3-26-2015

A Game Theoretic Model for the Optimal Disposition of Integrated Air Defense System Assets

Chan Y. Han

A Game Theoretic Model for the Optimal Disposition of Integrated Air Defense System Assets

THESIS

MARCH 2015

Chan Y. Han, Second Lieutenant, USAF

AFIT-ENS-MS-15-M-123

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-MS-15-M-123

A GAME THEORETIC MODEL FOR THE OPTIMAL DISPOSITION OF INTEGRATED AIR DEFENSE SYSTEM ASSETS

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Chan Y. Han, B.S.

Second Lieutenant, USAF

MARCH 2015

AFIT-ENS-MS-15-M-123

A GAME THEORETIC MODEL FOR THE OPTIMAL DISPOSITION OF INTEGRATED AIR DEFENSE SYSTEM ASSETS

THESIS

Chan Y. Han, B.S.
Second Lieutenant, USAF

Committee Membership:

LTC Brian J. Lunday, PhD
Co-Advisor

Lt Col Matthew J. Robbins, PhD
Co-Advisor

AFIT-ENS-MS-15-M-123

Abstract

We examine the optimal allocation of Integrated Air Defense System (IADS) resources to protect a country's assets, formulated as a Defender-Attacker-Defender three-stage sequential, perfect information, zero-sum game between two opponents. We formulate a trilevel nonlinear integer program for this Defender-Attacker-Defender model and seek a subgame perfect Nash equilibrium, for which neither the defender nor the attacker has an incentive to deviate from their respective strategies. Such a trilevel formulation is not solvable via conventional optimization software and an exhaustive enumeration of the game tree based on the discrete set of strategies is intractable for large problem sizes. As such, we test and evaluate variants of a tree pruning algorithm and a customized heuristic, which we benchmark against an exhaustive enumeration. Our tests demonstrate that the pruning strategy is not efficient enough to scale up to a larger problem. We then demonstrate the scalability of the heuristic to show that the model can be applied to a realistic size problem.

Key words: game theory, combinatorial optimization, Double Oracle, trilevel, IADS

For the fellows of Miami Valley Philosophical and Lifting Society

Acknowledgements

I want to thank LTC Brian J. Lunday and Lt Col Matthew J. Robbins for guiding me in the right direction. I want to thank my classmates for giving me sound advice when needed, which was quite often.

Chan Y. Han

Table of Contents

Page

List of Figures

List of Tables

A GAME THEORETIC MODEL FOR
THE OPTIMAL DISPOSITION OF
INTEGRATED AIR DEFENSE SYSTEM ASSETS

I. Introduction

1.1 Motivation

Counterinsurgency has been a primary focus of U.S. military studies due to the nature of recent conflicts [1]. However, the events such as the Crimean crisis in Ukraine [2] and the Hamas-Israel conflict in Gaza Strip [3] certainly draw more attention to studies in conventional warfare. Within this research, we seek to examine an issue within the conventional warfare framework: optimal decision making pertaining to the design and operation of an Integrated Air Defense System (IADS). Herein, we consider a sequence of defender and attacker IADS-related decisions within a game theoretic framework.

Since the advent of the V-2 rockets in the World War II [4], the threat of missiles has been present in the U.S. military defensive framework. More recently, U.S. allies, such as Israel and South Korea, face significant threats from their neighboring states. Since World War II, Israel has been the largest recipient of U.S. foreign aid at $121 billion. Within this aid, the U.S. contributed $2.365 billion to the Arrow anti-missile program since 1988 [5]. In FY2014 alone, the U.S. and spent $729 million for US-Israeli Missile Defense budget [6]. Also, South Korea is experiencing increased threats from North Korea; consider, for example, its recent display of nuclear missile testing and Unmanned Aerial Vehicle patrols [7].

We motivate our problem by using Israel's Iron Dome air defense system as an example. Iron Dome is a short-range anti-rocket system designed to selectively intercept short-range rocket threats that are in-flight and targeting population centers. The Iron Dome system consists of a portable missile firing unit, detection radar, and battle management center [5]. This system was designed by Israel to counter attacks from either the Gaza Strip or Syria. The question for Israel, then, is where should they place Iron Dome batteries? Assuming that the main objective of an IADS is to protect valuable assets (e.g., population or infrastructure), the set of decisions involves the location of batteries to protect the most valuable targets. This logic is applicable and will yield the best coverage based on the defender's valuation of targets. However, if we assume that Hamas, the attacker, knows the layout of Israel's defense, they may attack smaller, unprotected targets that an IADS system with limited resources cannot protect, or they may attack the most valuable targets that are protected by the IADS if there is a sufficient likelihood of penetrating the air defense.

The same argument applies for Hamas. If Hamas seeks to maximize the damage to Israel without consideration of Israel's defensive resources, they must attack as many valuable targets as possible. However, with Israel's air defense system there is a probability that a rocket attack is ineffective, which increases with improvements to Israel's defense.

Herein, we consider a game theoretic study of resource allocation in a strategic environment, wherein each of two decision makers must allocate resources, and the other's strategies affect their own payoffs. In this particular example, the locations at which Israel places its Iron Dome assets affect Hamas' strategies and payoffs, and vice versa. In addition, we apply the concept of Nash equilibrium: a set of respective strategies from which neither player has an incentive to deviate.

Even though Iron Dome was used as an example, it is only one component of

Israel's IADS. Other components include anti-rocket systems such as David's Sling, as well as Arrow I, II, and III intercept missiles, each of which has different flight characteristics. For the purpose of this initial research on this strategic problem, we generalize the defensive batteries as Surface-to-Air Missile (SAM) batteries that carry a certain number of interceptor missiles (IM), and we generalize the attacker's assets as attacker missiles (AM).

1.2 Problem Statement

Given a set of cities N, each with a value $v_j, j \in N$, a limited number of SAM batteries, and a fixed number of IMs for each SAM site for the defender, and a fixed number of AMs for the attacker, what are the best respective defender and attacker strategies to locate and/or allocate their assets? We model this situation as a two-player, sequential, three-stage, perfect information, zero-sum game. The term *zero-sum* indicates that the attacker's gain in target destruction will equal the defender's loss. We model this situation as a sequential game wherein the defender initially sets up its SAM sites, then the attacker targets certain cities by firing a single salvo of AMs, upon which the defender launches interceptors against the incoming AMs. We refer to this framework as a defender-attacker-defender or D-A-D model.

1.3 Assumptions

For the purpose of this research, we make the following initial assumptions:

General Assumptions.

1) All parameters are common knowledge. That is, both players know how many SAM batteries and IMs the defender has and how many AMs the attacker has. Given the current state of information availability and intelligence capability, this

is a reasonable assumption to make. This assumption makes the game a complete information game.

2) After each stage, both players know exactly what happened in the previous stage. This assumption makes the game a perfect information game.

3) Targets are valued equally by both the defender and the attacker. This is a necessary assumption for a zero-sum game model.

4) An unintercepted AM will destroy a city with 100% probability. This is a simplifying assumption.

Defender Assumptions.

1) A SAM can only be placed at a city. We make this assumption to limit the strategy space for the defender.

2) At most one SAM battery can be located at a given site. This is a simplifying assumption.

3) No more than one interceptor will be launched against each incoming AM. This assumption comes from the fact that the defender will not have time to launch a second salvo if the IM fails to destroy the AM. Since we are considering missiles that are subject to be intercepted within a limited radius of coverage, the flight characteristics of the attacker missiles we are dealing with have a short flight time.

4) Each of the SAM batteries has the same number of interceptors. This is a simplifying assumption.

Attacker Assumptions.

1) An attacker will only launch AMs.

2) All AMs have identical destructive capabilities and flight performance. We also assume that all AM launched can be detected by the defender.

3) An attacker will fire all of its AMs in a single salvo.

1.4 Organization

The remainder of this paper is organized as follows. Chapter 2 describes the current framework of U.S. Integrated Air Defense Systems (IADS) according to Joint Publication 3-01, *Countering Air and Missile Threats* [8] and reviews relevant research in the field that influences our examination of the problem. Chapter 3 presents the methodologies we examine herein to solve the problem. Chapter 4 presents the results and analysis of our testing on representative instances of varying sizes and complexities. Chapter 5 summarizes the contributions of this research and proposes directions for further studies.

II. Literature Review

This chapter provides a review of relevant doctrine, studies, and models from the literature. First, we outline the definition of an IADS and its components according to U.S. military doctrine. Then, we examine game theory and related solution concepts. We also review research concerning security games, the r-Interdiction Median problem, bilevel formulations, and infrastructures.

2.1 Background

According to Joint Publication 3-01 (JP3-01), *Countering Air and Missile Threats*, an Integrated Air Defense System (IADS) is defined as "the aggregate of Service/-functional component air and missile defense (AMD) systems comprising sensors, weapons, C2, communications, intelligence systems, and personnel..." that follow the principles of centralized planning, decentralized execution, planned responses, effective and efficient communications, layered defense, 360-degree coverage, identification and tracking, alert and warning, and establishing modes of control [8]. Optimization of an IADS configuration satisfies the planned responses component of the principles wherein the defender places SAM sites before the attacks occur. JP3-01 also mentions that while the flight profiles of most ballistic missiles are very predictable, unpredictable targets require 360-degree coverage, which justifies the generalization of attacker's assets as AMs. Also, the publication mentions many different attacker AM launch platforms (e.g., fighters, bombers, and UAVs) against which the defender employs its tactics, but we simplify the model to consider a defender having identical SAM batteries with identical interceptor missiles (IM). Another important distinction is that there are active AMD and passive AMD measures. Active AMD refers to the use of aircraft, weapons, and sensors to nullify attacks, whereas passive AMD refers to

the use of detection, warning, or concealment to increase survivability [8]. The focus of this research is on active AMD; we seek to maximize the defender's effectiveness of intercepting attacks given that the attacker seeks to inflict the maximum damage.

2.2 Game Theory

Game theory is a study of resource allocation in a strategic environment, where strategic environment implies that your payoff is influenced by others' action. With a game theoretic model, we seek to find a reasonable prediction on the actions of the defender and attacker. The concept of a Nash equilibrium in a strategic game, developed by John Nash in 1950, identifies that every finite game has an equilibrium for which no player has an incentive to deviate [9]. Selten [10] proved that any sequential game tree can be broken into sub-games and, as a result, will also have a sub-game perfect Nash equilibrium (SPNE). For a moderately-sized game-tree, a SPNE can be found via backward induction.

