Heterogeneous Air Defense Battery Location: A Game Theoretic Approach

THESIS

MARCH 2016

Nicholas T. Boardman, Second Lieutenant, USAF

AFIT-ENS-MS-16-M-091

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense, or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENS-MS-16-M-091

HETEROGENEOUS AIR DEFENSE BATTERY LOCATION: A GAME THEORETIC APPROACH

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

Nicholas T. Boardman, B.S.

Second Lieutenant, USAF

MARCH 2016

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT-ENS-MS-16-M-091

HETEROGENEOUS AIR DEFENSE BATTERY LOCATION: A GAME THEORETIC APPROACH

THESIS

Nicholas T. Boardman, B.S.
Second Lieutenant, USAF

Committee Membership:

LTC Brian J. Lunday, PhD
Co-Advisor

Lt Col Matthew J. Robbins, PhD
Co-Advisor

Abstract

The Unites States and its allies confront a persistent and evolving threat from missile attacks as nations around the world continue to invest and advance their current capabilities. Within the air defense context of a missile-and-interceptor engagement, a challenge for the defender is that surface to air interceptor missile batteries often must be located to protect high-value targets dispersed over a vast area, subject to an attacker observing the disposition of batteries prior to developing and implementing an attack plan. To model this scenario, we formulate a two-player, three-stage, perfect information, sequential move, zero-sum game that accounts for, respectively, a defender's location of batteries, an attacker's launch of missiles against targets, and a defender's assignment of interceptors to incoming missiles. The resulting trilevel math programming formulation cannot be solved via direct optimization and it is not suitable to solve via full enumeration for realistically-sized instances. We instead utilize the game tree search technique Double Oracle, within which we embed alternative heuristics to solve an important subproblem for the attacker. We test and compare these solution methods to solve a designed set of 26 instances of parametric variation, from which we derive insights regarding the nature of the underlying problem. Whereas full enumeration required up to 8.6 hours to solve the largest instance considered, our superlative implementation of Double Oracle terminates in a maximum of 3.39 seconds over the set of instances, with an average termination time of less than one second. Double Oracle also properly identifies the optimal SPNE strategies in 75% of our test instances and, regarding those instances for which Double Oracle failed, we note that the relative deviation is less than 2.5% from optimal, on average, yielding promise as a solution method to solve realistically-sized instances.

Acknowledgements

I would like to express my sincere gratitude to my advisors, LTC Lunday and Lt Col Robbins, for their continuous guidance and insight during this effort. Without their support this endeavor would not have been possible.

Nicholas T. Boardman

Contents

Page

List of Figures

List of Tables

HETEROGENEOUS AIR DEFENSE BATTERY LOCATION: A GAME THEORETIC APPROACH

I. Introduction

Missile attacks have long been viewed as a diverse and effective alternative for air attacks on enemy targets as they can be safer than using manned aircraft, and they are also more capable of striking long range targets. Germany was one of the first nations to use missile attacks with the V-1 and V-2 missiles during World War II, and since then missiles have been used in conflicts such as the Afghan civil war, the Persian Gulf conflicts, and most recently in Syria [26]. There are over 20 countries with missile system capabilities, and Russia, China, and North Korea are just a few among those investing in advancing their current capabilities. A large concern with missile attacks is that the missiles can be armed with a variety of warheads such as conventional explosives, nuclear, biological, or chemical weapons [26]. Long range ballistic missiles are also capable of striking targets over 3,000 miles away with decisive force. The difficulty in defending against such an attack is that, compared to an attacker missile which only has to hit its target at a fixed location, a defender interceptor has to correctly locate, track, and strike the incoming missile in flight [20]. The U.S. may not face the largest of threats from a missile attack, but there are several U.S. allies that are well within range of a missile strike from neighboring countries. For example, it is estimated that North Korea has as many as 1,000 ballistic missiles capable of reaching South Korea and Japan [20]. In a similar manner, according to Joint Publication 3-01, *Countering Air and Missile Threats*, the predominant threat is most likely from a rogue state or a terrorist group, rather than from a competing

superpower [33].

As such, the U.S. seeks to research and develop air defense systems capable of providing high quality defense against missile attacks and has initiated several domestic and joint programs to achieve this goal. The Standard Missile-3 (SM-3) Block IIA is one such example which began in 2006 working with Japan as a partner. Since the program began, approximately $3 billion has been invested [8]. Of the total amount, the U.S. is responsible for slightly over $2 billion, which highlights just how costly research and acquisition programs are. Considering only acquisition expenditures in FY2015, the U.S. Department of Defense budget request for missile defense programs was $8.2 billion [31]. For FY2016, the request rose to $8.8 billion, an increase of $600 million [32]. Again, both of these amounts do not include spending on research and development for integration of newer technologies and weapon systems.

Acquisition expenditures are also broken down by Army, Air Force, Navy, or Defense-wide spending, with missile defense being a capability within each of the services. Of the $8.8 billion requested for missile defense in FY2016, over $5 billion of the expenditure is classified as Defense-wide rather than being assigned to any one service. A similar scenario applies to the FY2015 request [32]. A few of the air defense systems operated by the U.S. that contribute towards defense spending include the AEGIS missile defense, the Terminal High Altitude Area Defense (THAAD), the Ground-Based Midcourse Defense (GMD), and the Patriot/PAC-3 programs [32]. The SM-3 Block IIA in development with Japan is scheduled to be deployed in Poland beginning in 2018, adding to current assets available [27]. The SM-3 Block IIA is also designed to be deployable on land or at sea, allowing for greater employment flexibility.

The preceding discussion primarily addresses current missile defense assets already developed and available for acquisition. However, focus has also been shifting to

research and develop directed energy and other kinetic missile defense capabilities. The main reason for the growing interest in these areas is that they are better suited for a large-scale missile attack and offer a better cost return [12]. Specifically with regard to directed energy, it offers the advantage of a low cost per shot, large magazine size, and rapid engagement of multiple targets necessary to counter a mass attack [12]. Current missile defense assets are very costly on a per missile basis, and their ability to stop a large-scale missile attack has been questioned [13]. Newer technologies look to combat these criticisms and, with the integration of such technologies, each system offers distinct characteristics that make it advantageous to utilize whether it be coverage radius, probability of successful interception, or acquisition cost.

In a further attempt to combat such attacks, Goure [13] presents the idea of regional missile defense architecture in which nations collaboratively work together in order to provide protection over an area. Specifically mentioned is Israel's defense network which consists of systems such as the Arrow, Iron Dome, and the Patriot. Also discussed are various missile systems used by U.S. allies such as Japan investing in the Patriot, the AEGIS missile system, and the SM-3 Block IIA, as well as South Korea investing in systems such as the Patriot and THAAD. This helps illustrate the need not only for organization over different missile systems, but also between nations to provide adequate defense against a large-scale attack.

The acquisition of air defense systems reflects only part of the challenge to decision makers. Once acquired, air defense batteries must also be properly located to protect the desired area. The mobility of air defense systems suggests they can be deployed in what seems to be an infinite number of locations, whereas the defining characteristic of an asset may make it advantageous for operation at a certain location over a different asset. The combinatorics of allocating SAM batteries would overwhelm decision makers for even moderately-sized instances, and on large-scale instances it

is almost impossible to consider all possible strategies. Meanwhile, a system having a smaller range but a very high probability of successfully intercepting an incoming missile may be preferable to one having a larger range but lower probability of success, even if it means locating more assets within a given area. The dispersed nature of cities and other defense objectives also causes trade-offs between ensuring that high value targets will be sufficiently protected, and preventing other targets from being left completely undefended, given limited resources to accomplish both objectives.

Once the defender allocates SAM batteries, a defense plan of how to use available interceptor missiles still needs to be implemented to properly defend cities targeted by an attack. Simply assigning each air defense asset to protect the city/cities closest to it may result in a suboptimal strategy and cause incoming missiles to strike unopposed. Even though a SAM battery is located at a city, it may be advantageous to use this SAM to protect a second city farther out which only that SAM can protect, and instead use surrounding SAMs to protect the first city. A city may also need to be protected using missiles from multiple surrounding SAM batteries resulting in a large number of missile combinations, each of which yields a different probability the city survives. While strategies such as this that shift interceptor missiles around or use missiles from multiple launching locations may be slightly harder to explore, it may also result in a higher utility for the defender.

The acquisition process alone costs the U.S. billions of dollars every year, adding to the importance of properly deploying and utilizing air defense assets. Once these resources have been developed and acquired, a strategic decision remains concerning where to locate the assets among the various sites. In today's fiscal environment, it is imperative that resources are employed to maximum benefit. If not properly distributed, there is the possibility that a larger number of defense sites will be required, while some locations may not be as effective or provide unnecessary coverage. Even

with all the advances in intelligence operations, threats can also be unpredictable. Having defense plans and alternatives in place can make the difference between successfully intercepting incoming missiles and cities being destroyed. Factors such as cost, the strategic nature of locating resources, and the possibility of catastrophic loss suggest the importance of missile defense and contribute towards the large amount of resources and effort dedicated toward improving upon current capabilities and their strategic deployment around the world.

The remainder of this paper is organized as follows. Chapter II presents documents pertaining to U.S. Integrated Air Defense Systems, along with relevant research and discussion that influence our problem formulation and methodology. Chapter III presents the framework for our problem development and the associated model formulations. Chapter IV discusses the solution methodologies examined to solve the model formulations, and Chapter V provides the results and discussion on various test instances. Chapter VI presents conclusions regarding the implemented solution methods in addition to directions for further research.

II. Literature Review

2.1 Missile Defense Background

Joint Publication 3-01 (JP3-01) defines Integrated Air Defense Systems not as a formal system itself but as the aggregate of air and missile defense (AMD) systems such as sensors, weapons, C2, communications, intelligence systems, and personnel operating in a theater [33]. These systems can further be distinguished as active AMD or passive AMD. Passive AMD relates to detection, warning, or concealment to minimize the effectiveness of enemy air and missile threats [33]. Our focus aligns closer with active AMD which relies on the use of aircraft, weapons, sensors, and other direct defensive measures to destroy or nullify the effectiveness of air and missile threats [33]. JP3-01 also discusses the importance of streamlined coordination in the decision-making process, highlighting the importance of having air defense assets in place and a strategy for defensive counter air operations, should they be necessary. Finally, the JP3-01 mentions several different AMD systems available to the defender such as SAMs, AAA, and electronic warfare systems, which is a motivating feature of this research.

2.2 Weapon Target Assignment Problem

The Weapon Target Assignment (WTA) Problem involves allocating m weapons to n targets such that the expected damage to the enemy's targets is maximized [1]. WTA problems can further be divided into two groups: static or dynamic. Static WTA problems assume a perfect information game with fixed parameters in which all targets are engaged in a single stage [1]. The static WTA problem is similar to the one faced by the attacker in the second stage of our model, in which the attacker must decide on a target assignment for each attacker missile. The dynamic WTA

problem is a multistage game, and strategies in each period of the game can change based on information gained from previous stages.

Ahuja et al. [1] formulate the WTA problem as a minimum cost flow problem, and solve this formulation to obtain a lower bound on the objective function. The author's main focus is then on applying a branch-and-bound algorithm to optimally solve the WTA problem; this algorithm performs very well for small instances but requires a long run time on larger-size instances. To counter the increased run time, they apply a heuristic involving a Very Large-Scale Neighborhood (VLSN) search technique to decrease the required computation time while still finding near-optimal solutions for static WTA problem instances. Ahuja et al. [1] apply the VLSN to solve larger instances of the WTA problem involving up to 200 weapons and 400 targets, and they find near optimal solutions in a matter of seconds.

2.3 Game Theory

Game theory is concerned with the allocation of resources in a strategic environment, wherein the term strategic implies that the payoff to each player is a function not only of their own action but also a function of the actions taken by other players. An important development in the field of game theory is the concept of Nash Equilibrium, and Myerson [24] discusses the development of economic theory and the influence that John Nash had on the study of game theory. Nash [25] explains three critical components to characterize a game: the players, their strategies, and their payoffs. Nash illustrates that every finite game has an equilibrium point, where an equilibrium point is "the set of all pairs of opposing 'good strategies'" [25]. Nash equilibrium can also be interpreted as a set of strategies such that neither player has an incentive to deviate from their strategy selection. Nash concludes by illustrating how these concepts can be applied to a three-person poker game.

Selten [29] used the concept of Nash equilibrium as it applies to sequential move games to explain subgame perfect equilibrium (SPNE). Selten [29] explains that sequential move games can be broken into subgames that will themselves have subgame perfect equilibria. Notably, Selten proves that any finite extensive form game with perfect information will have at least one SPNE.

Galati & Simaan [9] apply the concept of Nash equilibrium to the Multi-Team Target Assignment problem, a slight variation of the WTA. In the multi-team version, instead of only one attacker allocating m weapons, there are now two teams facing the WTA problem simultaneously. In addition, every weapon for one team is a target for the other team, and vice versa [9]. Within the multi-team WTA problem, the concept of Nash equilibrium as a solution has been criticized because it assumes that the opponent is also implementing a Nash equilibrium strategy [9]. Galati & Simaan [9] consider a random strategy, a unit greedy strategy, and a team optimal strategy as three additional strategies that may be selected in the multi-team WTA problem. The team optimal strategy is analogous to the WTA problem because it selects the best allocation of weapons for the team, but it ignores possible strategy choices by the opposing team. Of importance, Galati & Simaan [9] conduct a simulation in which the players implement all 16 different combinations of strategies against each other. The results indicated that, on average, the unit greedy strategy appears to be the worst, whereas the team Nash strategy returned the best payoffs. This result supports the concept of Nash equilibrium (or SPNE) as a viable solution method.

Another criticism is that Nash equilibrium can be computationally expensive to find, especially as the strategy space grows. Due to this, Galati & Simaan [9] also discuss a neighborhood search algorithm called ULTRA. The algorithm works by fixing the strategy of one team, and calculating a reaction strategy for the other team. The strategy for one team can be represented as a vector u_i, and a neighborhood is

calculated that allows no more than ξ entries in u_i to change [9]. The algorithm can then be customized to find very good solutions at the cost of computation time for high values of ξ, or require a short run time but risk getting caught at local optima for small values of ξ.

