L Cplex	Cplex CPU(s) in Lower L Cplex	Opened Facilities
(4,40,low,1/1)	17,000	246,446	36	0	35	2
(5,50,low,1/1)	23,000	268,604	89	0	88	2
(6,60,low,1/1)	27,600	345,852	234	10	223	3
(7,70,low,1/1)	35,600	498,530	529	13	513	3
(8,80,low,1/1)	35,200	490,522	1886	19	1863	3
(9,90,low,1/1)	37,800	602,468	7286	153	7126	3
(10,100,low,1/1)	39,800	715,158	4373	63	4302	3
(11,110,low,1/1)	46,200	743,824	36208	103	36091	4
(12,120,low,1/1)	58,000	851,586	42735	117	42604	4
(13,130,low,1/1)	60,400	920,170	47096	593	46469	4
(14,140,low,1/1)	61,400	967,252	44208	1337	42836	5
(15,150,low,1/1)	65,000	1,121,089	26095	563	25500	4

Table A.4. Low Fixed Cost with 1-Swap Ratio 1/10.

Instance	Attacker Budget	OFV	CPU(s)	Cplex CPU(s) in Upper L Cplex	Cplex CPU(s) in Lower L Cplex	Opened Facilities
(4,40,low,1/10)	17,000	246,446	35	0	34	2
(5,50,low,1/10)	23,000	268,431	88	0	87	2
(6,60,low,1/10)	27,600	345,852	211	9	201	3
(7,70,low,1/10)	35,600	499,014	375	3	371	3
(8,80,low,1/10)	35,200	490,022	2947	15	2929	3
(9,90,low,1/10)	37,800	602,188	2651	54	2592	3
(10,100,low,1/10)	39,800	715,158	3001	31	2964	3
(11,110,low,1/10)	46,200	744,310	4366	40	4319	4
(12,120,low,1/10)	58,000	850,930	5510	52	5452	4
(13,130,low,1/10)	60,400	923,577	46142	142	45989	4
(14,140,low,1/10)	61,400	967,838	39580	399	39168	5
(15,150,low,1/10)	65,000	1,118,007	15497	378	15102	5

REFERENCES

1. Snyder, L. V. and M. S. Daskin,"Reliability Models for Facility Location: The Expected Failure Cost Case", *Transportation Science*, Vol. 39, No. 3, pp. 400-416, 2005.

2. Snyder, L. V. and M. S. Daskin, "Models for Reliable Supply Chain Network Design", in A. T. Murray and T. H. Grubesic (eds.), *Advances in Spatial Science*, Critical Infrastructure Reliability and Vulnerability, pp. 257-289, Springer, Berlin Heidelberg, 2007.

3. Wollmer, R., "Removing Arcs from a Network", *Operations Research*, Vol. 12, No.6, pp. 934-940, 1964.

4. Smith, J. C., C. Lim and F. Sudargho, "Survivable Network Design Under Optimal and Heuristic Interdiction Scenarios", *Journal of Global Optimization*, Vol. 38, pp.181-199, 2007.

5. McMusters, A. W., and T. M. Mustin," Optimal Interdiction of a Supply Network", *Naval Research Logistics Quarterly*, Vol. 17, pp. 261–68, 1970.

6. Ghare, P. M., D. C. Montgomery, and W. C. Turner, "Optimal Interdiction Policy for a Flow Network", *Naval Research Logistics Quarterly*, Vol. 18, pp. 37–45, 1971.

7. Corley, H. W., and H. Chang, "Finding the n Most Vital Nodes in a Flow Network", *Management Science*, Vol. 21, pp. 364-64, 1974.

8. Whiteman, P. S., "Improving Single Strike Effectiveness for Network Interdiction", *Military Operations Research*, Vol. 4, pp. 15–30, 1999.

9. Corley, H. W., and D. Y. Sha, "Most Vital Links and Nodes in Weighted Networks", *Operations Research Letters*, Vol. 1, pp. 157–60, 1982.

10. Ball, M. O., B. L. Golden, and R. V. Vohra, "Finding the Most Vital Arcs in a Network, *Operations Research Letters*, Vol. 8, pp. 73–76, 1989.

11. Malik, K., A. K. Mittal, and S. K. Gupta, "The k Most Vital Arcs in the Shortest Path Problem", *Operations Research Letters*, Vol. 8, pp. 223–27, 1989.

12. Israeli, E., and R. K. Wood, "Shortest-path Network Interdiction", *Networks*, Vol. 40, pp. 97–111, 2002.

13. 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.

14. 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.

15. 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.

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

17. Liberatore, F., M. P. Scaparra, and M.S. Daskin, "Analysis of Facility Protection Strategies against Uncertain Number of Attacks: The Stochastic r-Interdiction Median Problem with Fortification", *Computers and Operations Research*, Vol. 38, No. 1, pp. 357-366, 2011.

18. 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.

19. Aksen, D., and N. Aras, "A Bilevel Fixed Charge Location Model for Imminent Attack", *Computers and Operations Research*, Vol. 39, No. 7, pp. 1364-1381, 2011.

20. Losada, C., M. P. Scaparra, and J. O'Hanley, "Optimizing System Resilience: A Facility Protection Model with Recovery Time", *European Journal of Operational Research*, Vol. 217, pp. 519–30, 2012.

21. Scaparra, M. P. and R. L. Church, "Protecting Supply Systems to Mitigate Potential Disaster:A Model to Fortify Capacitated Facilities", *International Regional Science Review*, Vol. 35, No 2, pp. 188-210, 2012.