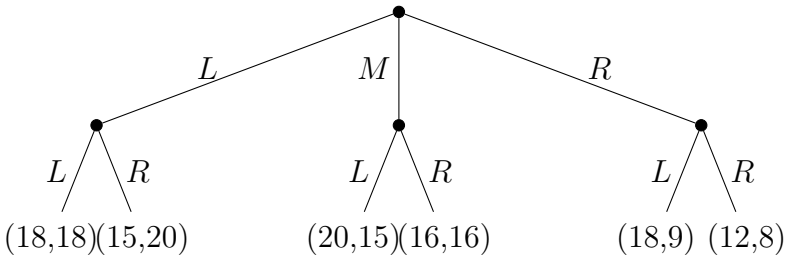


Figure 1. Small Example Game

For this example of a two-player, two-stage non-zero sum game tree (motivated from [11]) depicted in Figure 5, we can find the SPNE by examining each sub-game. The pair (x, y) represents the payoff for each terminal node of the game tree, wherein x is the payoff for the first mover and y is the payoff for the second mover. By inspection we observe that the equilibria for each sub-game, from left to right in Figure 5, is respectively R, R, and L because the second player will seek to achieve

the higher payoff in each case. Backward induction uses the fact that the second player will follow such a strategy and, with that information, the first mover will choose R. Thus, the strategy profile (R, L) is the SPNE.

Backward induction is feasible when the size of the game tree is scalable. Notice that there are $(3)(2) = 6$ terminal nodes in the tree displayed in Figure 5, where 3 is size of the first mover's strategy space, and 2 is size of the second mover's strategy space. For our model, there exist $\binom{|N|}{m}$ ways that the defender can place its SAM batteries, where $|N|$ is the number of cities and m is the number of SAM batteries available, and there exists $\binom{|N|+n-1}{n}$ ways that the attacker can allocate AMs against cities, where n is the number of total AMs. For example, with an instance having 50 cities, 5 SAM batteries, and 5 AMs, the defender has $\binom{50}{5} = 2,118,760$ ways to allocate its SAM sites, the attacker has $\binom{54}{5} = 3,162,510$ ways to allocate its AMs, and the resulting number of nodes at the second level exceeds 6.7×10^{12}. Because of the combinatorial expansion of the strategy spaces, we cannot possibly enumerate all terminal nodes within our IADS problem to apply backward induction. We discuss this further in Chapter 4.

2.3 Combinatorial Game Theory

Combinatorial game theory is a subset of game theory that typically deals with two-player perfect-information games that have a finite number of moves [12]. Some examples of *combinatorial games* are chess, go, checkers, and tic-tac-toe. In combinatorial game theory, each game's complexity is measured based on average length of the game and how many decisions each player can make at each stage. In chess, Claude Shannon [13] estimated that the number of possible moves that each player can make is in the order of 10^{43}. Such number is defined as the state-complexity in combinatorial games, and in this model, we can determine the exact number of

moves of the defender and attacker at each stage based on the number of resources each player has. One difference from combinatorial games and our model is that the outcome is not simply win, tie, or loss.

2.4 Tree Search

A substantial number of research endeavors have focused on finding a solution to a large game-tree. Specifically within the field of artificial intelligence, finding solutions for computationally intractable games such as chess have been of interest [14]. At each move in a game of chess, there is a discrete number of moves that a player can make, and the game tree expands for each move considered. Chess or any two-player board game is very similar in formulation to our problem in that they are two-player, zero-sum, perfect information, sequential games. In comparison, the defender-attacker-defender model is similar to a game of chess that ends in total of three moves.

One method to speed up the search for a SPNE in a large game tree is $\alpha - \beta$ pruning, which is similar to the branch-and-bound technique in integer programming. Knuth et al. [15] outline a technique to speed up the process of exploring a large game-tree without loss of information. Pearl [16] subsequently proved the optimality of this solution technique. We use $\alpha - \beta$ pruning as one of the methodologies to improve the computation time in finding the SPNE.

2.5 Double Oracle

Another method to address large strategy spaces is the Double Oracle algorithm, which follows the notion of column and constraint generation in linear programming as the Double Oracle algorithm solves a subproblem and adds attacker and defender strategies one by one. McMahan et al. [17] develop and prove convergence and optimality of the Double Oracle algorithm in a two-player, two-stage perfect information

game. Because of the similarity in formulations, we predominantly reference Jain et al. [18] for their Double Oracle implementation. Jain et al. [18] use the algorithm to solve a large-scale combinatorial optimization security game that is further discussed in the next section.

2.6 Security Games

Security games are used to inform resource allocation decisions concerning infrastructure protection and are an important class of Stackelberg games [19]. A Stackelberg game is a class of sequential game wherein one player optimizes his strategy knowing that the opponent will observe the decision and subsequently select his optimal strategy. One example of a security game is airport security where there is a number of vulnerable sites within an airport (i.e., the protected asset), a limited number of security personnel (i.e., the defender's resource), and a potential attacker. Such a framework aligns very closely to our model. Jain et al. [18] use an algorithm called RUGGED, a double-oracle based algorithm, to solve their model of network security games. Security games are similar to our D-A-D model because it involves a defender and an attacker, and they are sequential games by nature. Other similar examples of Stackelberg games are *hider-seeker games* [20] and *network games* [21].

2.7 Maximum Expected Location Covering

Daskin [22] introduces the Maximum Expected Location Covering model (MEXCLP), which models a network in which demand nodes must be served by a limited number of facilities that are co-located with a subset of the demand nodes. Depending on the distances involved, a facility may be able to serve demand at nearby nodes. For example, placing fire stations in a county with multiples towns of different populations can be modeled using MEXCLP. Using integer programming, MEXCLP

maximizes the expected value of demand covered in the network.

The modeling assumptions made in Daskin's research are similar to our assumptions. One of his primary assumptions is binary coverage of a demand by a facility within a fixed radius; if a demand is within a certain critical distance, the facility covers the demand. In our model, this assumption is equivalent to stating that if a city is within the SAM's defensive envelope, we can consider the city covered by that SAM battery. That is, the SAM battery may engage incoming AMs targeting the covered city. Also, MEXCLP utilizes a probability of facility failure, thus the word *expected*. The probability of facility failure captures the uncertainty with which the facility is able to serve demand. In our model, that probability is characterized in the third level as the single intercept probability of a AM by an IM. Another modeling assumption is that redundancy in coverage increase the survivability of a covered node. In our model's context, redundant coverage does not mean we launch more interceptors. Rather, more interceptors would be available to protect the city in case more AMs are launched against that city. One assumption that is not applicable from the MEXCLP model is that each facility has an inherent and identical probability of failure, but we utilize a similar parameter: the probability of failure for an IM launched by a SAM to intercept a single incoming AM.

2.8 Weapon Target Assignment Problem

Our attacker's AM allocation problem is motivated by the Weapon Target Assignment problem (WTAP). The WTAP seeks to assign some discrete number of weapons to targets so as to minimize the total expected survival value of the targets after all engagements [23]. WTAP can be formulated as a nonlinear integer program in which survivability is minimized given some probability of target hit, number of weapons, and value of each target. Ahuja et al. [23] use a very large-scale neigh-

borhood (VLSN) search algorithm to solve the WTAP. The WTAP could be applied to the second-stage of our problem, wherein an attacker seeks to allocate its AMs in order to minimize the expected survival value of the cities. Ahuja et al. [23] also provide heuristics which solve the WTAP quickly, albeit sub-optimally, since the weapon target assignment problem may require quick solutions in its application. For example, the paper presents one example instance in which there are 200 weapons and 400 targets and which the heuristic solves in 1.953 seconds.

2.9 Lexicographic Ordering

In order to generate attacker's strategies, we use a graded lexicographic ordering index to generate either a single strategy or the entire strategy space. Attacker's strategies correspond to *multiset* or *bucketspace* in combinatorics where we can put b distinguishable balls into B distinguishable buckets. Using *grlex* algorithms from *Equilibrium Blog* [24], we can generate all possible attacker strategies given a number of cities ($|N|$) and a number of AMs (n). For example, if we wish to generate the

attacker strategy space for $n = 5$ and $|N| = 6$, we would have the following output:

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 5 \\
0 & 0 & 0 & 0 & 1 & 4 \\
0 & 0 & 0 & 0 & 2 & 3 \\
0 & 0 & 0 & 0 & 3 & 2 \\
0 & 0 & 0 & 0 & 4 & 1 \\
0 & 0 & 0 & 0 & 5 & 0 \\
0 & 0 & 0 & 1 & 0 & 4 \\
0 & 0 & 0 & 1 & 1 & 3 \\
 & & \vdots & & & \\
4 & 0 & 0 & 1 & 0 & 0 \\
4 & 0 & 1 & 0 & 0 & 0 \\
4 & 1 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

Each row represents a unique attacker strategy, each column corresponds to cities, and the number in each column corresponds to the number of AMs targeting that city.

2.10 r-Interdiction

The r-Interdiction Median Problem with Fortification (RIMF) by Church and Scaparra [25] models a problem that maximizes the defender's covered demand subject to an attacker's subsequent intentional interdiction of r facilities. This formulation is related to our research as it is a Stackelberg game. The first player is a defender who fortifies q facilities. The second player is an attacker who destroys r facilities that are not one of the q facilities. The resulting formulation is an integer-linear program.

Key assumptions in this paper are that a fortified facility will not be attacked and that the attacker will succeed in attacks that they mount against any non-fortified facilities. The authors use properties of the Condensed Balinski Constraints with the Reduction of Assignment Variables (COBRA) formulation to reduce the number of variables [25]. In one of the instances addressed utilizing COBRA, the number of decision variables and constraints were reduced from 11,482 and 18,530 to 923 and 1,568, respectively. The authors also propose heuristics and provide computational results for their performance.

2.11 Infrastructure

Two infrastructure related papers by Brown et al. [26] and Hausken [27] utilize bilevel formulation in an attacker-defender setting. These papers offer insights concerning the defense of infrastructures. Brown et al. [26] note that high-fidelity models are achievable as it is possible to gather data and create models of a complex system. Also, they found that in general, an attacker has the advantage since a defender must protect a huge, dispersed target set, whereas the attacker only focuses on a small set of selected target. Hausken [27] addresses the defense of infrastructures against multiple attackers. Three ways to value a target are proposed: monetary value, human value, and symbolic value. Although those could be exchanged in some cases, a target usually possesses at least one distinct value out of the three mentioned. The author also notes that the defender and the attacker generally have different values for the same target, which is not the case for our zero-sum game formulation.

III. Problem Description

3.1 Model Description

Consider a conflict situation in which a defender seeks to protect a set of cities, N, and wherein where each city $j \in N$ has a value $v_j \in \mathbb{R}^+$. An attacker seeks to destroy cities so as to maximize damage, which is equivalent to minimizing the total value of the defender's surviving cities. We model the situation as a perfect information, three-stage sequential, zero-sum game with two players, a defender and an attacker, with only pure-strategies available for both players.