A common assumption in game theoretic models is that the players involved all act rationally. In terms of the attacker, this implies that the attacker will choose to attack those targets that result in the highest expected utility. Yang et al. [34] suggest that it may also be worth exploring situations wherein the attacker occasionally selects sub-optimal strategies, as a defense strategy that does not take into account the behavior of the attacker may not be robust against attackers who use different decision processes. They mention a model named COBRA which attempts to correct for this assumption, wherein the attacker is allowed to deviate to ϵ-optimal strategies. Further, the optimal strategy may be computationally expensive to find while ϵ-optimal strategies are more readily available.

Yang et al. [34] then present Quantal Response Equilibrium as as solution method, which allows players to select strategies that do not maximize their utility, and the probability of selecting a non-optimal strategy increases as the cost of doing so decreases. They then present a MILP formulation and propose a heuristic called the Best Response to Quantal Response (BRQR) that can be used to solve for quantal response equilibrium strategies. BRQR is essentially a gradient ascent method that starts with a randomly generated point and then moves in the direction that improves the objective function.

2.4 Infrastructure Defense

Brown et al. [5] implement both bilevel and trilevel formulations on infrastructure defense models in addition to providing several interesting comments about their

development. They indicate that the attacker usually has the advantage, as the defender is forced to protect a large target area with finite resources, while the attacker can focus on a subset of this space and still inflict damage. One assumption made by Brown et al. [5] is that a protected resource is invulnerable. With regard to air defense models, this may not hold as a city that is covered by a SAM battery is not necessarily protected or invulnerable.

A related application of bilevel optimization models is the r-Interdiction median problem with fortification (RIMF). This problem involves the defender determining a strategy that selects a subset of facilities to protect in order to minimize damage inflicted from the attacker's interdiction of r facilities [28]. The sequence of events in RIMF problems is very similar to bilevel Defender-Attacker (DA) formulations in which the defender first allocates resources to protect a subset of facilities, and the attacker then observes this strategy and decides which facilities to attack. Scaparra & Church [28] also introduce an implicit enumeration algorithm to solve this bilevel problem, and they provide results for the implementation of this algorithm on instances of various size. They vary the number of attacker interdictions, r, and note one interesting result: there are a few facilities that appeared in every defender fortification strategy. These facilities indicate important locations, and thus should be fortified whether or not an attack is expected on these facilities [28]. This extends to the placement of SAM batteries, as the defender locating a SAM battery at a city may deter the attacker from targeting this city with missiles.

An important aspect of game theoretic problem formulations is determining the structure and relationship of components in a system, as it can affect the strategies employed by both the defender and the attacker. Hausken [18] discusses several aspects related to infrastructure defense, along with potential characteristics to determine the value of each target. Of importance is the consideration that the defender is

not only determining a strategy to protect against one attacker, but the possibility of facing m independent attackers. Hausken [18] also argues that there are three main factors that contribute to the valuation of a target: economic, human, and symbolic value. The total value of the target can then be found by taking a weighted combination of the above factors. Within our model, we do not necessarily require the individual component values, only the final value of each city.

Bier et al. [3] present a small-scale example related to infrastructure defense consisting of only two targets, and an attacker selecting exactly one location to target. They also assume a sequential game in which the defender first decides to allocate their resources between the targets, and the attacker then observes this allocation and decides where to attack. Compared to our assumption of perfect information, they assume that the attacker knows the defender's valuation of each target, but the defender does not know the attacker valuations. Further, there is a probability p_i of a successful attack on target i which is a function of defender resource allocations and targets selected by the attacker. Due to this, Bier et al. [3] point out the attacker will not necessarily always attack the undefended target. For example, if the defender chooses to locate their resource at City 1, then the attacker will still choose to attack this city as long as the expected utility from attacking City 1 is greater than the utility from attacking City 2 [3]. The importance of this result is that a lower-valued target that is left undefended is not necessarily targeted by the attacker.

Bier et al. [3] continue to analyze the effect of changing certain parameters in the problem, such as the defender's valuation of each city, the defender's cost of allocating defensive resources, and the attacker's valuation of each city. As expected, increasing the defender's valuation of City 1 decreases the probability of a successful attack on this city in the optimal solution. The attacker can also observe this change in probability and inform them of the defender's valuations in a case where the attacker

did not originally have this information, adding credibility to the assumption of a perfect information game [3]. Strengthening the defensive resource commitment at one location makes that target appear more important to the defender, so the attacker will generally achieve a higher payoff with a successful attack. The defender must carefully weigh this decision, as strengthening one position means another location may become more exposed, and after a point the attacker may decide to select an easy target [3].

2.5 Target Hardening and Overarching Protection

An important distinction in infrastructure defense models is the characteristics of the defender resources, specifically the capabilities they possess. There are two main distinguishing classes for defender resources: Target Hardening and Overarching Protection resources [17]. Target hardening refers to the case in which the defender protects targets individually, whereas in overarching protection the defender is capable of protecting multiple targets at once. Haphuriwat & Bier [17] argue that overarching protection should be used by the defender when they have a large number of assets to protect, and it will typically be more cost effective than fortifying individual targets. Overarching protection is also more applicable to models that assume a defender asset located at city i is capable of protecting any city within a certain radius.

Haphuriwat & Bier [17] also discuss the idea of using power-law functions to calculate the probability of a successful attack. Power-law functions relate to the idea of diminishing returns in that the first defender resource located at city i will increase the defender's expected utility, but a second or third resource located at the same city will not cause as large an increase. With regard to the defender, after a certain point the probability of a successful attack will drop sufficiently low that the resources would provide a larger benefit elsewhere [17]. This is generally true when

the defender has one type of resource, however when the situation expands to multiple different defender or multiple different attacker resources, the defender may want to place two different resources at the same city to protect against different threats.

2.6 Maximum Expected Covering Location Problem

The set covering and maximum covering location problems are two different problem formulations concerned with a facility or resource being located to cover a set of demand nodes [7]. The maximum covering location problem as discussed by Church & ReVelle [6] involves finding locations to place a set of facilities that maximizes the total demand covered, subject to a maximum service distance. Although this is not necessarily our objective, this is similar to locating a SAM battery such that it covers the maximum number of cities. Daskin [7] presents a variation of this he denotes as the maximum expected covering location problem (MEXCLP) which takes into account that, just because a demand node is within range to be covered by a facility, does not necessarily mean the facility will be able to service the demand node. As a result, one possible objective is to not only maximize the number of demand nodes covered, but also to subsequently maximize the number of nodes that are covered more than once [7].

Daskin [7] also introduces the idea of dominated and non-dominated facility locations. A node j is said to be dominated if there exists another node k such that locating a facility at node k covers every demand node covered by node j, while covering at least one additional node. The concept of dominance is important because if node j is dominated, then we know that in the optimal solution to MEXCLP a facility will never be located at node j [7]. This can also be extended to the defender's placement of SAM batteries based on the coverage radius and cities within range. He then presents a heuristic to solve the MEXCLP and concludes by illustrating the

performance on a frequently used 55-node network.

One key component of MEXCLP that Golalikhani & Zhuang [11] address is that a defense asset located at a node i is capable of protecting not just that node, but nodes within a certain radius of i. Many problem formulations discussed above assume that a resource is only capable of protecting the node at which it is placed, which is not necessarily the case. The capability of a defender resource protecting nodes around it relates to the aforementioned discussion regarding overarching protection and gives rise to the idea of a coverage parameter that indicates whether a resource located at i is capable of reaching node j. Golalikhani & Zhuang [11] demonstrate that this does not necessarily have to be based on the idea of geographical vicinity; it can be based on functional similarity. A coverage radius is also a key benefit to the defender that increases their expected utility. Naturally this also lowers the attacker's utility, and the decrease in the attacker's utility is typically larger than the increase in the defender's utility [11].

2.7 Theater Ballistic Missile Defense

Brown et al. [4] discuss the growing importance of Theater Ballistic Missile (TBM) Defense along with current TBM interceptor platforms and various existing analytical tools to aid in positioning missile defense assets. Brown et al. [4] make similar assumptions to models previously discussed, notably that all parameters are common knowledge, each attacker missile has a fixed probability of kill, p_k, and each interceptor missile has a probability of successfully destroying an attacker missile. Similar to Hausken [18], the value of a target can be broken down into four main factors: criticality, vulnerability, reconstitutability (ability of a target to recover from damage), and threat [4]. They also mention the advantage of adding secrecy to the model (i.e., no longer an assumption of perfect information) and conclude by testing their

formulation on a case study.

2.8 Security Games

Security games can be viewed as a variation of DA models and have been receiving increased attention due to their highly strategic nature, along with the importance of their applications. Security games involve a defender who places a set of resources on a graph to protect vulnerable locations, and an attacker who subsequently chooses a strategy to attack these locations [19]. As such, the development of security games closely aligns to the formulation of our trilevel Defender-Attacker-Defender (DAD) model. Jain et al. [19] use the Mumbai attacks that occurred in 2008 as a motivating example for the application of security games and discuss two methods available to help solve for defender and attacker strategies. They discuss a double-oracle algorithm called RUGGED to solve for player strategies, and they also present an improved algorithm called SNARES that is capable of solving problems much quicker and larger in size. Tsai et al. [30] and Halvorson et al. [15] also discuss how to implement double-oracle algorithms as they relate to security games and provide examples to illustrate an implementation of the algorithm. Double oracle is explained in further detail in the following subsection.

Arce et al. [2] consider the interaction between one player protecting n locations, and another player deciding which locations to attack and with what resources. They also bring up the concept of Colonel Blotto games in which each player allocates a fixed level of resources across N battlefields. The player allocating more resources to a battlefield wins that battlefield, and the objective is to maximize the number of battlefields won [2]. Arce et al. [2] also indicate the problem that resources allocated to one battlefield reduce the number of forces that can be allocated to other battlefields, highlighting how important it is to properly allocate resources. The bat-

tlefield allocation trade-off is related to the assumption that each SAM battery has a fixed radius r of cities it can protect, so the location of a SAM battery greatly influences which cities are covered. Another assumption commonly made in Colonel Blotto games is that the player who allocates more resources to a battlefield wins that battlefield [2]. This is a simplifying assumption that does not necessarily hold for air defense models. For example, if the defender places a SAM battery at city i, it does not necessarily mean that full value of city i is added to the defender's utility. The attacker could decide to attack city i, in which case there is a probability of survival, or the attacker could overwhelm the city and attack with more missiles than the defender could possibly launch interceptors against.

Korzhyk et al. [22] present an algorithm to find an equilibrium in security games that works by progressively adding defender resources. There are two important assumptions underlying this algorithm. First, if a target is not attacked, then it does not affect either player's utility. However, in general a target that is not attacked usually increases the defender's utility as this target is 'safe'. The second assumption is that the utility functions are additive, which simply implies that if the attacker targets two cities, their utility is the sum of the utilities of attacking each target individually [22]. If the defender has k resources to allocate, the algorithm works by starting at a variation of the game assuming the defender has no resources and determines the equilibrium at this point (i.e., what targets are selected by the attacker). This is calculated easily because the attacker will attack those targets that yield the highest utility [22]. The basic premise of the algorithm is to calculate how much the defender's resource level could increase until the attacker's strategy no longer is part of a Nash equilibrium. The defender's resources are then increased by this amount, and the strategies for the defender and the attacker are updated. A new equilibrium is calculated, and the algorithm continues in a similar fashion until the amount of

defender resource equals his actual commitment level k [22].

2.9 Double Oracle

Double Oracle is a solution method to find a Nash Equilibrium (or subgame perfect equilibrium) and solve either a normal form (or extensive form) game. The algorithm initially considers a restricted set of players' strategies and iteratively identifies the NE (or SPNE) to solve this 'restricted game' [19]. For each opponent's NE/SPNE strategy, the adversary's best response strategy is identified among the entire strategy space and added to the restricted game, terminating when no new strategies are added to the restricted game. Jain et al. [19] also provide results indicating that Double Oracle is capable of solving large-sized instances without a drastic increase in computation time.

2.10 Heuristics

While exact solution methods exist, they may not always be practical to implement. For example, the required computation time may not be viable for large-size problem instances, or a heuristic solution may be easier to discover and remain near-optimal. In such cases, heuristics offer an alternative solution method that are not guaranteed to reach a global optimum, but are capable of greatly reducing the computation time. Simulated annealing and tabu search are two heuristic search techniques that leverage different algorithmic properties to ensure high quality solutions are identified.

Simulated annealing is an iterative heuristic search technique based on the annealing process of metal or glass, along with a heuristic described by Metropolis. Metropolis first used the heuristic to find the equilibrium of atoms at a given temperature but it has since been used in other combinatorial optimization problems [21].

The heuristic begins with a current arrangement of atoms and generates a neighbor based on a small perturbation to this current arrangement. If the energy of this neighbor arrangement is less than the current arrangement, the neighbor configuration becomes the new arrangement and the process repeats. If the energy of the neighbor is greater than the current arrangement, the neighbor is moved to with some probability based on the difference in energy levels [21]. Repeating this process simulates the movement of atoms until a steady state is reached. Metropolis' heuristic has been applied to optimization problems typically by letting solutions represent the arrangement of atoms and the objective function value of the solution represent the energy level.

The annealing process is utilized to change the temperature of the system, which affects the probability that 'worse' moves are accepted. Initially the temperature is set very high so worse moves are selected with a higher probability which means that more of the solution space is explored. Over time, the temperature begins to decrease and the focus is shifted to refining the current solution towards a local optima [21].

Tabu search is a metaheuristic that has been applied to a wide range of applications to include scheduling problems, the traveling salesman problem, and graph coloring problems. Tabu search uses a memory-based structure to move from solution to solution in an attempt to reach a global optimum [10]. Memory is primarily used in two different ways. The first is to keep track of previous solutions visited and make them tabu for the next k iterations, meaning they cannot be revisited. The goal of this restriction is to allow moves away from local optima while still moving to high quality solutions at each step [10]. This short-term memory is also used to prevent cycling, wherein the algorithm would move away from a solution and immediately return to it in the next iteration. The second type of memory is long-term based and is used to periodically return to the best solution found so far, or move to a solution

or set of the strategy space that has not been visited yet. This effective use of memory is a key attribute of tabu search that allows the heuristic to explore diverse portions of the strategy space while attempting to avoid traps in local optima.