22. Vicente, Luis N., Paul H. Calamai, "Bilevel and Multilevel Programming: A Bibliography Review", *Journal of Global Optimization*, Vol. 5, No. 3, pp. 291-306, 1994.

23. Gaur, A., S. R. Arora, "Multi-Level Multi-Objective Integer Linear Programming Problem", *Advanced Modeling and Optimization*, Vol. 10, No. 2, 2008.

24. Stackelberg, H. von, *The Theory of Market Economy*, Oxford University Press, New York, 1952.

25. Colson, B., P. Marcotte, G. Savard, "An Overview of Bilevel Optimization", *Ann Oper Res*, Vol. 153, No. 1, pp. 235–256, 2007.

26. Bard, J. F.," Some Properties of the Bilevel Programming Problem", *Journal of Optimization Theory and Applications*, Vol. 68, pp. 371–378, 1991.

27. Hansen, P., B. Jaumard and G. Savard, "New Branch-and-Bound Rules for Linear Bilevel Programming", *SIAM Journal on Scientific and Statistical Computing*, Vol.13, pp. 1194–1217, 1992.

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

29. Wen, U.P., Y.H. Yang, "Algorithms for Solving the Mixed-Integer Two-Level Lin-

ear Programming Problem", *Computers and Operations Research, Vol.* 17, No. 2, pp.133-142, 1990.

30. Gümüş, 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.

31. Denegre, S. and T. Ralphs, "A Branch-and-cut Algorithm for Integer Bilevel Linear Programs", Technical report, COR@L Laboratory, Lehigh University, 2008.

32. Akca, S. Ş., "A Bilevel Partial Interdiction Problem with Capacitated Facilities and Demand Outsourcing", Master thesis, Institute for Graduate Studies in Science and Engineering, Boğaziçi University, 2011.

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

34. Liberatore F., M.P. Scaparra, M.S. Daskin, "Hedging against Disruptions with Ripple Effects in Location Analysis", *Omega*, Vol. 40, No.1, pp. 21-30, 2012.

35. Kennedy, J., and R. Mendes," Population structure and particle swarm performance", *Proceedings of the IEEE congress on evolutionary computation (CEC)*, pp. 1671–1676, Honolulu, HI. Piscataway, IEEE, 2002.

36. Birbil, Ş. İ. and S.-C. Fang, "An Electromagnetism-like Mechanism for Global Optimization", *Journal of Global Optimization*, Vol. 25, No. 3, pp. 263–282, 2003.

37. Kennedy, J., and R. C. Eberhart, "Particle swarm optimization", *Proceedings of the IEEE internationalconference on neural networks IV*, pp. 1942–1948, Piscataway: IEEE, 1995.

38. Reynolds, C. W., Flocks, "Herds and Schools: A Distributed Behavioral Model",

Computer Graphics, Vo. 21, No. 4, pp. 25-34, 1987.

39. Heppner, F. and U. Grenander, "A Stochastic Nonlinear Model for Coordinated Bird Flocks", In S. Krasner, Ed., In The Ubiquity of Chaos, pp. 233-238, 1990.

40. Poli, R., J. Kennedy, T. Blackwell, "Particle Swarm Optimization: An overview", *Swarm Intelligence Journal*, Vol.1, pp. 33-57, 2007.

41. Wang, Y. J., "Improving Particle Swarm Optimization Performance with Local Search for High-Dimensional Function Optimization", *Optimization Methods and Software*, Vol. 25, No. 5, pp. 781-795, 2010.

42. Shi, Y., and R. C. Eberhart, "A modified Particle Swarm Optimizer", *Proceedings of the IEEE international conference on evolutionary computation*, pp. 69–73, Piscataway: IEEE, 1998.

43. Clerc, M., and J. Kennedy, "The Particle Swarm—Explosion, Stability, and Convergence in a Multidimensional Complex Space", *IEEE Transaction on Evolutionary Computation*, Vol. 6, No. 1, pp. 58–73, 2002.

44. Noel, M. M., and T.C. Jannett, "Simulation of a New Hybrid Particle Swarm Optimization Algorithm", *System Symposium in Proceedings of the Thirty-Sixth Southeastern Symposium*, IEEE Press, Atlanta, GA, pp. 150–153, 2004.

45. Mendes, R., "Population Topologies and Their Influence in Particle Swarm Performance", PhD thesis, Departamento de Informatica, Escola de Engenharia, Universidade do Minho, 2004.

46. Debels, D., B. D. Reyck, R. Leus, M. Vanhoucke, "A Hybrid Scatter Search/ Electromagnetism Meta-Heuristic for Project Scheduling", European Journal of Operational Research 169 (2006) 638–653, 2004.

47. Chang, P.-C., S.-H. Chen, C.-Y. Fan, "A Hybrid Electromagnetism-Like Algorithm

for Single Machine Scheduling Problem", *Expert Systems with Applications*, Vol. 36, pp.1259-1267.

48. Naderi, B., R. Tavakkoli-Moghaddam, M. Khalili," Electromagnetism-like Mechanism and SimulatedAannealing Algorithms for Flowshop Scheduling Problems Minimizing the Total Weighted Tardiness and Makespan", *Knowledge-Based Systems*, Vol. 23, pp. 77–85, 2010.

49. Glover, F., M. Laguna, R. Marti, "Principles of Tabu Search", In: T. Gonzalez editor, *Handbook on approximation algorithms and metaheuristics*, Boca Raton: Chapman and Hall/CRC, 2007.

50. Aksen, D., S. S. Akca, N. Aras, "A bilevel Partial Interdiction Problem with Capacitated Facilities and Demand Outsourcing", *Computers and Operations Research*, 2012.