In the first stage of the game, the defender places m SAM batteries amongst the set of feasible SAM sites, $F \subseteq N$. Available SAM sites are co-located with a subset of the cities. We denote this defender strategy as $\boldsymbol{d} = (d_i)_{i \in F}$, where $d_i \in \{0, 1\}$ and require $\sum_{i \in F} d_i = m$. Also, based on a critical distance $r \in \mathbb{R}^+$, we generate a coverage matrix $A = [a_{ij}]_{|F| \times |N|}$ to indicate whether a SAM battery located at city i can cover city j, based on the Euclidean distance $\delta_{ij} > 0$ between city i and j, where

$$a_{ij} = \begin{cases} 1, & \text{if } \delta_{ij} \leq r, \\ 0, & \text{otherwise.} \end{cases}$$

In our formulation, a covered city does not denote protection or fortification against attacker missiles but merely the ability to do so depending on interception allocations. A city j is considered covered if a SAM gets placed at city i (i.e., $d_i = 1$, and city j is within the range to launch IMs against AMs coming to city j (i.e., $a_{ij} = 1$), regardless of the decision made.

In the second stage of the game, the attacker observes the defender's allocation $\boldsymbol{d}$ and launches all of its n AMs in a single salvo, implying that no further AMs will be launched. The attacker's strategies correspond to AM allocations to each city $j \in N$.

We denote the attacker's allocation strategy as $\boldsymbol{w} = (w_j)_{j \in N}$, where $w_j \in \mathbb{Z}^+$, and require $\sum_{j \in N} w_j = n$.

In the third stage of the game, the defender observes the attacker's AM allocation $\boldsymbol{w}$ and decides which AMs to intercept with available IMs. Each IM, if allocated, destroys the targeted AM with a probability $p \in (0, 1)$, and we assume that no more than one interceptor is launched against each AM. We denote the third stage defender allocation by $\boldsymbol{x} = (x_{ij})_{i \in F, j \in N}$ and $\boldsymbol{y} = (y_j)_{j \in N}$, where x_{ij} indicates the number of interceptors launched from SAM site $i \in F$ against AMs targeting city $j \in N$; we require that $x_{ij} \in \mathbb{Z}^+$ and $\sum_{j \in N} x_{ij} \leq c, \ \forall i \in F$, where $c \in \mathbb{Z}^+$ is the number of IMs that each SAM battery may fire. The binary decision variable y_j indicates whether the city $j \in N$ is protected. In order to protect city j, there must be at least as many IMs launched to protect it ($\sum_{i \in F} x_{ij}$), as the AMs attacking it (w_j). Moreover, the defender will not waste any interceptors. Together, this gives the equality constraint $\sum_{i \in F} x_{ij} = w_j$ if a city is protected. The number of possible strategies in the third stage is $2^{|N|}$; each city is considered either protected or unprotected ($y_j \in \{0, 1\}, \forall j \in N$), but some of the strategies are infeasible because defender cannot protect a city that is outside of SAM coverage.

The game proceeds in the following manner. The defender allocates its SAM batteries ($\boldsymbol{d}$), the attacker deploys AMs ($\boldsymbol{w}$), and the defender deploys interceptors against the incoming AMs to decide which cities to defend ($\boldsymbol{x}, \boldsymbol{y}$). The objective is to find the pure-strategy Nash equilibrium of this three-stage defender-attacker-defender (D-A-D) game.

Table 1. Parameters

N	Set of cities (targets) indexed by j
F	Set of possible SAM sites indexed by i, $F \subseteq N$
A	$\lvert F\rvert$ by $\lvert N\rvert$ coverage matrix, $a_{ij} \in \{0,1\}$
v_j	Value of city $j \in N$
m	Number of SAM batteries available
n	Number of AMs available
c	Number of IMs available per SAM battery
p	Probability of single IM destroying single AM
r	Maximum distance of coverage by each SAM battery

Table 2. Decision Variables

d_i	Defender's SAM allocation at city $i \in F$, $d_i \in \{0,1\}$
w_j	Attacker's AM allocation at city $j \in N$, $w_j \in \mathbb{Z}^+$
x_{ij}	Defender's IM allocation from SAM site located at city i to city j, $x_{ij} \in \mathbb{Z}^+$
y_j	Defender's decision variable to protect city $j \in N$, $y_j \in \{0,1\}$

$$\max_{\boldsymbol{d}} \quad \min_{\boldsymbol{w}} \quad \max_{\boldsymbol{x},\boldsymbol{y}} \quad \sum_{j\in N} v_j y_j p^{w_j} \tag{1a}$$

$$\text{subject to} \quad \sum_{i\in F} d_i = m, \tag{1b}$$

$$\sum_{j\in N} w_j = n, \tag{1c}$$

$$\sum_{j\in N} x_{ij} \le c d_i,\ \forall i \in F, \tag{1d}$$

$$\sum_{i\in F} a_{ij} x_{ij} \le w_j,\ \forall j \in N, \tag{1e}$$

$$\sum_{i\in F} a_{ij} x_{ij} \ge w_j y_j,\ \forall j \in N, \tag{1f}$$

$$d_i \in \{0,1\},\ \forall i \in F, \tag{1g}$$

$$w_j \in \mathbb{Z}^+,\ \forall j \in N, \tag{1h}$$

$$x_{ij} \in \mathbb{Z}^+,\ \forall i \in F, \quad j \in N, \tag{1i}$$

$$y_j \in \{0,1\},\ \forall j \in N. \tag{1j}$$

The trilevel nonlinear integer program is now presented. Equation (1a) is the objective function. The defender seeks to maximize the total expected survival value of its cities while the attacker seeks to minimize the total expected survival value. Equations (1b), (1c), and (1d) set the upper bounds on the number of SAM batteries, AMs, and IMs, respectively. Equation (1e) enforces the assumption that the defender will not launch more than one IM per incoming AM for each city. Equation (1f) ensures the defender sends enough IMs to intercept incoming AMs. Equations (1g), (1h), (1i), and (1j) enforce the binary and integrality constraints of the decision variables.

Based on our assumptions, each city $j \in N$ faces three different possibilities:

1) The city has no incoming AMs (i.e., $w_j = 0$). With no incoming AMs, equation (1f) enforces $y_j = 1$, and the city is considered protected (i.e., $y_j = 1$).

2) The city has incoming AMs and there are not enough IMs to intercept them all (i.e., $w_j > 0$ and $\sum_{i \in F} x_{ij} < w_j$). In this case, at least one AM reaches the city unintercepted and destroys the city with 100% probability as indicated in General Assumption 2. Equation (1f) forces $y_j = 0$.

3) The city has incoming AMs and there is an IM launched against each AM (i.e., $w_j > 0$ and $\sum_{i \in F} x_{ij} \geq w_j$). In this case, all incoming AMs to the city are matched up with an IM and the city is considered protected (i.e., $y_j = 1$).

The possible defender payoffs for each city j based on the three scenario are shown in Table 3.

Table 3. Possible Scenarios

Scenario	y_j	Defender Payoff
1	1	v_j
2	0	0
3	1	$v_j p^{w_j}$

3.2 Solution Methodology

The following approaches for solving the trilevel program are discussed and successively applied during testing in Chapter 4: exhaustive enumeration, $\alpha-\beta$ pruning, and Double Oracle.

Method 1: Exhaustive Enumeration

A first approach to solve the trilevel program is to enumerate all possible strategies in the first two levels and then utilize a commercial solver to optimize the third-stage integer program and obtain single-stage payoffs for the last level.

In order to index all possible strategy sets, we create index sets that correspond to the sets of all defender and attacker strategy sets. We denote a defender's pure strategy at the first stage as $s_d \in S_d$, where s_d^i is the ith strategy in a lexicographically ordered set of all possible first-stage defender strategies, S_d. Let τ_d denote the total number of possible first-stage defender strategies and define $S_d = \{s_d^1, \ldots, s_d^i, \ldots, s_d^{\tau_d}\}$. Recall that $s_d \mapsto \boldsymbol{d}$ and that $|S_d| = \tau_d = \binom{|F|}{m}$.

We denote an attacker's pure strategy at second stage as $s_a \in S_a$, where s_a^j is the jth strategy in a lexicographically ordered set of all possible attacker strategies, S_a. $S_a = \{s_a^1, \ldots, s_a^j, \ldots, s_a^{\tau_a}\}$. Recall that $s_a \mapsto \boldsymbol{w}$ and that $|S_a| = \tau_a = \binom{|N|+n-1}{n}$.

Such an index set allows us to model this game within the construct of a game tree wherein all possible combinations of the defender and attacker pure strategies in the first two stage are represented as $S \equiv S_d \times S_a$, and the number of possible defender-attacker combination can be represented as $|S| = \tau_d\tau_a = \binom{|F|}{m}\binom{|N|+n-1}{n}$.

For a particular strategy profile represented by the tuple (s_d, s_a), we solve the resulting integer program as presented in (2a) - (2f) to obtain a single payoff for the defender, which we denote as $u(s_d, s_a)$, for which $s_d \mapsto \boldsymbol{d}$ and $s_a \mapsto \boldsymbol{w}$.

$$\textbf{Payoff:} \underset{\boldsymbol{x},\boldsymbol{y}}{\text{maximize}} \quad u(\boldsymbol{d},\boldsymbol{w}) = \sum_{j\in N} v_j y_j p^{w_j} \tag{2a}$$

$$\text{subject to} \quad \sum_{j\in N} x_{ij} \leq c d_i,\ \forall i \in F, \tag{2b}$$

$$\sum_{i\in F} a_{ij} x_{ij} \geq w_j y_j,\ \forall j \in N, \tag{2c}$$

$$\sum_{i\in F} a_{ij} x_{ij} \leq w_j,\ \forall j \in N, \tag{2d}$$

$$y_j \in \{0,1\},\ \forall j \in N, \tag{2e}$$

$$x_{ij} \in \mathbb{Z}^+, \forall i \in F, j \in N. \tag{2f}$$

Exhaustive enumeration is only tractable for small values of $|N|$, $|F|$, and n, since τ_d and τ_a increase combinatorially in size with respect to increases in these parameters

Method 2: $\alpha - \beta$ Pruning

For a large game tree, there are more efficient ways to search for the maximin payoff, which corresponds to the SPNE in our formulation. Using $\alpha - \beta$ pruning, within an iterative exploration of the game tree, we fathom branches of nodes that will not yield a better solution than the incumbent solution of best response player strategies. In our implementation, α represents the lowest value that the maximizer (defender) may obtain, and β represents the highest value that the minimizer (attacker) may obtain. We set the initial values as $\alpha = 0$ and $\beta = \sum_{j\in N} v_j$, since the worst case scenario for the defender is when no city is protected (objective value= 0), and the worst case scenario for the attacker is when all cities are protected and no AMs are launched (objective value = $\sum_{j\in N} v_j$).