III. Model Formulation

3.1 Problem Formulation

We consider a game consisting of a set N of cities each with some value $v_j \in \mathbb{R}^+, j \in N$, a set T of Interceptor Missile (IM) types with m_q SAM batteries of type $q \in T$, and a total of n Attacker Missiles (AM). Each SAM battery with type q missiles has defining characteristics such as the probability of successfully intercepting an incoming attacker missile and the maximum coverage radius of the battery that distinguish it from other missile types. We formulate a two-player, three-stage, perfect information, sequential move, zero-sum game as follows. The defender first locates all SAM batteries among a set F of possible locations. The attacker observes this action and then decides on a single stage attack plan. The defender accurately detects this attack upon initiation and launches interceptor missiles to protect some subset of the $|N|$ cities. The strategies we wish to find are (a) where the defender locates its SAM batteries in the first stage, (b) how the attacker allocates its n missiles in the attack plan in the second stage, and (c) how the defender launches interceptor missiles in the final stage.

3.2 Model Assumptions

There are several simplifying assumptions that allow us to formulate our two models. The first assumption is that all parameters are common knowledge. That is, both players know how many IMs and of what type the defender has, and how many AMs the attacker has. Both players also know the coverage radius of each type of SAM battery and the probability that a single IM of type q successfully intercepts an AM. Given the current state of technology and rapid communication capabilities, this is a reasonable assumption that allows for a complete information

game. In addition, we assume that all targets are valued equally by both the defender and the attacker, a necessary assumption to model the game as zero-sum. We also assume that an unintercepted AM will destroy a city with 100% probability. This is a simplifying assumption that models the defender's utility in a worst case scenario. This assumption also allows us to rule out certain attacker strategies, described further in the following chapter.

There are several general assumptions made about the attacker, the first being that all AMs are launched in a single salvo. This assumption is made to reduce the strategy space of the attacker and represents the case of an attack designed to overwhelm air defenses and prevent the resupply of air defense batteries with additional interceptor missiles. We also assume that the attacker has perfect information regarding the location of all SAM batteries placed by the defender in the first stage, as well as the types of IMs at each SAM battery. This is an additional requirement for the complete information game assumption. Lastly, we assume that all AMs are identical, implying that they have the same flight performance and destructive capabilities.

With regard to the defender, we assume that SAM batteries can only be placed at predesignated locations, and at most one SAM battery can be placed at each given location. Again, this is a simplifying assumption made to reduce the size of the strategy space available. We allow the defender to have multiple different types of IMs, under the assumption that a SAM battery contains only one type of IM. In doing so, we also assume that each IM has a fixed probability of intercepting an AM; and we note that this probability need not be the same for all IM types. Since each SAM battery contains only one type of missile, each SAM battery has a fixed coverage radius but this radius may differ based on the IM type at the battery. The final defender assumption is that no more than one IM will be launched against each incoming AM. The Patriot PAC-3 and the THAAD missile systems previously

mentioned are designed to intercept the attacker missile in the terminal phase of flight, meaning there is a small time window remaining to intercept the missile, and if an interceptor fails the defender will not have time to launch a second IM. In the second model formulation, a further defender assumption is made which we discuss prior to presenting the model.

3.3 Model Formulation

The following section presents the model decision variables and parameters, followed by a discussion about the model formulation.

Sets:

- N : the set of all cities.
- F : the set of possible SAM sites.
- T : the set of IM types.

Parameters:

- v_j : the value of city $j \in N$.
- p_q : the probability that a single IM of type $q \in T$ successfully intercepts a single AM.
- m_q : the number of SAM batteries with IMs of type $q \in T$ available to the defender.
- n : the total number of AMs available to the attacker.
- c_q : the number of IMs available per SAM battery with missiles of type $q \in T$.
- A^q : a coverage matrix for a SAM battery with missiles of type $q \in T$, indicating whether a SAM battery at location $i \in F$ is capable of protecting city $j \in N$.

Decision Variables:

- d_i^q : 1 if a SAM battery with IMs of type $q \in T$ is placed at location $i \in F$, and 0 otherwise.
- w_j : Attacker's allocation of AMs to city $j \in N$.
- y_j : 1 if city $j \in N$ is protected, and 0 otherwise.
- x_{ij}^q : Defender's allocation of missiles of type $q \in T$ from a SAM battery placed at location $i \in F$ to intercept an AM launched against city $j \in N$.

Trilevel DAD Model 1:

$$\max_{d} \min_{w} \max_{x,y} \sum_{j \in N} v_j y_j \prod_{q \in T} p_q^{\left(\sum_{i \in F} x_{ij}^q \right)} \tag{1a}$$

$$\text{s.t.} \sum_{i \in F} d_i^q = m_q, \qquad \forall q \in T, \tag{1b}$$

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

$$\sum_{j \in N} x_{ij}^q \leq c_q d_i^q, \qquad \forall i \in F,\, q \in T, \tag{1d}$$

$$\sum_{i \in F} \sum_{q \in T} a_{ij}^q x_{ij}^q \leq w_j, \qquad \forall j \in N, \tag{1e}$$

$$\sum_{i \in F} \sum_{q \in T} a_{ij}^q x_{ij}^q \geq w_j y_j, \qquad \forall j \in N, \tag{1f}$$

$$\sum_{q \in T} d_i^q \leq 1, \qquad \forall i \in F, \tag{1g}$$

$$d_i^q \in \{0, 1\}, \qquad \forall i \in F,\, q \in T, \tag{1h}$$

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

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

$$x_{ij}^q \in \mathbb{Z}^+, \qquad \forall i \in F,\, j \in N,\, q \in T. \tag{1k}$$

The objective function (1a) represents the two-player, three stage optimization of the expected value of the surviving cities, which the defender seeks to maximize via SAM battery location, antecedent to the attacker seeking to minimize it by launching AMs, which precedes the defender's maximizing response of IM interception of incoming AMs. Constraint (1b) requires that all SAM batteries with missiles of type q are employed by the defender, and Constraint (1c) forces the attacker to use all AMs available. Constraint (1d) requires that the defender launch no more than the

total number of IMs available from each location $i \in F$. Constraint (1e) restricts the defender to launch no more than one IM per incoming AM, whereas Constraint (1f) requires that, in order for a city to be protected, the defender must launch an IM against each AM targeting that city. Constraint (1g) enforces that no more than one SAM battery will be placed at any location. Constraints (1h) and (1i) enforce binary integer restrictions, respectively, on the defender's decision to place a SAM battery at each location and the decision to protect a city, and Constraints (1j) and (1k) enforce the integral restrictions on the AM and IM allocations, respectively.

Depending upon how the attacker allocates its n missiles, there are three main possibilities a city can encounter at the beginning of the third stage. The first simply occurs when there are no incoming AMs to city j, that is $w_j = 0$. In this case, Constraint (1f) allows for the city to be protected ($y_j = 1$). Since the right hand side of Constraint (1f) equals zero, this implies that $\sum_{i \in F} \sum_{q \in T} a_{ij}^q x_{ij}^q = 0$ and equivalently that $\sum_{i \in F} x_{ij}^q = 0$, resulting in the full value of the city being added to the objective function (1a). The second situation occurs when the attacker launches AM(s) at a city ($w_j > 0$) and the defender decides *not* to protect the city. In this situation y_j equals zero, and since we assume that an unintercepted AM destroys a city with 100% probability, no value is added to the defender's expected utility (i.e., the city is lost). The final situation occurs when there are incoming AM(s) ($w_j > 0$) and the defender allocates enough IMs to protect the city ($\sum_{i \in F} \sum_{q \in T} x_{ij}^q = w_j$). In this situation, the expected value of the city depends on the number of each type of IM expended.

A feature of this model formulation is that the expected value of a city is a function of the type of IMs used to protect the city and the number of each type of IM used. For example, if there are two IM types (i.e., IM1 and IM2) and a city has five AMs targeting it, then a total of five IMs must be expended to protect the city. These IMs may also have different p_q-values, so using three of IM1 and two of IM2 will result

in a different expected value than when defending the city using two of IM1 and three of IM2. The second model described below reduces this complexity by adding a simplifying assumption.

3.4 Alternate Formulation

For this model, we assume that at most one type of interceptor missile may be used to protect a city. This assumption is made in an attempt to reduce the IM combinations available to the defender. Without this assumption, the defender not only has to decide to protect a city, but also must decide how many of each type of missile to use when protecting a city. This assumption constrains the defender, and it provides a lower bound on the optimal objective function value to the original problem. Lastly, this assumption serves to deconflict potential complications for the defender where SAM batteries from different locations have to communicate about which AM each type of IM is targeting. A major benefit of adding this assumption is it allows us to formulate the model as an integer linear program in the final stage, enabling a wider variety of solution techniques to be leveraged. In adding this assumption, we introduce a new binary decision variable $\psi_j^q \in \{0, 1\}$ indicating whether city $j \in N$ is protected using IMs of type $q \in T$. This variable equals 1 if true, 0 if false. In this model, the previous parameters, sets, and decision variables remain the same as those discussed above.

Trilevel DAD Model 2:

$$\max_{d} \min_{w} \max_{x,\psi} \sum_{j \in N} v_j \sum_{q \in T} \psi_j^q p_q^{w_j} \tag{2a}$$

$$\text{s.t.} \sum_{i \in F} d_i^q = m_q, \qquad \forall q \in T, \tag{2b}$$

$$\sum_{j \in N} w_j = n, \tag{2c}$$

$$\sum_{j \in N} x_{ij}^q \leq c_q d_i^q, \qquad \forall i \in F,\, q \in T, \tag{2d}$$

$$\sum_{i \in F} a_{ij}^q x_{ij}^q \leq w_j \psi_j^q, \qquad \forall j \in N,\, q \in T, \tag{2e}$$

$$\sum_{i \in F} a_{ij}^q x_{ij}^q \geq w_j \psi_j^q, \qquad \forall j \in N,\, q \in T, \tag{2f}$$

$$\sum_{q \in T} d_i^q \leq 1, \qquad \forall i \in F, \tag{2g}$$

$$\sum_{q \in T} \psi_j^q \leq 1, \qquad \forall j \in N, \tag{2h}$$

$$\psi_j^q \in \{0,1\}, \qquad \forall j \in N,\, q \in T, \tag{2i}$$

$$d_i^q \in \{0,1\}, \qquad \forall i \in F,\, q \in T, \tag{2j}$$

$$w_j \in \mathbb{Z}^+, \qquad \forall j \in N, \tag{2k}$$

$$x_{ij}^q \in \mathbb{Z}^+, \qquad \forall i \in F,\, j \in N,\; q \in T. \tag{2l}$$

While a majority of the formulation remains the same as DAD Model 1, there are three main changes. The first is from Constraint (1e) to Constraint (2e). This constraint now enforces that IMs of type q can only be used to defend city j if the binary decision ψ_j^q equals one. A new Constraint (2h) is also introduced that limits at most one IM type to be used to protect a city. Constraint (2h) is known as a

special ordered set of type 1 (i.e., SOS1), which refers to a set of variables in which at most one variable can take a strictly positive value [14]. In DAD Model 2, SOS1 enforces the assumption that at most one missile type may be used to defend a city, so at most one ψ_j^q variable can be positive for each $j \in N$.

The other major change to this model is the refinement of the objective function. In Model 2, we now have an objective function that, for a fixed attacker strategy $w_j = \tilde{w}_j,\ \forall j \in N$, the resulting formulation is an integer program having a linear relaxation, compared to the original formulation for which the corresponding relaxation was non-linear and non-convex. This results from the added assumption because we now know that, if there are five incoming AMs to a city and that city is protected, then five IMs of the same type are used to protect that city. This is true for all cities, allowing us to replace the $\sum_{i \in F} x_{ij}^q$ in the exponent of the objective function (1a) with w_j, substitute ψ_j^q for y_j as appropriate, and replace the production summation with a summation, resulting in a linear objective function in the final stage.

The same previous scenarios again apply to each city upon completion of the second stage. The first two scenarios remain the same, whereas the third only differs slightly. In the new model, the defender does not have to decide how many of each IM type are used to protect a city, only which type of IM is used to protect a city. Again, this is a more constricting limitation for the defender but reduces the combinations of decisions made in the final stage.

IV. Methodology

4.1 Full Enumeration

Focusing on DAD Model 2, for given solutions to the first and second stages, we are left to solve the integer linear program shown below in (3a) - (3g). One approach to solve the DAD problem is to consider all possible defender and attacker strategy combinations and solve the resulting ILP exactly. The difficulty with full enumeration is that it quickly becomes intractable, even for small-size instances. Full enumeration can also be represented in extensive form using a game tree as illustrated in Figure 1.

$$\max_{x,\psi} \sum_{j\in N} v_j \sum_{q\in T} \psi_j^q p_q^{w_j} \tag{3a}$$

$$\text{s.t.} \sum_{j\in N} x_{ij}^q \le c_q d_i^q, \qquad \forall i \in F,\, q \in T, \tag{3b}$$

$$\sum_{i\in F} a_{ij}^q x_{ij}^q \le w_j \psi_j^q, \qquad \forall j \in N,\, q \in T, \tag{3c}$$

$$\sum_{i\in F} a_{ij}^q x_{ij}^q \ge w_j \psi_j^q, \qquad \forall j \in N,\, q \in T, \tag{3d}$$

$$\sum_{q\in T} \psi_j^q \le 1, \qquad \forall j \in N, \tag{3e}$$

$$\psi_j^q \in \{0,1\}, \qquad \forall j \in N,\, q \in T, \tag{3f}$$

$$x_{ij}^q \in \mathbb{Z}^+, \qquad \forall i \in F,\, j \in N,\; q \in T. \tag{3g}$$

If we first restrict the problem to a case containing only one type of IM in a game consisting of $|F|$ possible SAM Locations with m SAM batteries, the defender can allocate these SAM batteries $\binom{|F|}{m}$ different ways. In allowing for SAM batteries with multiple IM types, the strategy space for the defender begins to grow as it is now

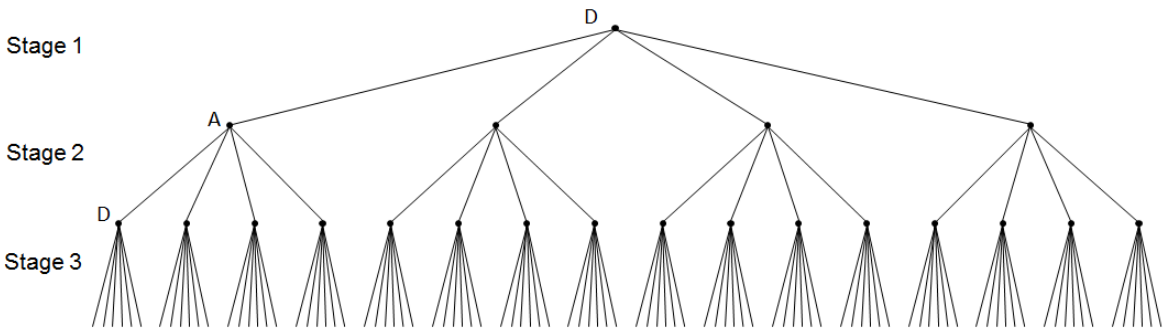


Figure 1. Example Game Tree

important to know not only where SAM batteries are located, but also what IM type is at each location. Examining the case when the defender has m_q SAM batteries with missiles of type q, $m_q \geq 1$ for all $q \in T$, the defender now has $S_d = \prod_{q \in T} \binom{|F| - \sum_{i<q} m_i}{m_q}$ strategies to consider. For example, if the defender has $|T| = 2$ IM types in which $m_1 = m_2 = 2$ SAM batteries corresponding to each missile type, for an instance with $|F| = 6$ possible locations, the defender can allocate these 4 total SAM batteries $\binom{6}{2}\binom{6-2}{2} = 90$ different ways.