To investigate the impact of search order on computation time, we execute the $\alpha - \beta$ pruning algorithm four times, each of which considers a different combina-

tion of strategy search ordering at the respective defender and attacker stages of the game tree. In other words, we search the game tree S in two different orders at both stages. For example, we search the game tree S in the following manner: $(s_d^1, s_a^1), (s_d^1, s_a^2), \ldots, (s_d^1, s_a^{\tau_a}), (s_d^2, s_a^1), \ldots, (s_d^{\tau_d}, s_a^{\tau_a})$, where each terminal node is represented by the tuple (s_d, s_a). This particular method is denoted as $\alpha\beta 11$, since we begin the search at s_d^1, s_a^1.

Because the cities' values can be ordered and the strategies are ordered lexicographically, the order in which we search the tree may impact algorithm computation time. The earlier we find the best value for the attacker (i.e., $\hat{\beta} \leq \hat{\alpha}$), the more branches we prune off in the tree search, saving computation time. Thus, it is in our interest to order the strategies that yields such a result.

The pseudocode for the defender-attacker $\alpha - \beta$ pruning algorithm is shown in Algorithm 1. Shown at the end of the algorithm, $\hat{s}_d$ and $\hat{s}_a$ correspond to the SPNE defender and attacker strategy. Because we know that the $\alpha - \beta$ pruning algorithm returns the SPNE of the game tree, we can conclude that $\hat{\alpha} = \alpha^*$, $\hat{s}_d = s_d^*$, and $\hat{s}_a = s_a^*$. Note that $\hat{\alpha} = u(\hat{s}_d, \hat{s}_a)$ corresponds to the SPNE payoff for the defender. The attacker's SPNE payoff is simply $-u(\hat{s}_d, \hat{s}_a)$ as this is a zero-sum game.

Method 3: Double Oracle

In our implementation of Double Oracle, developed by McMahan et al. [17], we solve a trilevel optimization problem using three interrelated algorithmic procedures: coreTree, Defender Oracle, and Attacker Oracle.

The first component is coreTree, wherein we solve the restricted game. In the restricted game, we consider only the set of defender and attacker strategies, S_d^k and S_a^k, currently identified at step k using backward induction. The resulting strategy profile $(\hat{s}_d, \hat{s}_a)$ is a Nash equilibrium for this restrict game, where $\hat{s}_d \in S_d^k$, $\hat{s}_a \in S_a^k$, and $S^k = S_d^k \times S_a^k$. The second component is the Defender Oracle sub-problem,

Algorithm 1 $\alpha - \beta$ pruning for D-A-D

$\hat{\alpha} := 0$
for $i = 1$ to τ_d **do**
  $\hat{\beta} := \sum_{j \in N} v_j$
  for $j = 1$ to τ_a **do**
    $\hat{\beta} \leftarrow min(\hat{\beta}, u(s_d^i, s_a^j))$
    if $\hat{\beta} \leq \hat{\alpha}$ **then**
      $\hat{s}_a \leftarrow s_a^j$
      break
    end
  end
  $\hat{\alpha} \leftarrow max(\hat{\alpha}, \hat{\beta})$
  if $\hat{\beta} \geq \hat{\alpha}$ **then**
    $\hat{s}_d \leftarrow s_d^i$
  end
end
return $\hat{\alpha}, \hat{s}_d, \hat{s}_a$

wherein we identify the defender's best response ($\bar{s}_d \in S_d$) given an attacker's current best response ($\hat{s}_a$) from the coreTree. The third component is the Attacker Oracle sub-problem, wherein we solve for the attacker's best response ($\bar{s}_a \in S_a$) given a defender's current best response ($\hat{s}_d$) from the coreTree. Because of the nature of our methodology, the Double Oracle procedure is a heuristic. The difference between the Double Oracle implementation by Jain et al. [18] and this research is that we do not generate mixed strategies from the core problem. Rather, we identify the pure-strategy SPNE of the smaller game using backward induction within the trees generated from the current strategy set. The pseudocode for a defender-attacker Double Oracle is shown in Algorithm 2.

Defender Oracle Problem.

Given an attacker's AM allocation, the Defender Oracle solves for the best allocation of SAM batteries and IM launches. For the defender, s_d^i is a best response to the attacker strategy s_a if $u(s_d^i, s_a) \geq u(s_d^{i'}, s_a)$ for all $s_d^{i'} \in S_d$. It is possible to have

Algorithm 2 Double Oracle

Initialize S_d^0 by generating an arbitrary set of defender strategies.
Initialize S_a^0 by generating an arbitrary set of attacker strategies.
$k := 0$
repeat
    $k \leftarrow k + 1$
    $(\hat{s}_d, \hat{s}_a) \leftarrow \text{coreTree}(S_d^{k-1}, S_a^{k-1})$
    $\bar{s}_d \leftarrow \text{DefenderO } (\hat{s}_a)$
    $S_d^k \leftarrow S_d^{k-1} \cup \bar{s}_d$
    $\bar{s}_a \leftarrow \text{AttackerO } (\hat{s}_d)$
    $S_a^k \leftarrow S_a^{k-1} \cup \bar{s}_a$
until $S_d^k = S_d^{k-1}$ and $S_a^k = S_a^{k-1}$
return $\hat{s}_d, \hat{s}_a$

multiple best responses, but we only consider pure strategies. Thus, we choose the first strategy identified. The Defender Oracle sub-problem can be modeled as a linear integer program. We can draw the parallel to MEXCLP from this problem, as Defender Oracle problem seeks to allocate facilities to maximize their expected coverage value. However, probability of failure is not characterized by the facility. Rather, it is a function of AM allocations by the attacker. Because this is a linear integer program and we can find find the exact solution, we have that $u(\bar{s}_d, \hat{s}_a) \geq u(s_d^i, \hat{s}_a), \quad \forall s_d^i \in S_d$ holds.

$$\textbf{DefenderO:} \underset{\boldsymbol{d},\boldsymbol{x},\boldsymbol{y}}{\text{maximize}} \quad \sum_{j \in N} v_j y_j p^{w_j} \tag{3a}$$

$$\text{subject to} \quad \sum_{i \in F} d_i = m, \tag{3b}$$

$$\sum_{j \in N} x_{ij} \leq c d_i, \ \forall i \in F, \tag{3c}$$

$$\sum_{i \in F} a_{ij} x_{ij} \geq w_j y_j, \ \forall j \in N, \tag{3d}$$

$$\sum_{i \in F} a_{ij} x_{ij} \leq w_j, \ \forall j \in N, \tag{3e}$$

$$d_i \in \{0, 1\}, \ \forall i \in F, \tag{3f}$$

$$y_j \in \{0, 1\}, \ \forall j \in N, \tag{3g}$$

$$x_{ij} \in \mathbb{Z}^+, \forall i \in F, j \in N. \tag{3h}$$

Attacker Oracle Problem.

Given a defender's SAM allocations, the Attacker Oracle sub-problem solves for the best allocation of AMs, subject to a defender's subsequent response by launching a fixed number of IMs to intercept them from a set of SAM batteries at predetermined locations. In other words, we seek to find some s_a^i that satisfies $u(s_d, s_a^i) \leq u(s_d, s_a^{i'})$ for all $s_a^{i'} \in S_a$. This problem can be modeled as a bilevel nonlinear integer program, and we utilize a heuristic to obtain a solution. This problem is similar to WTAP [23], as we seek optimal weapon target assignments, but the probability of destruction, p, is not a characteristic of the attacker's AM. Rather, it is a characteristic of the defender's IM. Because we utilize heuristics, we have that $u(\hat{s}_d, \bar{s}_a) \leq u(\hat{s}_d, s_a^i), \quad \forall s_a^i \in S_a$ does not necessarily hold.

$$\textbf{AttackerO:} \underset{\boldsymbol{w}}{\text{minimize}} \quad \underset{\boldsymbol{x},\boldsymbol{y}}{\text{maximize}} \quad \sum_{j \in N} v_j y_j p^{w_j} \tag{4a}$$

$$\text{subject to} \quad \sum_{j \in N} w_j \leq n, \tag{4b}$$

$$\sum_{j \in N} x_{ij} \leq c d_i, \ \forall i \in F, \tag{4c}$$

$$\sum_{i \in F} a_{ij} x_{ij} \geq w_j y_j, \ \forall j \in N, \tag{4d}$$

$$\sum_{i \in F} a_{ij} x_{ij} \leq w_j, \ \forall j \in N, \tag{4e}$$

$$w_j \in \mathbb{Z}^+, \forall j \in N, \tag{4f}$$

$$y_j \in \{0, 1\}, \ \forall j \in N, \tag{4g}$$

$$x_{ij} \in \mathbb{Z}^+, \forall i \in F, j \in N. \tag{4h}$$

In order to solve the Attacker Oracle sub-problem, we utilize a neighborhood search as shown in Algorithm 3. We give an initial feasible strategy, after which we iteratively generate neighboring strategies and move to the best such neighboring strategies. We use the notion of 2-opt exchange when generating our neighboring strategies. In this context, a neighborhood of a given attacker strategy is a set of all strategies that has exactly one AM moved from any city $j \in N$ to some city $j' \in N, \ j \neq j'$. For example, if a given initial strategy is [0 4 3], where 4 AMs are allocated to city 2 and 3 AMs to city 3 in a 3-city network, the neighborhood of solutions generated from such an attacker strategy for a possible 2-opt exchange would be as follows:

$$\textbf{neighbors}([0\ 4\ 3]) = \begin{bmatrix} 0 & 4 & 3 \\ 1 & 3 & 3 \\ 0 & 3 & 4 \\ 1 & 4 & 2 \\ 0 & 5 & 2 \end{bmatrix}.$$

This neighborhood search heuristic terminates when the incumbent feasible strategy gives the best payoff relative to its own neighborhood (i.e., $\bar{s}_a = \mathbb{B}(1)$). For the Attacker Oracle, the search terminates for an initial strategy s_a^i if $u(\hat{s}_d, s_a^i) < u(\hat{s}_d, s_a^{i'}), \quad \forall s_a^{i'} \in \mathbb{B}$, where $\mathbb{B}$ is the set of neighborhood strategies generated.

Algorithm 3 Attacker Oracle

```
1: s̄_a ← s_a^1
2: ū := ∞
3: repeat
4:     B ← neighbors(s̄_a)
5:     for i = 1 to |B| do
6:         if u(ŝ_d, B(i)) < ū then
7:             s̄_a ← B(i)
8:             ū ← u(ŝ_d, B(i))
9:         end
10:    end
11: until s̄_a = B(1)
12: return s̄_a
```

In the following chapter, we present an example network instance to compare the solution quality, algorithm quality, and robustness of the methodologies presented in this chapter using an experimental design to examine the effect of parameters on computation time and payoffs. In addition, we provide a scale up analysis of Double Oracle for problems that are infeasible to solve using exhaustive enumeration or the $\alpha - \beta$ pruning algorithm.

IV. Computational Results

In this chapter, we present a small network to apply our proposed solution methodologies in the previous chapter. We conduct experiments on different problem instances on the small network and present the findings. Subsequently, we present a network that is too large for all but Double Oracle to be applied and show the performance of Double Oracle.