Meanwhile, the attacker must decide on a missile allocation strategy $\tilde{w}_j$ to target each city. In stage 2, the attacker can allocate these n AMs $S_a = \binom{|N|+n-1}{n}$ different ways. This is also a combinatorial problem that grows both as the number of cities $|N|$ increases and the number of AMs n increases. Considering a small $|N| = 6$ city example again with $n = 5$ AMs, the attacker has $\binom{6+5-1}{5} = 252$ strategies available.

Using full enumeration there are $S_d \cdot S_a$ branches at the end of stage 2 which requires the same number of integer linear programs be solved. Table 1 shows how the number of stage 2 branches changes on a small instance where $|T| = 2$ and $|N| = |F| = 6$ for increasing values of m_1, m_2, and n. The largest example given in Table 1 occurs when $m_1 = m_2 = 2$ and $n = 4$, and even this relatively small instance requires over 11,000 ILPs be solved. Table 1 helps illustrate how the strategy space grows exponentially for increasing parameters of the problem, and thus why full

enumeration is not suitable for larger sized instances.

Table 1. Strategy Space for $|N| = |F| = 6$ example

m_1	m_2	n	S_d	S_a	Stage 2 Branches
1	1		30		630
2	1	2	60	21	1,260
2	2		90		1,890
1	1		30		1,680
2	1	3	60	56	3,360
2	2		90		5,040
1	1		30		3,780
2	1	4	60	126	7,560
2	2		90		11,340

4.2 Reduced Strategy Space

When implementing full enumeration we consider every possible defender and attacker strategy. However, based on certain parameters and at different stages of the game, we can start to rule out strategies to reduce the size of the strategy space. At the beginning of the game we are able to utilize the coverage matrices A^q to eliminate certain defender strategies. Recall that a SAM battery at location i is dominated by a SAM battery at location k if location k covers every city covered by location i while covering at least one additional city. In allowing multiple IM types we can no longer say that a SAM battery will never be placed at location i. For example, the defender may want to place a SAM battery with IM Type 2 at location k and a SAM battery with IM Type 1 at location i. What can be observed is that the defender will never place a SAM battery at location i *if* there is no SAM battery at location k. In this case the defender could move the SAM battery from location i to location k and not lose any coverage while covering at least one additional city. Utilizing this result, we can check all possible defender SAM allocation strategies to

see if this relation is true, and if so we do not need to explore this portion of the strategy space.

At the beginning of stage 2 we can leverage the fact that the defender has already placed all SAM batteries to rule out certain attacker strategies. We assume this is a perfect information game, so upon completion of stage 1 the attacker should know the maximum number of IMs of each type that can be used to protect any given city. Let θ_j^q represent the number of IMs of type q that can be used to protect city j, and $\eta_j = \max_q \theta_j^q$ be the maximum number of IMs of all types that can be used to protect city j. Once this is known, the attacker should not consider a strategy that allocates $w_j \geq \eta_j + 2$ AMs to city j. We can make this statement due to the assumption that any AM not intercepted destroys a city with 100% probability. So if the attacker wants to overwhelm a city and destroy it with certainty, this can be achieved with $w_j = \eta_j + 1$ AMs. Any additional AMs targeting this city would be "wasted" and are better utilized elsewhere. Again, we can check this relation for each attacker strategy given the current defender strategy and, if any $w_j \geq \eta_j + 2$, the resulting ILP does not need to be solved. There is a slight complication when the attacker can overwhelm a majority of cities regardless of the defender's SAM allocation, however the expression of Constraint (1c) and (2c) as an equality presumes that the attacker does not have more than the number of AMs required to both overwhelm the defender's IMs *and* destroy all uncovered cities. For such an instance, we would express Constraint (1c) and (2c) as an inequality. Herein, we set aside instances manifesting such an overmatch of capabilities and retain the form expressed above.

4.3 Double Oracle

Double Oracle is implemented by iteratively solving three main problems until convergence. The first component is *coreLP* which solves via enumeration a restricted version of the game having a limited set of strategies available to the defender and the attacker. Let S_d^k and S_a^k represent the set of strategies available to the defender and the attacker during the k^{th} iteration, respectively. The *coreLP* routine solves the problem formulated in (3a) - (3g) to find the SPNE for this restricted game. S_d^0 and S_a^0 are initialized by selecting an arbitrary set of defender and attacker strategies; however, initializing these sets with "smart" strategies may improve the performance of the algorithm as discussed by Jain et al. [19]. The defender and attacker SPNE strategies returned from *coreLP* are denoted $\hat{s}_d$ and $\hat{s}_a$ respectively.

Defender Oracle

The Defender Oracle subproblem solves for the defender's best response for a fixed AM allocation $\hat{s}_a$. This problem is also formulated in (4a) - (4j). In Defender Oracle the defender wants to find the best SAM allocation strategy in stage 1 and IM allocation in stage 3 to maximize the expected value of surviving cities. Since the attacker's strategy is fixed in this problem, Defender Oracle is an integer linear program which can be solved exactly. The defender's best response is then used to update the set of strategies currently available. Let $\bar{s}_d$ represent the defender's best response found using Defender Oracle. Upon completion of iteration k, we let $S_d^k \leftarrow S_d^{k-1} \bigcup \bar{s}_d$.

$$\max_{d,x,\psi} \sum_{j\in N} v_j \sum_{q\in T} \psi_j^q p_q^{w_j} \tag{4a}$$

$$\text{s.t.} \sum_{i\in F} d_i^q = m_q, \qquad \forall q \in T, \tag{4b}$$

$$\sum_{j\in N} x_{ij}^q \le c_q d_i^q, \qquad \forall i \in F,\, q \in T, \tag{4c}$$

$$\sum_{i\in F} a_{ij}^q x_{ij}^q \le w_j \psi_j^q, \qquad \forall j \in N,\, q \in T, \tag{4d}$$

$$\sum_{i\in F} a_{ij}^q x_{ij}^q \ge w_j \psi_j^q, \qquad \forall j \in N,\, q \in T, \tag{4e}$$

$$\sum_{q\in T} d_i^q \le 1, \qquad \forall i \in F, \tag{4f}$$

$$\sum_{q\in T} \psi_j^q \le 1, \qquad \forall j \in N, \tag{4g}$$

$$\psi_j^q \in \{0,1\}, \qquad \forall j \in N,\, q \in T, \tag{4h}$$

$$d_i^q \in \{0,1\}, \qquad \forall i \in F,\, q \in T, \tag{4i}$$

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

Attacker Oracle

The Attacker Oracle subproblem formulated in (5a) - (5i) is used to solve for the attacker's best response for a fixed defender SAM allocation strategy $\hat{s}_d$. This is a bilevel nonlinear integer formulation as the defender's SAM locations are fixed, but the defender has yet to decide which IMs to launch to protect the respective cities. We utilize two different heuristics described further below to solve for the attacker's best response $\bar{s}_a$, upon which we similarly let $S_a^k \leftarrow S_a^{k-1} \bigcup \bar{s}_a$.

$$\min_{w} \max_{x,\psi} \sum_{j \in N} v_j \sum_{q \in T} \psi_j^q p_q^{w_j} \tag{5a}$$

$$\text{s.t.} \sum_{j \in N} w_j = n, \tag{5b}$$

$$\sum_{j \in N} x_{ij}^q \leq c_q d_i^q, \qquad \forall i \in F,\, q \in T, \tag{5c}$$

$$\sum_{i \in F} a_{ij}^q x_{ij}^q \leq w_j \psi_j^q, \qquad \forall j \in N,\, q \in T, \tag{5d}$$

$$\sum_{i \in F} a_{ij}^q x_{ij}^q \geq w_j \psi_j^q, \qquad \forall j \in N,\, q \in T, \tag{5e}$$

$$\sum_{q \in T} \psi_j^q \leq 1, \qquad \forall j \in N, \tag{5f}$$

$$\psi_j^q \in \{0, 1\}, \qquad \forall j \in N,\, q \in T, \tag{5g}$$

$$w_j \in \mathbb{Z}^+, \qquad \forall j \in N, \tag{5h}$$

$$x_{ij}^q \in \mathbb{Z}^+, \qquad \forall i \in F,\, j \in N,\; q \in T. \tag{5i}$$

The first heuristic implemented to solve Attacker Oracle is simulated annealing. The heuristic starts with the initial feasible strategy $\hat{s}_a$ from *coreLP* and utilizes a neighborhood search technique to explore the strategy space while iteratively updating the current attacker strategy, denoted $\ddot{s}_a$. The neighborhood definition is influenced by that used by Han et al. [16] in examining a related problem assuming only one type of IM is available to the defender. The set of neighbor strategies is generated by moving one AM from city j, $w_j > 0$, to some other city j', $j \neq j'$. In a three city example, if the current strategy is $\ddot{s}_a = [3\ 5\ 0]$, the neighbor strategies would be

$$\text{neighbors}([3\ 5\ 0]) = \begin{bmatrix} 2 & 6 & 0 \\ 2 & 5 & 1 \\ 4 & 4 & 0 \\ 3 & 4 & 1 \end{bmatrix}.$$

Instead of evaluating the entire neighborhood, a single random neighbor is selected to compare against the current strategy. If the neighbor results in a lower expected value of surviving cities, then $\ddot{s}_a$ is updated. If the neighbor strategy results in an expected survival value greater than the current strategy, then it is probabilistically selected to replace the current solution based on the difference in objective function values and the current temperature of the system. The starting temperature is set very high, so there is initially a high probability that worse strategies are adopted and the solution space can be explored, while over time the temperature slowly cools so the heuristic can converge to a local optimum. The heuristic continues until the temperature is sufficiently small, at which point the best strategy found is returned as $\bar{s}_a$.

The second heuristic implemented is tabu search which also utilizes the same neighborhood definition. Compared to simulated annealing, tabu search evaluates each neighbor strategy for the expected city survival value based on the defender's best response in stage 3. The neighborhood strategy that results in the lowest expected value is selected to update the current strategy, regardless of its relation to the previous strategy. This implies that we allow moves to strategies that increase the value of surviving cities (i.e., worse for the attacker) in an attempt to escape local optima. The current strategy $\ddot{s}_a$ is then placed on the tabu list for the next l iterations, which prevents us from moving away from one strategy and returning to it in the following iterations. The heuristic terminates after a pre-determined number

of iterations at which point the best strategy found is returned as $\bar{s}_a$.

If no new defender and attacker strategies are added, meaning $S_d^k = S_d^{k-1}$ *and* $S_a^k = S_a^{k-1}$, then the algorithm has converged and $\hat{s}_d$ and $\hat{s}_a$ are accepted as the predicted SPNE strategies to the full game. If new strategies are added, the algorithm repeats, starting with *coreLP* and solving for new best response strategies.

Since simulated annealing and tabu search are both heuristic approaches to solve Attacker Oracle, there is no guarantee that they will return the optimal attacker strategy $\bar{s}_a$. Even so, both heuristics have unique features that can be utilized to ensure high quality solutions are still found. While McMahan et al. [23] proved that Double Oracle will converge to the optimal solution when mixed strategies are considered, we only consider pure strategies. In the context of this game, the issue is not necessarily that we do not consider mixed strategies, rather that we do not solve for alternative optima, if they exist. This implies that even if the heuristics used in the Attacker Oracle are exact, there is still the possibility that Double Oracle will fail to properly identify the SPNE.

V. Computational Results

5.1 6 City Network

To test the performance of the various solution methods, we use the same network Daskin [7] examined in the MEXCLP. The topology of this network along with the value assigned to each city is illustrated in Figure 2. For all instances we test, it is assumed that $|N| = 6$, $|F| = 6$, and $|T| = 2$. Test instances are based on those used by Han et al. [16] in examining a single IM type problem. We first initialize Double Oracle (DO) with a single defender strategy of placing SAM batteries in sequential order at the most valuable city without a SAM battery and single attacker strategy of all AMs targeting the most valuable city.

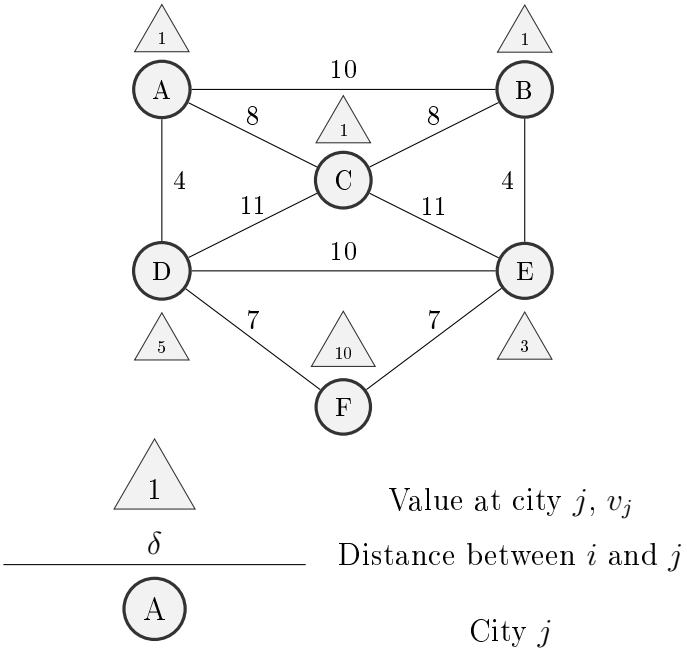


Figure 2. 6-city Network

5.2 Heuristic Parameters

Within DO using Simulated Annealing (DO/SA) in Attacker Oracle, the main algorithmic parameters are the initial temperature t_0, the temperature cooling scheme, and the stopping condition. These parameters were tested on smaller instances to determine which settings would likely yield the desired results. For all final test instances the initial temperature is $t_0 = 10,000$ and the temperature at iteration k is updated according to the function $t_k = 0.95 \cdot t_{k-1}$. In our implementation, we complete 250 outer iterations with 5 iterations at each temperature for a total of 1,250 iterations. This implies that the temperature at the end of the search is near 0.028. We use the same probability function of accepting a worse strategy discussed by Kirkpatrick et al. [21] of $p = e^{\Delta/t_k}$, where Δ represents the difference in objective function values between the current strategy and the neighbor strategy. The stochastic nature of DO/SA results in the possibility that different solutions will be returned each time it is implemented regardless of initialization strategies.

When implementing DO using Tabu Search (DO/TS) in Attacker Oracle, the current attacker strategy will have at most $|N| \cdot (|N| - 1) = 30$ neighbors for this 6 city network, so we evaluate the whole neighborhood in Attacker Oracle. This implies that DO/TS is deterministic in the sense that running the algorithm multiple times will return the same value, assuming the same defender/attacker initialization strategies are used each time. On a larger network, the neighborhood size begins to grow, and a subset of the neighborhood will likely need to be selected. Once a strategy has been selected for adoption, it is placed on the tabu list for the next 3 iterations. Any duration greater than one iteration will be sufficient to prevent us from cycling between strategies, and 3 was selected to meet this requirement. To determine the number of iterations to complete within Attacker Oracle, we first consider the worst case available. In such a scenario the attacker is using a strategy in which all AMs

target the last city when the optimal strategy is to have all AMs target the first city. Neighbor solutions will be generated and evaluated, and one AM will be moved from the last city to the first city each time. After completion of n iterations the best response strategy will be reached. Since Attacker Oracle is solved via a heuristic there is no guarantee this will be the solution to the full game. As such, we may not want to complete all n iterations, especially for large values of n. Instead, we simply want to find a "better" attacker response, similar to Jain et al. [19]. Upon testing the effect different iteration lengths have on solution quality in smaller instances, we fix the number of iterations in Attacker Oracle within DO/TS at 6.

5.3 Test Instance Results

Tables 2 and 3 present the various problem instances we test. In both of these tables, IM Type 1 is identical and intended to represent a "baseline" IM performance. For comparison, IM Type 2 reflects a more capable missile with a higher p_q and coverage radius; however, fewer SAM batteries of this type are available. We further distinguish IM Type 2 in terms of the number of IMs per SAM battery. IM Type 2A reflects a scenario in which the second IM type has an improved performance over the baseline, but fewer IMs are available. Thus, in Table 2 we set $c_2 = \frac{3}{4}c_1$ rounded up to the nearest even number. For example, this situation could be represented by the comparison of the PAC-3 and HIMARS. In contrast, IM Type 2B again represents a more capable missile, but with more IMs available than the baseline. For IM Type 2B we let $c_2 = \frac{5}{4}c_1$ rounded up to the nearest even number as illustrated in Table 3. This relation could be reflected by the comparison of the PAC-3 and THAAD for example.

Tables 2 and 3 also report the computation time required to solve each instance using full enumeration, reduced enumeration, DO/SA, and DO/TS. All instances

were solved using MATLAB calling IBM ILOG CPLEX Optimization Studio version 12.6 on an Intel(R) Xeon E5-1620 3.6 GHz processor having 32 GB memory.

Table 2. Problem Instances IM Type 2A

		IM Type 1				IM Type 2A				Computation Time (sec)			
Instance	n	m_1	c_1	p_1	r_1	m_2	c_2	p_2	r_2	Full	Reduced	DO/SA	DO/TS
1	5	2	12	0.5	6	1	10	0.6	9	15.84	12.41	4.36	0.33
2	10	1	8	0.3	6	1	6	0.4	9	94.98	36.31	8.29	1.37
3	10	1	16	0.3	6	1	12	0.4	9	93.62	34.00	8.05	1.45
4	10	1	8	0.7	6	1	6	0.8	9	95.65	36.94	10.87	1.55
5	10	1	16	0.7	6	1	12	0.8	9	91.63	35.44	7.18	0.75
6	10	3	8	0.3	6	1	6	0.4	9	200.07	157.34	6.50	1.11
7	10	3	16	0.3	6	1	12	0.4	9	187.12	148.42	2.93	0.60
8	10	3	8	0.7	6	1	6	0.8	9	196.53	162.81	8.16	0.73
9	10	3	16	0.7	6	1	12	0.8	9	187.84	148.47	4.37	0.25
10	15	2	12	0.1	6	1	10	0.2	9	1002.22	487.95	14.75	1.93
11	15	2	4	0.5	6	1	4	0.6	9	1140.25	192.52	7.26	1.82
12	15	2	12	0.5	6	1	10	0.6	9	1002.77	481.16	9.67	1.49
13	15	2	20	0.5	6	1	16	0.6	9	979.47	471.74	6.13	1.04
14	15	2	12	0.9	6	1	10	1	9	1006.36	485.23	10.33	0.68
15	15	3	12	0.5	6	2	10	0.6	9	1332.72	1353.93	11.41	1.71
16	20	1	8	0.3	6	1	6	0.4	9	1839.85	172.12	20.65	3.66
17	20	1	16	0.3	6	1	12	0.4	9	1723.53	356.48	9.28	2.12
18	20	1	8	0.7	6	1	6	0.8	9	1876.66	176.08	15.16	2.34
19	20	1	16	0.7	6	1	12	0.8	9	1733.28	346.94	9.62	1.40
20	20	2	8	0.6	6	2	6	0.7	9	13422.77	7701.52	40.63	5.84
21	20	3	8	0.3	6	1	6	0.4	9	4210.00	1986.72	11.35	1.86
22	20	3	16	0.3	6	1	12	0.4	9	4079.74	2421.24	6.64	1.89
23	20	3	8	0.7	6	1	6	0.8	9	4915.87	2108.95	19.66	1.26
24	20	3	16	0.7	6	1	12	0.8	9	3571.13	2433.24	14.79	1.53
25	25	2	12	0.5	6	1	10	0.6	9	10333.44	2986.15	16.29	2.76
26	25	2	12	0.7	6	2	10	0.8	9	31282.68	22650.34	58.93	3.53

Table 3. Problem Instances IM Type 2B

		IM Type 1				IM Type 2B				Computation Time (sec)			
Instance	n	m_1	c_1	p_1	r_1	m_2	c_2	p_2	r_2	Full	Reduced	DO/SA	DO/TS
1	5	2	12	0.5	6	1	16	0.6	9	15.69	12.12	4.14	0.33
2	10	1	8	0.3	6	1	10	0.4	9	92.09	34.14	7.78	1.20
3	10	1	16	0.3	6	1	20	0.4	9	94.17	34.11	7.71	1.62
4	10	1	8	0.7	6	1	10	0.8	9	93.19	33.76	8.32	0.91
5	10	1	16	0.7	6	1	20	0.8	9	92.99	33.83	8.29	0.68
6	10	3	8	0.3	6	1	10	0.4	9	191.76	152.25	2.82	0.56
7	10	3	16	0.3	6	1	20	0.4	9	189.93	150.02	2.82	0.60
8	10	3	8	0.7	6	1	10	0.8	9	190.87	151.26	4.27	0.25
9	10	3	16	0.7	6	1	20	0.8	9	189.93	150.84	4.27	0.25
10	15	2	12	0.1	6	1	16	0.2	9	971.47	470.57	11.24	1.93
11	15	2	4	0.5	6	1	6	0.6	9	1183.04	339.48	20.00	2.05
12	15	2	12	0.5	6	1	16	0.6	9	980.67	469.00	4.69	0.87
13	15	2	20	0.5	6	1	26	0.6	9	979.13	482.78	7.95	0.88
14	15	2	12	0.9	6	1	16	1	9	976.86	471.09	4.20	1.48
15	15	3	12	0.5	6	2	16	0.6	9	997.67	992.51	6.14	0.85
16	20	1	8	0.3	6	1	10	0.4	9	1808.44	325.28	31.59	2.79
17	20	1	16	0.3	6	1	20	0.4	9	1652.89	383.88	21.65	1.99
18	20	1	8	0.7	6	1	10	0.8	9	1826.66	331.70	19.84	2.66
19	20	1	16	0.7	6	1	20	0.8	9	1647.21	385.37	9.04	2.02
20	20	2	8	0.6	6	2	10	0.7	9	8659.12	6921.97	22.59	2.12
21	20	3	8	0.3	6	1	10	0.4	9	4226.24	2719.01	9.14	1.79
22	20	3	16	0.3	6	1	20	0.4	9	3404.64	2623.69	14.17	1.58
23	20	3	8	0.7	6	1	10	0.8	9	4219.76	2654.47	16.30	2.13
24	20	3	16	0.7	6	1	20	0.8	9	3403.84	2706.83	12.99	0.52
25	25	2	12	0.5	6	1	16	0.6	9	9716.15	3820.85	9.42	2.75
26	25	2	12	0.7	6	2	16	0.8	9	19993.36	15858.60	32.85	1.33

As expected, full enumeration takes a considerable amount of time to solve making the method impractical for even moderately-sized instances. Instance 26, the largest size instance in which $S_d \cdot S_a = \binom{6}{2}\binom{6-2}{2} \cdot \binom{6+25-1}{25} = 12,825,540$, required just over 8.6 hours and 5.5 hours to evaluate the game tree and identify the SPNE for IM Types 2A and 2B respectively. Although full enumeration is computationally expensive, it does explore the whole game tree, so we are guaranteed to find the equilibrium.

In all but one instance (instance 15 for IM Type 2A), reduced enumeration was able to reduce the computation time, on average approximately 50% faster than full enumeration for IM Type 2A and approximately 45% faster for IM Type 2B, but it still required up to 6.3 hours to complete. This method again explores the game tree similar to full enumeration, with the added benefit in that it does not evaluate any strategy combination in which either the defender or attacker are using a dominated strategy. While reduced enumeration is guaranteed to find the SPNE with a potential reduction in computation time, it still remains impractical as a method for larger instances.

Tables 4 and 5 provide the expected value of surviving cities for each instance when implementing DO/SA, and Tables 7 and 8 provide the corresponding results when implementing DO/TS. Since reduced enumeration is an exact method implemented primarily to compare the solution time with full enumeration, it has been omitted from Tables 4 - 8. Each of the tables also compare the defender and attacker strategy found using Double Oracle with the optimal strategy identified using full enumeration, as reported in columns 6 and 7. If the strategies identified by full enumeration and Double Oracle are the same, the column contains a 1; else the identified solutions differ and the entry is a 0. These columns provide potential indications of why Double Oracle failed to identify the SPNE or, in some instances, identify the presence of alternative optima.

Table 4. DO/SA Expected Survival Value IM Type 2A

Instance	Value				DO/SA strategy comparison with Full	
	Full	DO/SA	Relative Gap SA	Absolute Gap SA	Defender Strategy	Attacker Strategy
1	9.960	**9.960**			0	0
2	2.270	**2.270**			1	1
3	2.410	**2.410**			0	1
4	7.400	**7.400**			0	1
5	9.921	9.821	-1.01%	-0.100	0	0
6	2.710	2.610	-3.69%	-0.100	0	1
7	2.820	**2.820**			1	1
8	8.924	8.711	-2.39%	-0.213	0	0
9	10.581	**10.581**			1	1
10	0.150	0.146	-2.67%	-0.004	0	0
11	2.046	2.023	-1.12%	-0.023	0	0
12	3.192	**3.192**			1	0
13	3.401	**3.401**			1	1
14	10.430	11.000	5.47%	0.570	0	0
15	3.874	**3.874**			0	1
16	0.179	0.116	-35.33%	-0.063	0	0
17	0.274	0.188	-31.55%	-0.087	0	0
18	2.640	2.420	-8.33%	-0.220	0	0
19	3.883	**3.883**			0	1
20	3.317	3.312	-0.15%	-0.005	0	0
21	0.374	**0.374**			1	0
22	0.463	0.462	-0.30%	-0.001	0	0
23	4.345	3.984	-8.31%	-0.361	0	0
24	5.303	**5.303**			1	1
25	1.026	0.918	-10.55%	-0.108	0	0
26	4.728	4.730	0.05%	0.002	0	0

Note: Text in bold indicates DO reached the SPNE

In terms of required computation time, DO/SA offers a significant improvement over full enumeration. The average time required for DO/SA to solve the 26 test instances is approximately 13.2 seconds and 12.7 seconds for IM Type 2A and IM Type 2B, respectively, with the longest time just under 60 seconds. Whereas DO/SA is significantly faster than full enumeration, there are several instances for which it

Table 5. DO/SA Expected Survival Value IM Type 2B

Instance	Value				DO/SA strategy comparison with Full	
	Full	DO/SA	Relative Gap SA	Absolute Gap SA	Defender Strategy	Attacker Strategy
1	9.960	**9.960**			0	0
2	2.410	**2.410**			0	1
3	2.410	**2.410**			0	1
4	9.921	9.821	-1.01%	-0.100	0	0
5	9.921	9.821	-1.01%	-0.100	0	0
6	2.820	**2.820**			1	1
7	2.820	**2.820**			1	1
8	10.581	**10.581**			1	1
9	10.581	**10.581**			1	1
10	0.150	0.138	-8.00%	-0.012	0	0
11	2.521	**2.521**			1	1
12	3.401	**3.401**			1	1
13	3.401	**3.401**			1	1
14	18.900	14.900	-21.16%	-4.000	0	0
15	3.874	**3.874**			0	1
16	0.274	**0.274**			1	1
17	0.326	**0.326**			1	1
18	3.882	3.756	-3.24%	-0.126	0	0
19	5.373	5.209	-3.05%	-0.164	0	0
20	4.132	4.137	0.12%	0.005	1	0
21	0.461	0.462	0.08%	$<$0.001	1	0
22	0.500	**0.500**			1	1
23	5.295	5.197	-1.86%	-0.098	0	0
24	6.582	**6.582**			1	0
25	1.196	**1.196**			1	1
26	5.381	5.386	0.09%	0.005	0	0

Note: Text in bold indicates DO reached the SPNE

fails to identify the SPNE. Tables 4 and 5 provide the optimality gap for the instances in which Double Oracle failed to properly identify the SPNE, and Table 6 provides selected aggregate performance measures for the instances where it failed (across all 52 total instances). Table 6 shows that simulated annealing fails to identify the SPNE more often below the optimal than above. DO/SA has a maximum optimality gap of

35.33% when it fails below the SPNE, but is only 5.47% when it fails above the SPNE. The average difference in values with full enumeration is 0.13 for IM Type 2A and 0.46 for IM Type 2B, with the largest difference in values at 4.00 for instance 14 on IM Type 2B. Further, the instance with the largest relative optimality gap does not necessarily correspond to the instance with the largest absolute gap. It is important again to note that, due to the stochastic nature of simulated annealing as a heuristic, running DO/SA multiple times will most likely return different values. Thus, we could run DO/SA again and might reach the SPNE in more (or fewer) instances, however the results provided are from a single run through all the instances.