4.1 Analysis on 6-city Network

We first examine the 6-city network used in Daskin's MEXCLP [22], as shown in Figure 2, with the same parameters and assumptions: $F = N$ and $r = 9$. We examine the 6-city network first because it is a small enough network to implement all methodologies proposed in the previous chapter.

Because exhaustive enumeration and $\alpha - \beta$ are computationally expensive, we generate problem instances via a Central Composite Design (CCD) having the values shown in Table 4. Excluding the repeated center runs, we solve a total of 25 problem instances and investigate the effect of the number of SAM batteries (m), number of AMs (n), number of IMs (c), and the probability of single intercept (p), on the required computation time and the solution quality. For each problem instance, we apply six solution methodologies: exhaustive enumeration, $\alpha - \beta$ pruning using each of four different search order combinations for the respective defender and attacker levels of the game tree, and Double Oracle. All instances were solved by invoking IBM ILOG CPLEX Optimization Studio (version 12.6) with MATLAB on a Intel(R) Xeon E5-1620 3.6 GHz processor having 32 GB memory.

The resulting computation times are reported in Table 5. Columns m, n, c, p represent a combination of defender and attacker capabilities that comprise each

Table 4. Central Composite Design

	m	n	c	p
High	4	20	16	.7
Low	2	10	8	.3

instance, represented in a given row. The first sub-column of computation time represents the time required to fully enumerate the game tree and identify the SPNE. In successive sub-columns, four variations of $\alpha - \beta$ search represent the order in which the tree was searched (See Method 2 in Chapter III for an explanation). Finally, the DoubleO column reports the computation time required for the heuristic to terminate. An asterisk (*) indicates that Double Oracle terminated before identifying the SPNE.

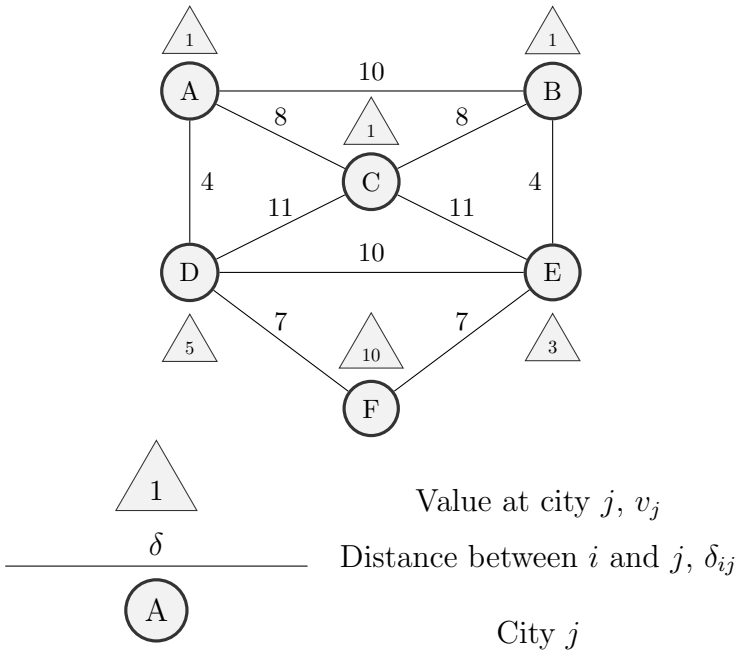


Figure 2. 6-city Network

Solution Quality

Both exhaustive enumeration and the $\alpha - \beta$ variants are exact algorithms. Based on the effects test using the result of Table 5, Table 6 shows that all four parameters comprising a problem instance (i.e., m, n, c, p) have a statistically significant effect

Table 5. Computation time on 6-city Network

Problem Instance				Computation time (sec)					
m	n	c	p	full	$\alpha\beta 11$	$\alpha\beta 12$	$\alpha\beta 21$	$\alpha\beta 22$	DoubleO
3	5	12	0.5	9.75	0.77	8.78	1.25	8.32	0.41
2	10	8	0.3	88.58	35.25	42.78	35.07	42.15	1.35
2	10	8	0.7	88.33	19.35	50.09	18.28	44.16	1.55*
2	10	16	0.3	87.50	34.74	42.35	34.61	41.67	1.44
2	10	16	0.7	86.40	18.24	53.18	17.73	46.82	1.71
4	10	8	0.7	93.69	7.06	94.20	7.12	94.28	0.84
4	10	8	0.3	105.84	48.00	58.03	52.39	66.90	1.09
4	10	16	0.3	101.55	45.78	62.05	46.36	63.24	0.99
4	10	16	0.7	89.98	6.74	91.57	6.75	91.16	1.30
1	15	12	0.5	190.55	79.39	142.41	140.60	53.74	3.92*
5	15	12	0.5	203.29	83.19	155.41	83.40	155.57	2.16
3	15	4	0.5	1288.50	466.71	727.49	486.80	655.39	5.59*
3	15	12	0.1	796.86	454.52	383.54	455.20	380.27	1.77
3	15	12	0.5	924.61	262.38	634.65	276.23	604.69	1.40
3	15	12	0.9	764.09	63.60	758.64	99.63	553.48	3.91
3	15	20	0.5	773.12	226.12	523.28	226.56	508.31	1.63
2	20	8	0.3	3688.33	2263.86	1331.19	1942.82	1496.63	5.41
2	20	8	0.7	3189.35	921.37	1751.27	795.94	1697.74	8.72
2	20	16	0.3	2928.64	1835.90	789.69	1429.54	1165.64	2.92
2	20	16	0.7	2481.01	914.47	867.86	365.81	1433.88	3.14
4	20	8	0.3	3792.72	2360.05	1691.40	2390.64	1698.26	1.12
4	20	8	0.7	3689.57	652.39	3682.91	934.97	3196.53	2.25
4	20	16	0.3	3136.53	2006.02	1370.78	1991.22	1378.16	2.08
4	20	16	0.7	3101.86	665.69	3018.79	668.44	3023.00	2.52
3	25	12	0.5	24757.28	10955.51	12599.12	10979.81	12376.85	5.44

* indicates that Double Oracle terminated before identifying the SPNE.

on the SPNE payoff. On the other hand, Double Oracle is a heuristic, so it cannot guarantee to find the SPNE before it terminates. As shown in Table 5 under the DoubleO column, 3 out of 25 instances (12%) fail to reach the SPNE and provide a suboptimal solution. In all three instances, the reported payoff was less than the SPNE payoff. Such results indicate that the defender (maximizer) failed to generate correct strategies. Conversely, a reported payoff larger than the SPNE payoff indicates that the attacker (minimizer) failed to reach the global minimum in the Attacker Oracle procedure.

Table 6. Effects Test on Payoff

Variable	F Ratio	p-Value
m	22.4804	< 0.0001
n	95.2264	< 0.0001
c	13.9343	< 0.0001
p	805.5464	< 0.0001

Table 7. Double Oracle Suboptimal Instances

Problem Instance				Payoff		
m	n	c	p	SPNE	DoubleO	% Gap
2	10	8	0.7	7.676	7.216	6.0%
1	15	12	0.5	1.000	0.719	28.1%
3	15	4	0.5	2.188	1.313	40.0%

Algorithm Quality

In this section, we discuss how efficiently each algorithm converges to the SPNE. We primarily examine computation times, since exhaustive enumeration and $\alpha - \beta$ are guaranteed to converge to the SPNE. Also, we compare exhaustive enumeration and $\alpha - \beta$ with Double Oracle in terms of computation time to show how well Double Oracle scales in larger instances.

Table 8. Computational Efficiency

Method	Average % of S pruned
$\alpha\beta 11$	64.4%
$\alpha\beta 12$	41.4%
$\alpha\beta 21$	64.1%
$\alpha\beta 22$	38.5%

Comparison of Search Order in $\alpha - \beta$

The order in which we search the respective levels of the game tree appears to exhibit a difference on the required computation time. On average, $\alpha\beta 11$ obtains the fastest computation time or, equivalently stated, $\alpha\beta 11$ needed to compute payoffs (i.e., Equation 2) the least before terminating. Such a result is intuitive, as we expect to find the SPNE near the top of the lexicographically ordered strategies (i.e., the defender covers its most valuable cities, and the attacker attacks the most valued cities).

Table 8 shows the proportion of the entire strategy space, S, generated before convergence for the 25 CCD instances. Detailed results are shown in Table 9.

Robustness of the Algorithms

While Double Oracle failed to identify the SPNE in 3 out of 25 instances, the algorithm does scale well with respect to computation time. For the largest problem instance of $m = 3$, $n = 25$, $c = 12$, $p = 0.5$, the exhaustive enumeration procedure required $|S| = \tau_d \tau_a = \binom{6}{3}\binom{6+25-1}{25} = 2850120$ combinations of defender-attacker strategies to calculate and terminated after about 7 hours of computation time. The best case $\alpha - \beta$ algorithmic variant, $\alpha\beta 11$, found the SPNE in 10955 seconds (≈ 3.04 hours). In contrast, Double Oracle terminated in 5.44 seconds after identifying the SPNE. Although $\alpha - \beta$ pruning invokes the stage 3 calculation of payoff functions significantly fewer times in the game tree than the exhaustive enumeration procedure,

Table 9. % of S generated on 6-city Network

Problem Instance				% generated					
m	n	c	p	full	$\alpha\beta 11$	$\alpha\beta 12$	$\alpha\beta 21$	$\alpha\beta 22$	DoubleO
3	5	12	0.5	100%	7.94%	84.96%	12.84%	89.60%	0.179%
2	10	8	0.3	100%	39.86%	47.49%	39.67%	48.21%	0.027%
2	10	8	0.7	100%	21.91%	49.65%	20.95%	56.41%	0.027%*
2	10	16	0.3	100%	39.86%	47.49%	39.67%	48.21%	0.027%
2	10	16	0.7	100%	21.31%	53.82%	20.71%	60.82%	0.036%
4	10	8	0.3	100%	46.57%	60.12%	46.57%	60.12%	0.013%
4	10	8	0.7	100%	7.69%	99.01%	7.69%	99.01%	0.013%
4	10	16	0.3	100%	46.57%	60.12%	46.57%	60.12%	0.013%
4	10	16	0.7	100%	7.69%	99.01%	7.69%	99.01%	0.013%
1	15	12	0.5	100%	42.74%	29.56%	71.30%	72.11%	0.026%*
5	15	12	0.5	100%	41.99%	74.68%	41.99%	74.68%	0.006%
3	15	4	0.5	100%	37.67%	48.68%	39.34%	54.65%	0.003%*
3	15	12	0.5	100%	31.66%	61.18%	31.88%	62.53%	0.003%
3	15	12	0.9	100%	10.16%	64.97%	15.21%	82.16%	0.005%
3	15	20	0.5	100%	31.66%	61.18%	31.88%	62.53%	0.003%
2	20	8	0.3	100%	57.47%	44.72%	49.09%	37.92%	0.003%
2	20	8	0.7	100%	31.95%	48.92%	27.69%	49.63%	0.004%
2	20	16	0.3	100%	54.59%	37.91%	49.23%	33.27%	0.001%
2	20	16	0.7	100%	36.56%	49.23%	18.11%	34.86%	0.002%
4	20	8	0.3	100%	59.51%	47.16%	59.51%	47.16%	0.001%
4	20	8	0.7	100%	23.89%	72.69%	31.04%	79.74%	0.001%
4	20	16	0.3	100%	59.51%	47.16%	59.51%	47.16%	0.001%
4	20	16	0.7	100%	28.54%	78.12%	28.54%	78.12%	0.001%
3	25	12	0.5	100%	45.45%	48.85%	45.51%	49.48%	0.001%

* indicates that Double Oracle terminated before identifying the SPNE.

the method does not scale well for large instances.