Table 6. DO Performance with Single Strategy Initialization

	DO/SA		DO/TS	
	Below SPNE	Above SPNE	Below SPNE	Above SPNE
# of instances	19	5	19	4
Average %	-7.62%	1.16%	-13.13%	1.55%
Max %	-35.33%	5.47%	-41.00%	3.97%
Average Value Diff	-0.310	0.117	-0.264	0.080
Max Value Diff	-4.00	0.570	-0.960	0.172

DO/TS again offers a significant improvement in computation time over full enumeration while also performing faster than DO/SA. Compared to full enumeration which required up to 8.6 hours to solve, DO/TS required a maximum of 5.84 seconds before completion. For IM Type 2A, DO/TS terminated in an average of 1.73 seconds, whereas for IM Type 2B, DO/TS terminated in an average of 1.39 seconds. While DO/TS is significantly faster, there are again instances where it fails to identify the SPNE. Tables 7 and 8 again report the relative and absolute gap for these instances. For IM Type 2A, DO/TS reached the optimal objective function value in 46% of the instances, while IM Type 2B was slightly higher at 65% of the instances. The optimality gap for instances where DO/TS did fail is as high as 41% in multiple

instances, however the optimal SPNE value for these instances are also very small so even small deviations will appear as large relative deviations. For example, instance 16 and 17 for IM Type 2B both have relative optimality gaps above 40%, but the absolute deviation is only 0.112 and 0.134, respectively. For all 52 test instances, the maximum absolute deviation from the SPNE value is 0.960 with an average deviation of approximately 0.264.

Table 6 again illustrates that, in the instances when DO/TS did fail to properly find the SPNE, it tended to return an expected value lower than the optimal value. Outcomes such as these indicate that Double Oracle converged prematurely and portions of the game tree where the optimal strategies exist were never explored. In every instance for which Double Oracle returned a smaller value than the SPNE, the defender strategy returned failed to match that of full enumeration. Conversely, when Double Oracle returned a larger expected value, the conjecture is that Attacker Oracle failed to generate the correct strategy. This is indeed a likely cause since we use a heuristic to solve Attacker Oracle, but it cannot be stated with certainty in the absence of further information.

Table 7 shows there are two instances where Double Oracle failed and the attacker strategy matched that of full enumeration (i.e., instances 6 and 16). These occurrences indicate that Double Oracle failed due to only considering pure strategies. Due to the current formulation, in order for the defender to have a mixed strategy, the strategies in the support must result in the same optimal SPNE value. Otherwise, the defender would always select the strategy that resulted in the larger value. Thus, the cause of this can also be attributed to not solving for alternative optima, if they exist. Lastly of interest are those instances for which Double Oracle returned the same value as full enumeration, and either the defender or attacker strategy differ (e.g., instances 1 and 12). These occurrences indicate the presence of alternative optima.

Table 7. DO/TS Expected Survival Value IM Type 2A

Instance	Value		Relative Gap TS	Absolute Gap TS	DO/TS strategy comparison with Full	
	Full	DO/TS			Defender Strategy	Attacker Strategy
1	9.960	**9.960**			0	1
2	2.270	**2.270**			0	1
3	2.410	1.450	-39.83%	-0.960	0	0
4	7.400	**7.400**			0	1
5	9.921	9.821	-1.01%	-0.100	0	0
6	2.710	2.610	-3.69%	-0.100	0	1
7	2.820	**2.820**			1	1
8	8.924	8.711	-2.39%	-0.213	0	0
9	10.581	**10.581**			1	1
10	0.150	**0.150**			1	1
11	2.046	2.023	-1.12%	-0.023	0	0
12	3.192	**3.192**			1	0
13	3.401	**3.401**			1	1
14	10.430	9.710	-6.90%	-0.720	0	0
15	3.874	**3.874**			0	1
16	0.179	0.170	-5.24%	-0.009	0	1
17	0.274	0.271	-1.13%	-0.003	0	0
18	2.640	2.420	-8.33%	-0.220	0	0
19	3.883	**3.883**			0	1
20	3.317	3.250	-2.02%	-0.067	0	0
21	0.374	**0.374**			1	1
22	0.463	0.461	-0.38%	-0.002	0	0
23	4.345	4.517	3.97%	0.172	0	0
24	5.303	**5.303**			1	1
25	1.026	0.918	-10.55%	-0.108	0	0
26	4.728	4.812	1.78%	0.084	1	0

Note: Text in bold indicates DO reached the SPNE

Of the 52 total instances examined, tabu search is faster than simulated annealing in each of them. On average tabu search is 11.47 seconds faster than simulated annealing for IM Type 2A and 10.31 seconds faster for IM Type 2B, with the maximum difference in time around 55 seconds. Within Attacker Oracle, simulated annealing does complete more iterations than tabu search due to the random nature of ex-

Table 8. DO/TS Expected Survival Value IM Type 2B

Instance	Value		Relative Gap TS	Absolute Gap TS	DO/TS strategy comparison with Full	
	Full	DO/TS			Defender Strategy	Attacker Strategy
1	9.960	**9.960**			0	1
2	2.410	1.450	-39.83%	-0.960	0	0
3	2.410	1.450	-39.83%	-0.960	0	0
4	9.921	9.821	-1.01%	-0.100	0	0
5	9.921	9.821	-1.01%	-0.100	0	0
6	2.820	**2.820**			1	1
7	2.820	**2.820**			1	1
8	10.581	**10.581**			1	1
9	10.581	**10.581**			1	1
10	0.150	**0.150**			1	1
11	2.521	**2.521**			1	1
12	3.401	**3.401**			1	1
13	3.401	**3.401**			1	1
14	18.900	18.957	0.30%	0.057	0	0
15	3.874	**3.874**			0	1
16	0.274	0.162	-40.88%	-0.112	0	0
17	0.326	0.192	-41.00%	-0.134	0	0
18	3.882	3.756	-3.24%	-0.126	0	0
19	5.373	**5.373**			0	1
20	4.132	**4.132**			1	1
21	0.461	**0.461**			1	1
22	0.500	**0.500**			1	1
23	5.295	5.303	0.15%	0.008	0	0
24	6.582	**6.582**			1	1
25	1.196	**1.196**			1	1
26	5.381	**5.381**			0	1

Note: Text in bold indicates DO reached the SPNE

ploring neighbors which explains one reason for the difference in computation times. While this comparison helps illustrate the difference in computation time required by the two heuristics, a direct comparison is not entirely accurate. A more comparable measure would also include the number of iterations Double Oracle performed before converging, as this may also account for the difference in computation time. Table

6 also shows that simulated annealing fails in one more instance above the SPNE but in the same number of instances below the SPNE, compared to tabu search. While DO/TS results in a larger average optimality gap both below and above the SPNE than DO/SA, the average absolute optimality gap and the maximum absolute optimality gap are both lower. As Tables 7 and 8 illustrate this is due to DO/TS failing most often when the SPNE is ≤ 3, where deviations will result in a large relative optimality gap but not necessarily a large absolute optimality gap, as discussed previously.

Figure 3 provides an illustration of the different cases in which either DO/SA or DO/TS fail to identify the SPNE along with the frequency with which they occur. In Cases 1 to 5, DO/TS returns a value less than the optimal SPNE, in Cases 6 to 8 DO/TS successfully attains the optimal SPNE, and in Cases 9 to 13 DO/TS returns a value larger than optimal. Case 7 is the ideal outcome in which both DO/SA and DO/TS properly identify the SPNE and also has the highest frequency, occurring 23 out of the 52 test instances. The case with the next highest frequency is Case 2 which is also intriguing because this indicates that both DO/SA and DO/TS failed below the SPNE, and they terminated at the same value. Lastly, of the 29 instances for which either DO/SA and/or DO/TS fail, there are 18 instances in which they both fail.

5.4 Initializing DO with Multiple Defender/Attacker Strategies

In an attempt to improve the performance of Double Oracle, we test the effect of initializing DO with multiple defender and attacker strategies on the solution quality. We generate three new defender strategies and two new attacker strategies in addition to the single strategies used above to initialize Double Oracle. The expectation of adding these additional strategies is that they will either (a) improve the

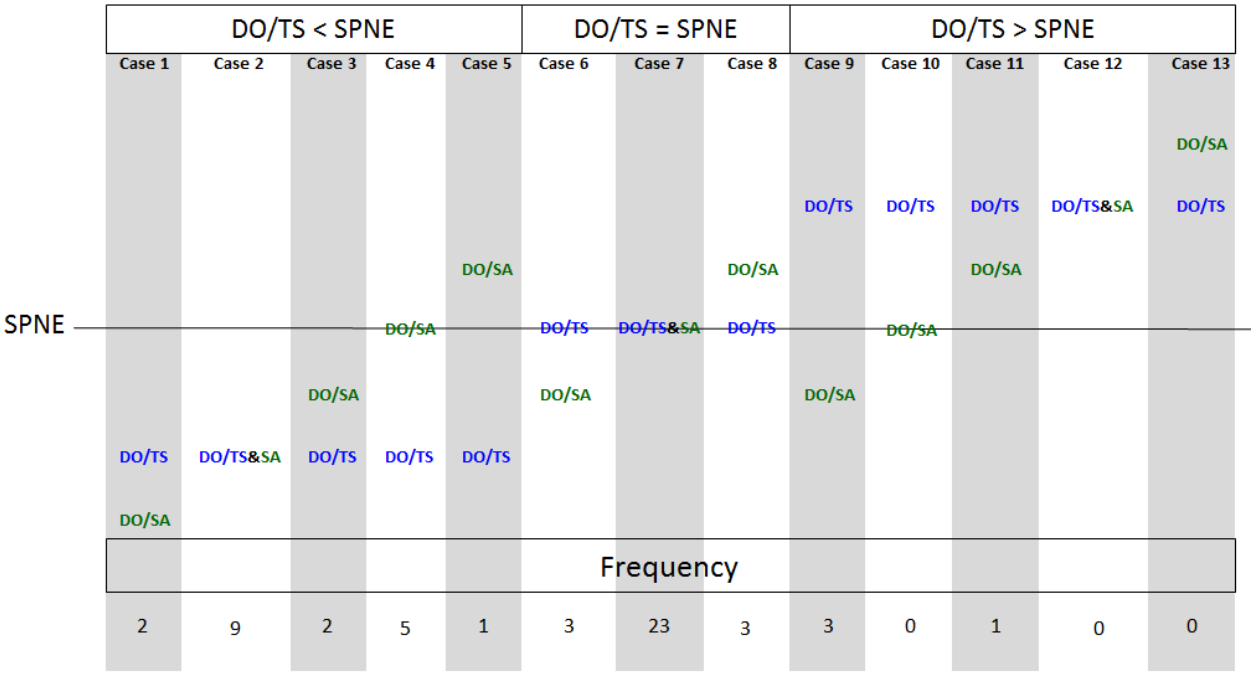


Figure 3. Cases in which DO/TS and/or DO/SA Fail

solution quality in instances which Double Oracle failed by reaching new portions of the strategy space, and/or (b) reduce the computation time by reaching portions of the strategy space quicker. Thus, by adding these new strategies we do not expect the performance to be any worse than the initial results.

The first new defender strategy included in the initial restricted game is the solution to the weighted maximum covering problem formulated in (6a) - (6f). The variable d_i^q is again the decision to locate a SAM battery with IM of type q at location i and r_j indicates whether city j is covered by a SAM battery or not.

Weighted Max Covering:

$$\max_{d,r} \sum_{j \in N} v_j r_j \tag{6a}$$

$$\text{s.t. } r_j \leq \sum_{i \in F} \sum_{q \in T} a_{ij}^q d_i^q, \qquad \forall j \in N, \tag{6b}$$

$$\sum_{i \in F} d_i^q = m_q, \qquad \forall q \in T, \tag{6c}$$

$$\sum_{q \in T} d_i^q \leq 1, \qquad \forall i \in F, \tag{6d}$$

$$d_i^q \in \{0,1\}, \qquad \forall i \in F,\, q \in T, \tag{6e}$$

$$r_j \in \{0,1\}, \qquad \forall j \in N. \tag{6f}$$

The objective function in (6a) represents the objective to maximize the weighted value of cities covered by a SAM battery, where the weight assigned to each city is its city value v_j. Constraint (6b) requires that a SAM battery can only cover a city if the city is within range of the SAM battery while Constraint (6c) requires that all SAM batteries with missiles of type q are employed. Constraint (6d) enforces that no more than one SAM battery will be placed at any location and Constraints (6e) and (6f) enforce binary integer restrictions on the decision to locate a SAM battery at a city and the decision to cover a city, respectively.

The second new defender strategy included in the initial restricted game utilizes a greedy heuristic to place a SAM battery at the most valuable city that is not covered by a SAM battery already placed. In this strategy the first SAM battery is always placed at the most valuable city. The next SAM battery is placed at the next most valuable city uncovered, given the placement of the first SAM battery. For example, if the SAM battery at City 1 also covers City 2 but not City 3, then the second SAM battery would be placed at City 3. If all cities are covered and there are SAM

batteries remaining, the SAM batteries are sequentially placed at the most valuable city without a SAM battery.