In fact, neither exhaustive enumeration nor $\alpha - \beta$ scale well to larger instances. Because the required computation time for the exhaustive enumeration and $\alpha - \beta$ solution methods are directly related to the size of the Cartesian product of the defender and attacker strategies (i.e., $|S|$), as n and/or $|N|$ increases, the time required to identify the SPNE grows in relation to the problem's size. For an instance having $m = 3$, $n = 25$, $c = 12$, and $p = 0.5$, we have $|S| = 2850120$ via exhaustive enumeration, as calculated previously. As n increases to 35 and 45, $|S|$ increases to approximately 13.1 million and 42.4 million, respectively. However, Double Oracle can still handle such instances with little computational effort. Double Oracle terminated after 10.95 seconds for $n = 35$ and after 18.01 seconds for $n = 45$.

4.2 Scaled Up Analysis

This section presents the result of experiments using Double Oracle on the 55-city network utilized by Daskin [22], as shown in Figure 3. The cities are distributed on a Cartesian plain, and their values are monotonically non-increasing in order (i.e., $v_i \geq v_j, \forall\ i < j$). The total value of the cities, $\sum_{j \in N} v_j$, is 6400. We apply the same assumptions (i.e., $F = N$ and $r = 15$) as Daskin's MEXCLP, and we use $c = 20$ and $p = 0.9$ as representative for our instance, based on the operating characteristics of Iron Dome [28].

Exhaustive enumeration and $\alpha - \beta$ are not practical to use for this network size, since τ_d and τ_a are intractably large. Even for the smallest instance we examine in the 55-city network (i.e., $m = 1, n = 10$, and $|N| = 55$), we find that $\tau_d = \binom{55}{1}$ and $\tau_a = \binom{55+10-1}{10}$, thus $|S| = \tau_d \tau_a \approx 8.33 \times 10^{12}$. Each payoff calculation was approximately $2 \times 10^{-3} s$, which means CPLEX processes about 1000 stage 3 calculations in every 2 seconds. Assuming equal calculation time, the exhaustive enumeration would take

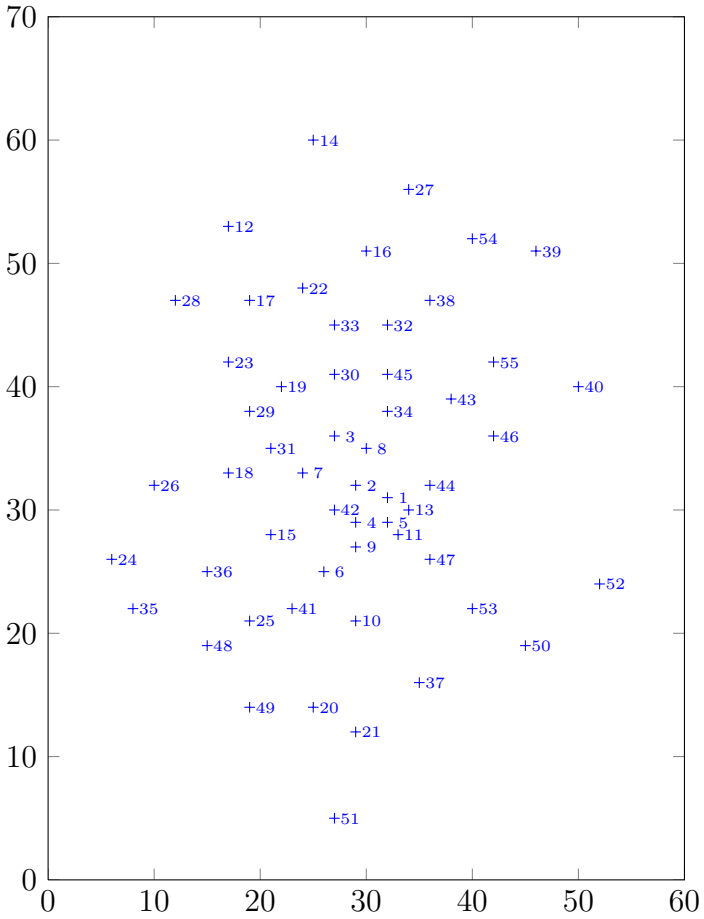


Figure 3. 55-city Network. +: Cities

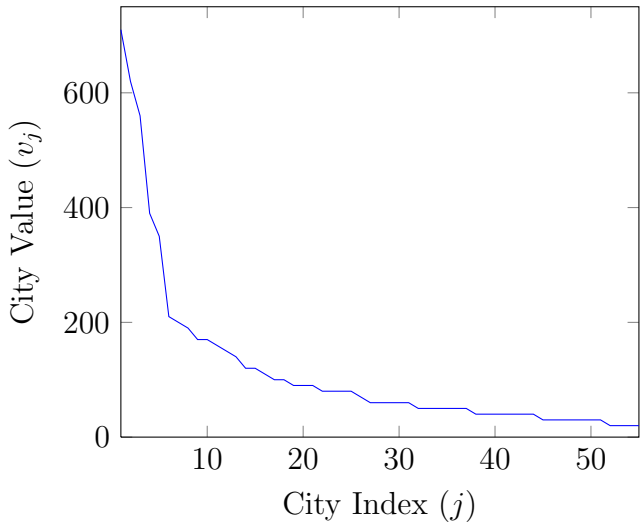


Figure 4. Value Distribution of 55-city Network

$(8.33 \times 10^{12})(2 \times 10^{-3})s \approx 1.67 \times 10^{10}s \approx 528$ years. Thus, we cannot verify the optimality of the results of Double Oracle through exhaustive enumeration or $\alpha - \beta$ for large instances.

Table 10. Computation time (s) on 55-city Network with $c = 20$ and $p = 0.9$.

		Number of SAM batteries (m)			
		1	3	6	10
Number of AMs(n)	10	23.83	18.49	22.17	15.00
	20	75.83	59.90	51.18	40.20
	30	120.21	98.37	206.75	130.88
	40	274.42	212.91	213.01	189.07
	50	470.88	365.37	240.97	261.47

Table 11. Approximate SPNE Payoffs on 55-city Network with $c = 20$ and $p = 0.9$.

		Number of SAM batteries (m)			
		1	3	6	10
Number of AMs(n)	10	5400.00	5831.41	5553.10	5834.83
	20	4763.79	5190.84	5414.11	5435.66
	30	4165.66	5051.60	5077.03	5119.18
	40	3585.66	4705.86	4862.84	4862.84
	50	3281.05	4427.80	4749.59	4652.31

Total value of the cities in this network is 6400.

Table 10 shows the computation time before Double Oracle terminates with varying n and m. Table 11 shows the payoffs for the defender with varying n and m. Notice that defender's payoff sometimes decreases as it gets more resources. For example, at $n = 10$, the defender gets payoff of 5831.41 with $m = 3$, but 5751.63 with $m = 6$. Such phenomenon could be attributed to the Double Oracle failing to identify the SPNE, being a heuristic.

Example Result

Shown in Figure 5 and 6 are example result of a Double Oracle executed on a specific instance. The interpretation of the plot is that after the three stages, we

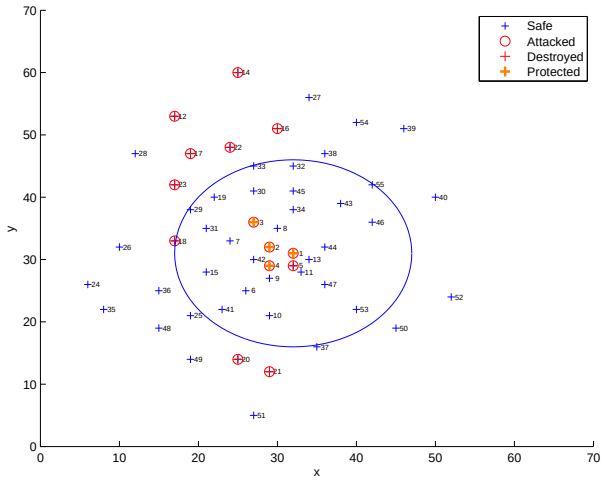


Figure 5. $m = 1$, $n = 30$, $c = 20$, $p = 0.9$

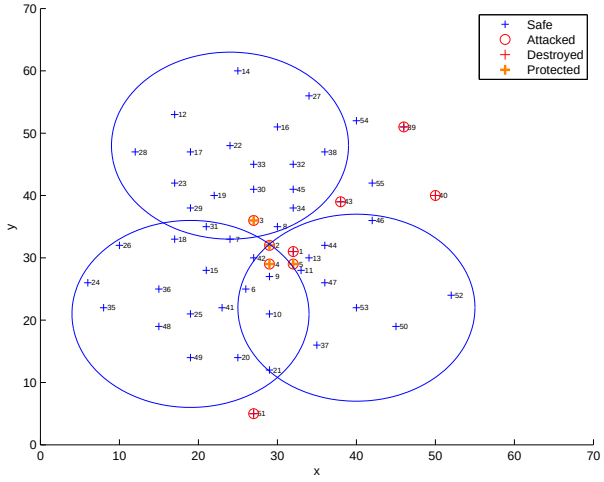


Figure 6. $m = 3$, $n = 50$, $c = 20$, $p = 0.9$

expect to see what is shown in the plot. For Figure 5, in the first stage, we expect the defender to place SAM at city 1; in the second stage, we expect the attacker to launch AMs at cities with red circles; in the third stage we expect the defender launch interceptors to protect cities 1,2,3, and 4. Also, if we assume that the outcome is the SPNE, then we can assume that one who deviates from the displayed outcome will achieve less payoff than the current outcome.

Sensitivity Analysis

Shown in Figure 7 is a sensitivity of payoff in changing p. The monotonic increase in payoff as p increases is consistent with what we expect. Also, we expect smaller payoff with larger n, and such trend is only violated past $p = 0.9$.

Figure 8 shows the increase in computation time as n increases, but except for $n = 50$, trend of computation time as p changes does not seem to exist. Also, unlike exhaustive enumeration and $\alpha - \beta$, computation time does not increase exponentially as n increases for Double Oracle.