The final new defender strategy included in the initial restricted game is the opposite of the single initialization strategy. Instead of SAM batteries being placed in sequential order at the most valuable city without a SAM battery, SAM batteries are placed in sequential order at the *least* valuable city without a SAM battery. This strategy is added in an attempt to diversify the set of initialization strategies.

As previously mentioned, we generate two new attacker strategies to add to the initialization set. The first strategy assigns AMs to cities based on the city value. This strategy assigns $w_j = \left\lceil \frac{v_j}{\sum_{j \in N} v_j} n \right\rceil$ AMs to each city j until all n missiles have been allocated.

The second new attacker strategy included in the initial restricted game iteratively assigns AMs to the city with the highest expected city survival value given the allocation of previous AMs. Let v_j^k represent the expected city survival value and w_j^k represent the number of AMs targeting city j after k AMs have been allocated. Initially, $v_j^0 = v_j$ and the allocation of AMs continues until $k = n$. At each iteration the next attacker missile is assigned to the city with the largest v_j^k value. If $w_j^k = w_j^{k-1}$, implying that the k^{th} AM is not targeting city j, then the expected value remains unchanged and $v_j^k = v_j^{k-1}$. If the k^{th} attacker missile is assigned to target city j, then $w_j^k \neq w_j^{k-1}$ and the expected city survival value is updated according to $v_j^k = v_j^{k-1} p_{max}$, where $p_{max} = \max_q p_q$.

Since DO/TS will always return the same output assuming the same heuristic parameter settings and same initialization strategies, we know that any alteration to the performance of DO/TS from initializing Double Oracle with single defender/attacker strategies will result from the additional strategies in the initialization set. Due to the stochastic element of simulated annealing, the same will not necessarily hold

true; however, the results from DO/TS may be used to infer effects (i.e., if DO/SA performs better and DO/TS performs better, it is likely DO/SA improved due to the different initialization strategies).

5.5 Multiple Strategy Initialization Test Instance Results

The same 26 instances for IM Type 2A and IM Type 2B are tested again to see what effect initializing Double Oracle with multiple defender and attacker strategies will have. Tables 9, 10, 11, and 12 provide the results from initializing Double Oracle with multiple strategies, along with the original results from full enumeration. Table 13 again summarizes the performance in those instances in which DO/SA and DO/TS fail to identify the SPNE.

Overall, initializing Double Oracle with multiple defender and attacker strategies improves DO/SA both in terms of computation time and solution quality. There are 45 instances that required a smaller computation time, with a majority of these instances completing approximately 2 to 4 seconds faster. DO/SA also fails in fewer instances below the SPNE than previously and results in a smaller average and maximum value difference. In addition, there are eight new instances for which DO/SA attains the SPNE and for which it previously failed. Again, it is important to note that this may not always be the outcome as the performance of simulated annealing will most likely vary each time it is run, and thus it is difficult to conclude that this difference in performance is solely attributed to initializing Double Oracle with multiple strategies.

DO/TS also improves upon single strategy initialization both in terms of computation time and solution quality. Of the 52 total test instances, initializing Double Oracle with multiple strategies reduces the computation time in 48 instances, with a maximum reduction time just under 3 seconds. While DO/TS already performs very

Table 9. DO/SA Results with Multiple Strategy Initialization for IM Type A

	Time (sec)		Value			
Instance	Full	DO/SA	Full	DO/SA	Relative Gap SA	Absolute Gap SA
1	15.84	2.74	9.960	**9.960**		
2	94.98	4.60	2.270	**2.270**		
3	93.62	1.40	2.410	**2.410**		
4	95.65	7.46	7.400	**7.400**		
5	91.63	6.97	9.921	**9.921***		
6	200.07	8.00	2.710	2.610	-3.69%	-0.100
7	187.12	1.38	2.820	**2.820**		
8	196.53	4.76	8.924	8.711	-2.39%	-0.213
9	187.84	4.11	10.581	**10.581**		
10	1002.22	10.67	0.150	**0.150***		
11	1140.25	7.22	2.046	**2.046***		
12	1002.77	5.63	3.192	**3.192**		
13	979.47	2.98	3.401	**3.401**		
14	1006.36	5.55	10.430	10.400	-0.29%	-0.030
15	1332.72	4.72	3.874	**3.874**		
16	1839.85	9.14	0.179	0.217	20.86%	0.037
17	1723.53	22.32	0.274	**0.274***		
18	1876.66	15.48	2.640	2.517	-4.67%	-0.123
19	1733.28	6.11	3.883	3.882[1]	-0.02%	-0.001
20	13422.77	15.94	3.317	3.309	-0.24%	-0.008
21	4210.00	7.30	0.374	**0.374**		
22	4079.74	12.09	0.463	0.461	-0.38%	-0.002
23	4915.87	12.57	4.345	4.109	-5.44%	-0.236
24	3571.13	10.48	5.303	**5.303**		
25	10333.44	11.75	1.026	1.019	-0.70%	-0.007
26	31282.68	38.77	4.728	4.734	0.12%	0.006

Text in bold indicates DO reached the SPNE
* indicates improvement over single strategy initialization
[1] indicates DO optimal with single strategy initialization but failed with multiple strategy

quickly and this reduction does not appear too large, this difference may become more noticeable on larger-sized instances or networks. Tables 11 and 12 also identify 12 instances for which DO/TS now properly identifies the SPNE, but for which it failed when initialized with a single defender/attacker strategy. As Table 13 shows,

Table 10. DO/SA Results with Multiple Strategy Initialization for IM Type B

	Time (sec)		Value			
Instance	Full	DO/SA	Full	DO/SA	Relative Gap SA	Absolute Gap SA
1	15.69	5.35	9.960	**9.960**		
2	92.09	7.36	2.410	**2.410**		
3	94.17	10.56	2.410	**2.410**		
4	93.19	5.39	9.921	**9.921***		
5	92.99	6.74	9.921	**9.921***		
6	191.76	2.75	2.820	**2.820**		
7	189.93	2.75	2.820	**2.820**		
8	190.87	4.11	10.581	**10.581**		
9	189.93	6.91	10.581	**10.581**		
10	971.47	15.14	0.150	0.169	12.67%	0.019
11	1183.04	14.85	2.521	**2.521**		
12	980.67	4.53	3.401	**3.401**		
13	979.13	5.79	3.401	**3.401**		
14	976.86	30.47	18.900	18.962	0.33%	0.062
15	997.60	6.06	3.874	**3.874**		
16	1808.44	19.37	0.274	0.281[1]	2.30%	0.006
17	1652.89	14.23	0.326	0.327[1]	0.48%	0.002
18	1826.66	12.66	3.882	3.888	0.16%	0.006
19	1647.21	11.59	5.373	**5.373***		
20	8659.12	19.21	4.132	**4.132***		
21	4226.24	19.56	0.461	0.462	0.08%	<0.001
22	3404.64	9.07	0.500	**0.500**		
23	4219.76	21.59	5.295	5.197	-1.86%	-0.098
24	3403.84	12.52	6.582	6.613[1]	0.48%	0.032
25	9716.15	15.67	1.196	1.260[1]	5.35%	0.064
26	19993.36	26.15	5.381	5.291	-1.67%	-0.090

Text in bold indicates DO reached the SPNE
* indicates improvement over single strategy initialization
[1] indicates DO optimal with single strategy initialization but failed with multiple strategy

initializing Double Oracle with multiple strategies also improves the performance of DO/TS in every measure when it predicts an equilibrium below the optimal SPNE. This is attributed to two main effects, the first simply that DO/TS fails in fewer instances. The second, as Tables 11 and 12 show, is that there are multiple instances

where DO/TS still fails but results in a smaller optimality gap. Interestingly, initializing Double Oracle with multiple strategies had no effect on those instances where DO/TS failed above the SPNE. The reason for this unchanged performance is not immediately clear and warrants further investigation to determine the cause.

Table 11. DO/TS Results with Multiple Strategy Initialization for IM Type A

	Time (sec)		Value			
Instance	Full	DO/TS	Full	DO/TS	Relative Gap TS	Absolute Gap TS
1	15.84	0.19	9.960	**9.960**		
2	94.98	0.72	2.270	**2.270**		
3	93.62	0.18	2.410	**2.410***		
4	95.65	1.39	7.400	**7.400**		
5	91.63	0.55	9.921	**9.921***		
6	200.07	0.70	2.710	2.610	-3.69%	-0.100
7	187.12	0.17	2.820	**2.820**		
8	196.53	0.57	8.924	8.711	-2.39%	-0.213
9	187.84	0.17	10.581	**10.581**		
10	1002.22	1.29	0.150	**0.150**		
11	1140.25	0.98	2.046	**2.046***		
12	1002.77	1.18	3.192	**3.192**		
13	979.47	0.41	3.401	**3.401**		
14	1006.36	1.05	10.430	9.710	-6.90%	-0.720
15	1332.72	0.48	3.874	**3.874**		
16	1839.85	1.00	0.179	**0.179***		
17	1723.53	1.64	0.274	**0.274***		
18	1876.66	2.30	2.640	2.517	-4.67%	-0.123
19	1733.28	1.12	3.883	3.882[1]	-0.02%	-0.001
20	13422.77	2.80	3.317	3.309	-0.24%	-0.008
21	4210.00	0.80	0.374	**0.374**		
22	4079.74	1.80	0.463	0.461	-0.38%	-0.002
23	4915.87	1.11	4.345	4.517	3.97%	0.172
24	3571.13	1.84	5.303	**5.303**		
25	10333.44	1.02	1.026	1.011	-1.49%	-0.015
26	31282.68	3.39	4.728	4.812	1.78%	0.084

Text in bold indicates DO reached the SPNE

* indicates improvement over single strategy initialization

[1] indicates DO optimal with single strategy initialization but failed with multiple strategy

Table 12. DO/TS Results with Multiple Strategy Initialization for IM Type B

	Time (sec)		Value			
Instance	Full	DO/TS	Full	DO/TS	Relative Gap TS	Absolute Gap TS
1	15.69	0.18	9.960	**9.960**		
2	92.09	0.39	2.410	**2.410***		
3	94.17	0.30	2.410	**2.410***		
4	93.19	0.55	9.921	**9.921***		
5	92.99	0.55	9.921	**9.921***		
6	191.76	0.17	2.820	**2.820**		
7	189.93	0.17	2.820	**2.820**		
8	190.87	0.17	10.581	**10.581**		
9	189.93	0.18	10.581	**10.581**		
10	971.47	1.45	0.150	**0.150**		
11	1183.04	1.13	2.521	**2.521**		
12	980.67	0.36	3.401	**3.401**		
13	979.13	0.52	3.401	**3.401**		
14	976.86	1.43	18.900	18.957	0.30%	0.057
15	997.60	0.54	3.874	**3.874**		
16	1808.44	1.10	0.274	**0.274***		
17	1652.89	1.48	0.326	**0.326***		
18	1826.66	1.23	3.882	**3.882***		
19	1647.21	0.44	5.373	**5.373**		
20	8659.12	0.61	4.132	**4.132**		
21	4226.24	1.18	0.461	**0.461**		
22	3404.64	0.84	0.500	**0.500**		
23	4219.76	2.26	5.295	5.303	0.15%	0.008
24	3403.84	0.39	6.582	**6.582**		
25	9716.15	1.12	1.196	**1.196**		
26	19993.36	1.60	5.381	5.291[1]	-1.67%	-0.090

Text in bold indicates DO reached the SPNE
* indicates improvement over single strategy initialization
[1] indicates DO optimal with single strategy initialization but failed with multiple strategy

A final point worthy of discussion are those instances for which Double Oracle succeeded in identifying the SPNE when initialized with a single defender/attacker strategy but failed to properly identify the SPNE with the addition of new strategies. Since DO/SA randomly explores the strategy space, it is harder to determine the

Table 13. DO Performance with Multiple Strategy Initialization

	DO/SA		DO/TS	
	Below SPNE	Above SPNE	Below SPNE	Above SPNE
# of instances	11	10	9	4
Average %	-1.94%	4.28%	-2.38%	1.55%
Max %	-5.44%	20.86%	-6.90%	3.97%
Average Value Diff	-0.083	0.023	-0.141	0.080
Max Value Diff	-0.236	0.064	-0.720	0.172

cause of this outcome. One advantage of DO/TS is that it will always return the same output, allowing us to investigate the cause of this difference.

There is one instance (i.e., instance 19) for IM Type 2A and one instance (i.e., instance 26) for IM Type 2B where such a failure occurs for DO/TS. In both of these instances, the expected value returned is smaller than the SPNE, implying that the defender is not doing as well as it could. We explore this disparity by observing the strategies returned from *coreLP*, Attacker Oracle, and Defender Oracle upon completion of each iteration from initializing Double Oracle with single strategies, and then with multiple strategies.

In both instances when Double Oracle is initialized with a single defender/attacker strategy, the optimal defender strategy is a best response to the attacker strategy of allocating all n missiles to the first city. When the newly generated attacker strategies are added to the initialization set, the attacker no longer selects this strategy from the set of strategies available. As a result, the optimal defender strategy is no longer returned as a best response in Defender Oracle. This is not to say that the first attacker strategy is the only one that causes the optimal defender best response, but none of the other attacker strategies in the initialization set or those generated out of Attacker Oracle induce this defender best response strategy.

5.6 Implementation Note

The final test instances with the results provided in Tables 4 through 12 were run on the computer with specifications given in Section 5.3. When first developing the code in MATLAB, these instances were occasionally run on various other computers. All computers executed the same set of code and invoked the same version of CPLEX solving each instance. An interesting result of utilizing different computers is that there exist a few instances within the test set for which different computers returned different expected values for DO/TS when additional initialization strategies were utilized. Specifically, a few computers solved Instance 19 for IM Type 2A and Instance 26 for IM Type 2B optimally, whereas the final test computer failed for these two instances. After examining each iteration of Double Oracle, we were able to attribute the cause of this difference to one factor: the presence of alternative optima when solving the weighted max covering problem. The computers returned different optimal strategies to the weighted max covering problem, so the initialization strategies created for Double Oracle were not the same on all computers.