Figure 9 and 10 show a side-by-side comparison of the expected outcome if the city distribution is reversed.

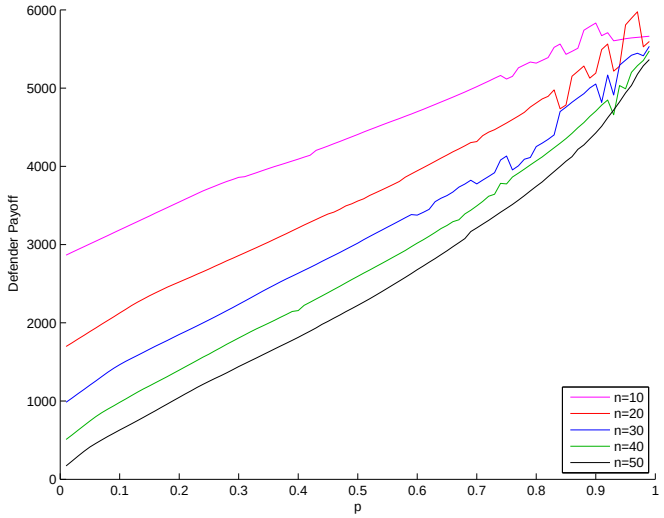


Figure 7. p vs. Defender Payoff with $m = 3$, $c = 20$.

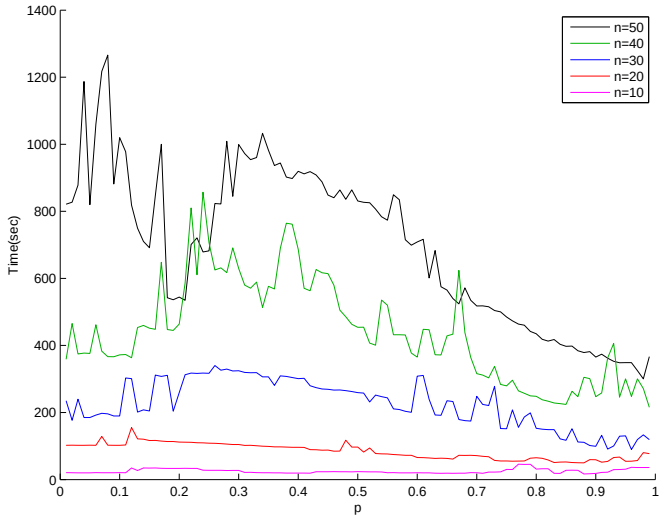


Figure 8. p vs.computation time with $m = 3$, $c = 20$.

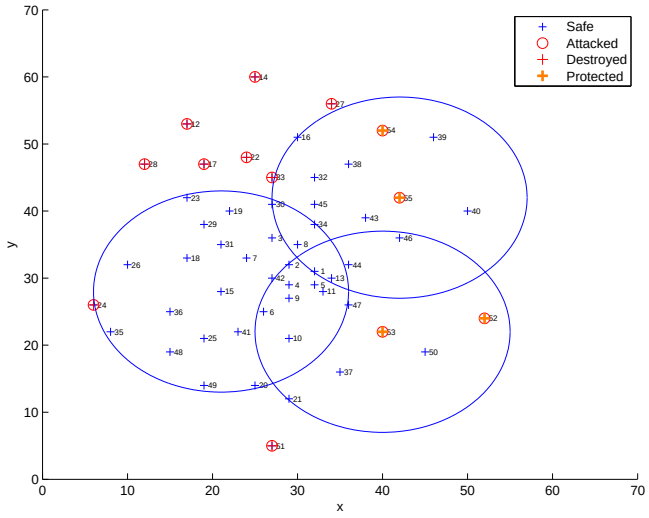


Figure 9. Reverse Ordered Cities with $m = 3$, $n = 30$, $c = 20$, $p = 0.9$

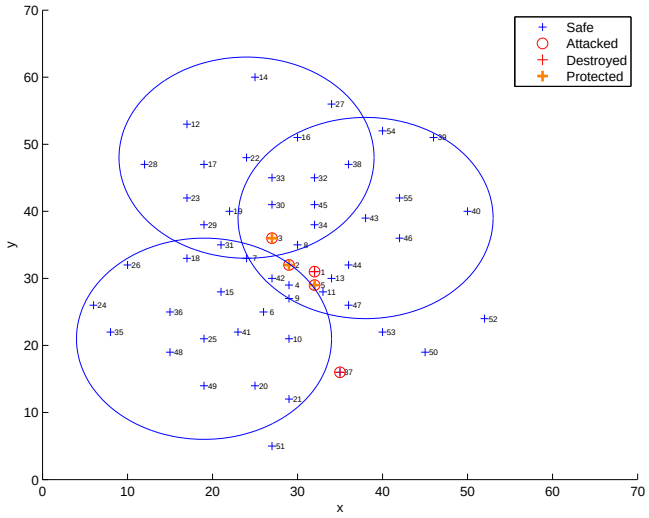


Figure 10. Original Order

V. Conclusions

5.1 Summary and Conclusions

Advanced technologies for missiles and guidance systems have increased the threat to the U.S. national security. Optimal disposition of IADS assets is an important problem for the U.S military and its allies for that reason. In this thesis, we present a trilevel nonlinear integer program to model the Defender-Attacker-Defender situation as a two player, three-stage sequential, perfect and complete information, zero-sum game. We initially attempt to solve by enumerating all possible strategies as a game tree. However, the size of the strategy space increase to intractable size because this is a combinatorial problem. We attempt to be more efficient by using $\alpha - \beta$ pruning, but $\alpha - \beta$ does not scale well. Then, we adapt Double Oracle algorithm to solve for the pure strategy SPNE in order to solve problems that are realistic in size. We present algorithm quality, solution quality, and sensitivity analysis.

In conclusion, we found that Double Oracle is the only methodology that can provide the solution quickly. We also demonstrated that the time to solve a Double Oracle instance does not increase exponentially as do exhaustive enumeration and $\alpha - \beta$. While Double Oracle is not an exact algorithm in our adaptation, we show that in most cases, it does not violate monotonicity and the solution quality trends as we expected.

5.2 Potential Future Research

Relaxing Assumptions

Many of the model assumptions in this thesis can be relaxed and examined for future research. General assumption 1 states that all parameters are common knowledge. Relaxing this assumption requires the game model to have imperfect informa-

tion, which is more computationally expensive. General assumption 2 states that the defender and attacker value targets equally. Relaxing this assumption will make the game non-zero-sum.

Defender assumption 1 states that a SAM battery can only be co-located with a city. Relaxing this assumption will make the defender's strategy space continuous, which would make the model more realistic, but it would be computationally expensive. Defender assumption 2 states that at most one SAM battery can be located at a given site. Relaxing this assumption will increase the number of possible defender strategies from $\binom{|F|}{m}$ to $\binom{|F|+m-1}{m}$, but given Double Oracle's performance, increased strategy space should not be an issue computationally. Defender assumption 3 states that only one interceptor will be launched against incoming AM. This assumption was put in place assuming that the defender will not have time to observe and react to failed interception. However, relaxing this assumption will allow the defender to protect cities with large values with more redundancy. In order to relax this assumption, a penalty function for over-launching IM must be implemented.

Attacker assumption 2 states that all missiles are identical. This assumption may not be true, as different attacker missiles have different flying characteristics. If relaxed, the probability of a single intercept, p, will become a vector that represents the characteristics of different AMs. Attacker assumption 3 states that an attacker will fire all of its AMs in a single salvo. If relaxed, the attacker stage becomes dynamic, and the attacker oracle will resemble a dynamic weapon target assignment, as opposed to the static weapon target assignment. This would make the problem more difficult and Double Oracle may not be applicable anymore, but the model would be more realistic. Simulation may be a good tool to implement the relaxation of attacker assumption 3.

Improvement on Tree Search Algorithm

Besides $\alpha - \beta$, there are many other tree-search algorithms that have been proven to perform better in artificial intelligence field. Some of the examples include SSS* [29], NegaScout [30], and MTD(f) [31]. For the purpose of further research in this topic, we do not recommend examining tree-search algorithms. As we have a finite and known game tree (defender-attacker-defender) depth, there are more efficient algorithms to implement, such as Double Oracle.

Improvement on Double Oracle

Our current implementation of Double Oracle as developed by Jain et al. [18] can be improved in several ways. Not generating mixed strategies for coreTree may have an impact on identifying the SPNE before the algorithm terminates. Also, one could consider generating multiple best-response functions via the Defender and Attacker Oracle routines. Also, the Attacker Oracle heuristic can be improved upon with regard to its neighborhood search. A closer examination of VLSN techniques by Ahuja et al. [32] would help in that case. Finally, one could examine an improved version of the Double Oracle solution method developed by Jain et al. [33]: Securing Networks by Applying a Randomized Emplacement Strategy (SNARES).

A Game Theoretic Model for the Optimal Disposition of Integrated Air Defense System Assets

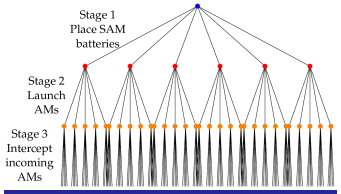
2d Lt Chan Y. Han
Co-advisor: LTC Brian J. Lunday, PhD
Co-advisor: Lt Col Matthew J. Robbins, PhD
Department of Operational Sciences (ENS)
Air Force Institute of Technology

Objectives:

- Inform decision makers on optimal placement of IADS assets
- Model a situation with a defender and an attacker where in the 1st stage, the defender places surface-to-air missile (SAM) batteries, in the 2nd stage, the attacker launches attacker missiles (AM), and in the 3rd stage, the defender launches interceptor missiles (IM)
- Find the Subgame Perfect Nash Equilibrium (SPNE) of the three-stage game

Literature Review:

- Maximum Expected Covering Location Problem
- Weapon Target Assignment Problem
- Double Oracle Algorithm for Zero-Sum Security Games



Approach:

- Model as a defender-attacker-defender, three-stage sequential, perfect information, zero-sum game
- Solve using exhaustive enumeration, α-β pruning, and Double Oracle algorithm

$$\max_{d} \min_{w} \max_{x,y} \sum_{j \in N} v_j y_j p^{w_j}$$ ← Defender's Expected City Survival Values

subject to

$$\sum_{i \in F} d_i = m,$$
$$\sum_{j \in N} w_j = n,$$
$$\sum_{j \in N} x_{ij} \le c d_i, \quad \forall i \in F,$$ ← Resource constraints

$$\sum_{i \in F} a_{ij} x_{ij} \le w_j, \quad \forall j \in N,$$ ← "No-waste" constraint

$$\sum_{i \in F} a_{ij} x_{ij} \ge w_j y_j, \quad \forall j \in N,$$ ← Protection constraint

$$d_i \in \{0,1\}, \quad \forall i \in F,$$
$$w_j \in \mathbb{Z}^+, \quad \forall j \in N,$$
$$x_{ij} \in \mathbb{Z}^+, \quad \forall i \in F, \quad j \in N,$$
$$y_j \in \{0,1\}, \quad \forall j \in N.$$ ← Variable constraints

Challenges:

- Nonlinear integer programs can't be solved directly so we collapse the third stage to make it a two-stage problem
- Problem size increases combinatorially, so exhaustive enumeration and α-β pruning are not tractable

Double Oracle:

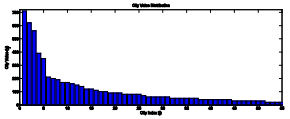
- Solves a large-scale combinatorial optimization problem by only considering strategies that matter
- Utilizes coreTree, Defender Oracle, and Attacker Oracle
- coreTree is a backward induction of the restricted game
- Oracles solve for the best response given the SPNE of the opponent in the restricted game

Double Oracle Pseudocode

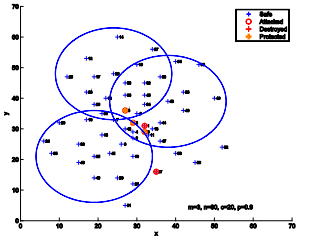
1: $k := 0$
2: Initialize S_d^0 by generating an arbitrary set of defender strategies.
3: Initialize S_a^0 by generating an arbitrary set of attacker strategies.
4: **repeat**
5: $k \leftarrow k+1$
6: $(\delta_d, \delta_a) \leftarrow$ coreTree(S_d^{k-1}, S_a^{k-1})
7: $\delta_d \leftarrow$ DefenderO (δ_a)
8: $S_d^k \leftarrow S_d^{k-1} \cup \delta_d$
9: $\delta_a \leftarrow$ AttackerO (δ_d)
10: $S_a^k \leftarrow S_a^{k-1} \cup \delta_a$
11: **until** $S_d^k = S_d^{k-1}$ and $S_a^k = S_a^{k-1}$
12: **return** δ_d, δ_a

Computation time (s) with $c = 20$ and $p = 0.9$ on 55-city Network

		Number of SAM batteries (m)			
		1	3	6	10
Number of AMs(n)	10	23.83	18.49	22.17	15.00
	20	75.83	59.90	51.18	40.20
	30	120.21	98.37	206.75	130.88
	40	274.42	212.91	213.01	189.07
	50	470.88	365.37	240.97	261.47

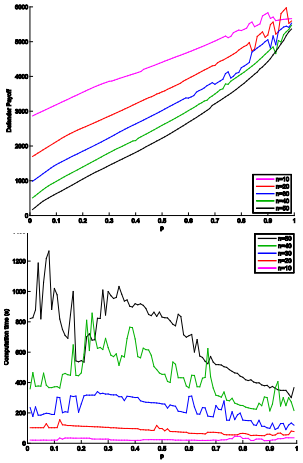


Approximate SPNE



Interpretation: Plot above is the expected outcome and neither the defender nor the attacker has an incentive to deviate

Double Oracle Performance with m=3, c=20



Conclusions:

- Trilevel nonlinear integer program can't be solved directly
- Double Oracle can aid the decision makers to optimally allocate its resources in a reasonable time

Further Studies:

- Consider multiple types of missiles/interceptors
- Consider real-life city networks

AIR FORCE INSTITUTE OF TECHNOLOGY

Bibliography

1. Watson Institute for International Studies, Brown University, "Summary costs of war Iraq, Afghanistan, and Pakistan FY2001-2014," 2014. [Online; accessed 6-September-2014;http://www.costsofwar.org].

2. C. Morello and G. Witte, "Crimeas parliament votes to join russia." http://www.washingtonpost.com/world/crimeas-parliament-votes-to-join-russia/2014/03/17/5c3b96ca-adba-11e3-9627-c65021d6d572_story.html, March 2014.

3. "Gaza-Israel conflict: Is the fighting over?," August 2014. [Online; accessed 6-September-2014; http://www.bbc.com/news/world-middle-east-28252155].

4. National Aeronautics and Space Administration, "Brief history of rockets," 2009. [Online; accessed 5-January-2015; http://www.grc.nasa.gov/WWW/k-12/TRC/Rockets/history_of_rockets.html].

5. J. Sharp, "U.S. Foreign Aid to Israel," *CRS Report for Congress*, 2014.

6. J. Zanotti, "Israel: Background and U.S. Relations," *CRS Report for Congress*, 2014.

7. M. A. Piotrowski, "South Korea's Aid and Missile Defence: Below the Threat Level," *Bulletin*, 2014.

8. U.S. Joint Chiefs of Staff, *Joint Publication 3-01: Countering Air and Missile Threats*, 2014.

9. J. F. Nash, "Equilibrium points in n-person games," *Proceedings of the National Academy of Sciences*, vol. 36, no. 1, pp. 48–49, 1950.

10. R. Selten, "Reexamination of the perfectness concept for equilibrium points in extensive games," *International Journal of Game Theory*, vol. 4, no. 1, pp. 25–55, 1975.

11. D. Fudenberg and J. Tirole, *Game Theory*. MIT Press, 1991.

12. E. D. Demaine, "Playing games with algorithms: Algorithmic combinatorial game theory," in *Mathematical Foundations of Computer Science 2001*, pp. 18–33, Springer, 2001.

13. C. E. Shannon, "Programming a computer for playing chess," *Philosophical magazine*, vol. 41, no. 314, pp. 256–275, 1950.

14. A. Parker, D. Nau, and V. Subrahmanian, "Game-tree search with combinatorially large belief states," in *IJCAI*, pp. 254–259, 2005.

15. D. E. Knuth and R. W. Moore, "An analysis of alpha-beta pruning," *Artificial Intelligence*, vol. 6, no. 4, pp. 293–326, 1976.

16. J. Pearl, "The solution for the branching factor of the alpha-beta pruning algorithm and its optimality," *Communications of the ACM*, vol. 25, no. 8, pp. 559–564, 1982.

17. H. B. McMahan, G. J. Gordon, and A. Blum, "Planning in the presence of cost functions controlled by an adversary," in *International Conference on Machine Learning*, pp. 536–543, 2003.

18. M. Jain, D. Korzhyk, O. Vaněk, V. Conitzer, M. Pěchouček, and M. Tambe, "A double oracle algorithm for zero-sum security games on graphs," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 327–334, International Foundation for Autonomous Agents and Multiagent Systems, 2011.

19. M. Jain, E. Kardes, C. Kiekintveld, F. Ordónez, and M. Tambe, "Security games with arbitrary schedules: A branch and price approach.," in *Association for the Advancement of Artificial Intelligence*, 2010.

20. E. Halvorson, V. Conitzer, and R. Parr, "Multi-step multi-sensor hider-seeker games.," in *IJCAI*, vol. 9, pp. 159–166, 2009.

21. A. Washburn and K. Wood, "Two-person zero-sum games for network interdiction," *Operations Research*, vol. 43, no. 2, pp. 243–251, 1995.

22. M. S. Daskin, "A maximum expected covering location model: formulation, properties and heuristic solution," *Transportation Science*, vol. 17, no. 1, pp. 48–70, 1983.

23. R. K. Ahuja, A. Kumar, K. C. Jha, and J. B. Orlin, "Exact and heuristic algorithms for the weapon-target assignment problem," *Operations Research*, vol. 55, no. 6, pp. 1136–1146, 2007.

24. S. Huntsman, "A graded lexicographic index," 2009. [Online; accessed 25-September-2014; `http://blog.eqnets.com`].

25. R. L. Church and M. P. Scaparra, "Protecting critical assets: The r-interdiction median problem with fortification," *Geographical Analysis*, vol. 39, no. 2, pp. 129–146, 2007.

26. G. Brown, M. Carlyle, J. Salmerón, and K. Wood, "Defending critical infrastructure," *Interfaces*, vol. 36, no. 6, pp. 530–544, 2006.

27. K. Hausken, "Protecting complex infrastructures against multiple strategic attackers," *International Journal of Systems Science*, vol. 42, no. 1, pp. 11–29, 2011.

28. Y. S. Shapir, "Lessons from the iron dome," *Military and Strategic Affairs*, vol. 5, no. 1, pp. 81–94, 2013.

29. G. C. Stockman, "A minimax algorithm better than alpha-beta?," *Artificial Intelligence*, vol. 12, no. 2, pp. 179–196, 1979.

30. A. Reinefeld, "An improvement of the scout tree-search algorithm," *ICCA Journal*, vol. 6, no. 4, pp. 4–14, 1983.

31. A. Plaat, J. Schaeffer, W. Pijls, and A. de Bruin, "Best-first fixed-depth minimax algorithms," *Artificial Intelligence*, vol. 87, no. 1, pp. 255–293, 1996.

32. R. K. Ahuja, Ö. Ergun, J. B. Orlin, and A. P. Punnen, "A survey of very large-scale neighborhood search techniques," *Discrete Applied Mathematics*, vol. 123, no. 1, pp. 75–102, 2002.

33. M. Jain, V. Conitzer, and M. Tambe, "Security scheduling for real-world networks," in *Proceedings of the 2013 International Conference on Autonomous agents and multi-agent systems*, pp. 215–222, International Foundation for Autonomous Agents and Multiagent Systems, 2013.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
26–03–2015	Master's Thesis	Sept 2013 — Mar 2015

4. TITLE AND SUBTITLE

A Game Theoretic Model for the Optimal Disposition of Integrated Air Defense System Assets

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Han, Chan Y., Second Lieutenant, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/ENS)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENS-MS-15-M-123

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution Statement A.
Approved for Public Release; distribution unlimited.

13. SUPPLEMENTARY NOTES

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

We examine the optimal allocation of Integrated Air Defense System (IADS) resources to protect a country's assets, formulated as a Defender-Attacker-Defender three-stage sequential, perfect information, zero-sum game between two opponents. We formulate a trilevel nonlinear integer program for this Defender-Attacker-Defender model and seek a subgame perfect Nash equilibrium, for which neither the defender nor the attacker has an incentive to deviate from their respective strategies. Such a trilevel formulation is not solvable via conventional optimization software and an exhaustive enumeration of the game tree based on the discrete set of strategies is intractable for large problem sizes. As such, we test and evaluate variants of a tree pruning algorithm and a customized heuristic, which we benchmark against an exhaustive enumeration. Our tests demonstrate that the pruning strategy is not efficient enough to scale up to a larger problem. We then demonstrate the scalability of the heuristic to show that the model can be applied to a realistic size problem.

15. SUBJECT TERMS

game theory, combinatorial optimization, zero-sum games, trilevel, IADS, Double Oracle

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Lt Col Matthew J.D. Robbins, AFIT/ENS
U	U	U	UU	58	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636, x4539;matthew.robbins@afit.edu