This disparity highlights an important feature of the problem in that (a) it clearly shows that the initialization strategies have an effect on the performance of Double Oracle, and (b) the presence of alternative optimal strategies impacts the performance of Double Oracle. From Table 4, for example, we found that there are alternative optimal defender/attacker strategies to the full game in those instances where DO/TS did attain the SPNE values but returned a different strategy (or strategies) than reported by full enumeration. This implies that, when we solve Defender Oracle for the optimal defender strategy given a fixed attacker strategy $\hat{s}_a$, the strategy returned may affect the ability of DO/TS to attain the SPNE if there are alternative defender best response strategies.

Due to this observation, it may be worth considering solving for these alternative

optima if they exist. However, returning to the weighted max covering problem, there are multiple instances for which the number of alternative optimal solutions is upwards of 30, about half the size of the total number of strategies available to the defender. Therefore, it is not practical to initialize Double Oracle with this many strategies, especially in larger instances, as the effect on increasing computation time would be too large to be of value.

5.7 Initialization Pairs

As a final focus of analysis within Double Oracle, we initialize the restricted game with each combination of defender/attacker strategies from the multiple initialization strategy set. There are four initial defender strategies and three initial attacker strategies, for a total of 12 combinations. D1 and A1 represent the defender and attacker strategies used in single strategy initialization as discussed in Section 5.1, respectively. D2 represents the defender strategy to the solution of the weighted max covering problem, D3 represents the defender strategy using the greedy allocation of SAMs to uncovered cities, and D4 represents the last defender strategy of all SAM batteries at the the least valuable cities. A2 represents the attacker strategy of assigning $w_j = \left\lceil \frac{v_j}{\sum_{j \in N} v_j} n \right\rceil$ AMs to each city j, and A3 represents the attacker strategy of iteratively assigning AM to the most valuable city based on the expected value of all previously assigned missiles.

The results from initializing Double Oracle with each strategy pair are provided in Tables 14 and 15. In this sense, the term 'initialization pair' refers to the specific combination of a single defender/ single attacker initialization strategy combination (e.g., D1A3). First, note that these results are only provided for DO/TS and, because the combination D1A1 represents the same initialization pair as the original results in Section 5.3, they have been omitted from these tables. The attacker strategy A2

and A3 are also identical for the first instance with only 5 AM available and, as a result, the strategy A3 was omitted from the initialization pair.

Tables 14 and 15 provide further insight into the performance of Double Oracle. First, across all 52 instances, there are 16 instances for which all 12 initialization pairs successfully identify the SPNE. In addition, there are 10 instances in which all initialization pairs fail to identify the SPNE. Of the 10 instances for which all fail, there are 3 instances for which all 12 initialization pairs fail at the same SPNE value. This is an interesting result as it indicates that the initialization strategies do not have a large impact on the performance of DO/TS in these instances.

In addition, none of the initialization pairs exactly matches the performance of DO/TS under multiple strategy initialization, as reported in Tables 11 and 12. Further, if we compare each combination of initialization pairs (i.e., D1A1 with D1A2, D1A1 with D1A3, ..., D4A2 with D4A3), no two combinations yield the exact same performance. That is, no combination of initialization pairs identify the same SPNE value for every instance. This indicates that each defender/attacker strategy in the multiple strategy initialization set contributes to the performance of Double Oracle and as a result, each strategy is important to include in the initialization set. Subsequently, Tables 14 and 15 again illustrate how the performance of Double Oracle can vary with different initialization strategies.

Table 14. DO/TS Results with Single Strategy Initialization Combinations for IM Type A

Instance	Full	D1 A2	D1 A3	D2 A1	D2 A2	D2 A3	D3 A1	D3 A2	D3 A3	D4 A1	D4 A2	D4 A3
1	9.960	**9.960**		**9.960**	**9.960**		**9.960**	**9.960**		**9.960**	**9.960**	
2	2.270	**2.270**	**2.270**	**2.270**	**2.270**	**2.270**	**2.270**	**2.270**	**2.270**	**2.270**	**2.270**	**2.270**
3	2.410	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***
4	7.400	**7.400**	**7.400**	**7.400**	7.071[1]	7.071[1]	**7.400**	7.071[1]	7.071[1]	**7.400**	**7.400**	**7.400**
5	9.921	**9.921***	**9.921***	**9.921***	**9.921***	**9.921***	**9.921***	**9.921***	**9.921***	9.821	9.821	9.821
6	2.710	2.610	2.610	2.570	2.570	2.570	2.570	2.570	2.570	2.570	2.570	2.570
7	2.820	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**
8	8.924	8.711	8.711	8.711	8.711	8.711	8.711	8.711	8.711	8.711	8.711	8.711
9	10.581	**10.581**	**10.581**	10.281[1]	10.281[1]	10.281[1]	10.281[1]	10.281[1]	10.281[1]	10.281[1]	10.281[1]	10.281[1]
10	0.150	0.146[1]	**0.150**	**0.150**	**0.150**	**0.150**	**0.150**	0.146[1]	**0.150**	**0.150**	0.146[1]	**0.150**
11	2.046	1.725	2.023	**2.046***	**2.046***	**2.046***	2.023	1.725	2.023	2.023	2.023	2.023
12	3.192	**3.192**	**3.192**	**3.192**	**3.192**	**3.192**	**3.192**	**3.192**	**3.192**	**3.192**	**3.192**	**3.192**
13	3.401	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**
14	10.430	9.500	9.710	9.710	16.500	9.710	9.710	14.900	9.710	9.710	17.832	9.710
15	3.874	**3.874**	**3.874**	**3.874**	**3.874**	**3.874**	**3.874**	**3.874**	**3.874**	**3.874**	**3.874**	**3.874**
16	0.179	0.136	0.170	0.170	0.136	0.170	**0.179***	**0.179***	**0.179***	**0.179***	0.136	0.170
17	0.274	0.271	**0.274***	0.271	0.271	0.271	**0.274***	**0.274***	**0.274***	0.271	0.271	**0.274***
18	2.640	2.420	2.229	2.517	3.026	3.026	2.420	2.420	2.420	2.517	2.517	2.517
19	3.883	**3.883**	**3.883**	3.607[1]	3.882[1]	3.882[1]	3.882[1]	3.882[1]	3.882[1]	3.621[1]	3.882[1]	3.621[1]
20	3.317	3.309	3.309	3.111	3.245	3.245	3.149	3.245	3.221	3.149	3.245	2.886
21	0.374	**0.374**	**0.374**	**0.374**	**0.374**	**0.374**	**0.374**	**0.374**	**0.374**	**0.374**	**0.374**	**0.374**
22	0.463	0.461	0.461	0.461	0.456	0.461	0.461	0.456	0.456	0.461	0.456	0.456
23	4.345	4.517	4.517	4.517	4.517	4.517	4.517	4.517	4.517	4.517	4.109	4.109
24	5.303	**5.303**	**5.303**	**5.303**	**5.303**	**5.303**	**5.303**	**5.303**	**5.303**	**5.303**	**5.303**	5.045[1]
25	1.026	1.011	0.918	1.011	1.011	1.011	1.011	1.011	0.918	0.918	1.011	0.918
26	4.728	4.812	4.902	4.812	**4.728***	4.812	4.812	**4.728***	4.058	4.902	4.812	4.112

Text in bold indicates DO reached the SPNE

* indicates improvement over single strategy D1A1 initialization

[1] indicates DO optimal with single D1A1 strategy, but failed with new combination

Table 15. DO/TS Results with Single Strategy Initialization Combinations for IM Type B

Instance	Full	D1 A2	D1 A3	D2 A1	D2 A2	D2 A3	D3 A1	D3 A2	D3 A3	D4 A1	D4 A2	D4 A3
1	9.960	**9.960**		**9.960**	**9.960**		**9.960**	**9.960**		**9.960**	**9.960**	
2	2.410	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***
3	2.410	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***	**2.410***
4	9.921	**9.921***	**9.921***	**9.921***	**9.921***	**9.921***	**9.921***	**9.921***	**9.921***	9.821	9.821	9.821
5	9.921	**9.921***	**9.921***	**9.921***	**9.921***	**9.921***	**9.921***	**9.921***	**9.921***	9.821	9.821	9.821
6	2.820	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**
7	2.820	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**	**2.820**
8	10.581	**10.581**	**10.581**	10.281[1]	10.281[1]	10.281[1]	10.281[1]	10.281[1]	10.281[1]	10.281[1]	10.281[1]	10.281[1]
9	10.581	**10.581**	**10.581**	10.281[1]	10.281[1]	10.281[1]	10.281[1]	10.281[1]	10.281[1]	10.281[1]	10.281[1]	10.281[1]
10	0.150	0.146[1]	**0.150**	**0.150**	**0.150**	**0.150**	**0.150**	0.146[1]	**0.150**	**0.150**	0.146[1]	**0.150**
11	2.521	**2.521**	**2.521**	**2.521**	**2.521**	**2.521**	**2.521**	**2.521**	**2.521**	**2.521**	**2.521**	**2.521**
12	3.401	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**
13	3.401	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**	**3.401**
14	18.900	18.957	18.957	18.957	18.957	18.957	18.957	18.957	18.957	18.957	18.957	18.957
15	3.874	**3.874**	**3.874**	**3.874**	**3.874**	**3.874**	**3.874**	**3.874**	**3.874**	**3.874**	**3.874**	**3.874**
16	0.274	0.160	0.162	0.247	0.160	0.162	**0.274***	**0.274***	**0.274***	0.247	0.160	0.162
17	0.326	0.192	0.311	0.311	0.192	0.311	**0.326***	**0.326***	**0.326***	0.311	0.196	0.311
18	3.882	**3.882***	**3.882***	3.385	3.321	**3.882***	**3.882***	**3.882***	**3.882***	3.385	3.385	3.385
19	5.373	**5.373**	**5.373**	**5.373**	**5.373**	**5.373**	**5.373**	**5.373**	**5.373**	5.189[1]	**5.373**	5.189[1]
20	4.132	**4.132**	**4.132**	**4.132**	**4.132**	**4.132**	**4.132**	**4.132**	**4.132**	**4.132**	**4.132**	**4.132**
21	0.461	**0.461**	**0.461**	**0.461**	0.400[1]	0.427[1]	**0.461**	0.400[1]	0.423[1]	**0.461**	0.400[1]	0.400[1]
22	0.500	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**	0.479[1]	**0.500**	**0.500**	0.479[1]
23	5.295	5.303	5.303	5.303	5.303	5.303	5.303	5.303	5.303	5.303	5.303	5.303
24	6.582	**6.582**	**6.582**	**6.582**	**6.582**	**6.582**	5.882[1]	**6.582**	5.882[1]	**6.582**	**6.582**	**6.582**
25	1.196	**1.196**	**1.196**	**1.196**	**1.196**	**1.196**	**1.196**	**1.196**	**1.196**	**1.196**	**1.196**	**1.196**
26	5.381	5.291[1]	**5.381**	**5.381**	5.291[1]	**5.381**	**5.381**	5.291[1]	**5.381**	**5.381**	5.291[1]	**5.381**

Text in bold indicates DO reached the SPNE

* indicates improvement over single strategy D1A1 initialization

[1] indicates DO optimal with single D1A1 strategy, but failed with new combination

Finally, there is only one initialization strategy pair that appears to dominate another strategy pair, meaning it matches the performance and successfully reaches the SPNE in at least one additional instance. This is the initialization pair D1A3, which dominates D1A1. The pair D1A3 successfully identifies the SPNE for Instance 17 on IM Type 2A, and it is also the only pair that does not fail in an instances for which D1A1 was successful.

5.8 Full Enumeration IM Swap

At this point, we return focus to DAD Model 1. Recall that the main difference in this model compared to DAD Model 2 is that the defender is able to launch multiple types of IMs to protect any given city. While this is a more difficult model to solve, we can use the stage 3 strategy identified by full enumeration for DAD Model 2 as a starting feasible solution, and examine *IM Swaps* based on this solution. In doing so, we define an *IM swap* in one of two ways. For IM Swap 1, we first determine the number of IM Type 2 remaining and the cities within range capable of being protected by these missiles. If there is a city within range that is currently protected using only IM Type 1, we begin to substitute IM Type 2 for IM Type 1, thereby mixing IM types used to protect the city. If there are multiple cities within range currently protected using IM Type 1, we begin the swap at the most valuable city.

IM Swap 2 is not a swap of IMs used; rather it allows the defender to use previously unallocated IMs to protect a new city by mixing IM types. In such a search, we begin with the most valuable city currently not protected and determine the number of IM of both types that could be used to protect the city. If the total number is at least as large as the number of AMs, we begin allocating IMs to defend the city until enough IMs have been allocated to match incoming AMs one to one, starting with IM Type 2.

In applying the above IM Swaps to the solution from full enumeration, there is the possibility that the order the swaps are applied will be important in terms of the increase in the defender's expected utility. As such, we apply the IM Swap in both orders. That is, we perform one iteration in which we use the full enumeration solution and apply IM Swap 1 followed by IM Swap 2, and one iteration in which we apply IM Swap 2 followed by IM Swap 1. The larger expected utility to the defender of the two is then selected.

Table 16 provides the results for applying this swap technique for IM Type 2A, and Table 17 provides the results for IM Type 2B. Further, since this swap is not guaranteed to increase the defender's expected value (and it will never decrease it), results are only reported for instances in which the defender's expected value increased.

Table 16. Full Enumeration SPNE Value IM Swap Type 2A

Instance	Full	IM Swap	Increase in SPNE
8	8.924	9.343	0.420
14	10.430	12.524	2.094
18	2.640	2.643	0.003
20	3.317	3.357	0.040
21	0.374	0.405	0.031
22	0.463	0.484	0.021
23	4.345	4.450	0.106
24	5.303	5.528	0.226
26	4.728	4.784	0.056

Table 17. Full Enumeration SPNE Value IM Swap Type 2B

Instance	Full	IM Swap	Increase in SPNE
21	0.461	0.473	0.012
23	5.295	5.377	0.082

Table 16 and 17 show that the largest increase in the SPNE is 2.094 for instance 14 with IM Type 2A. Returning to Table 2, we note that this is also the instance

in which $p_2 = 1$, and this is attributed as the main cause for the large increase. Excluding this instance, the largest increase in the SPNE is 0.420, with a majority of the instances increasing the SPNE by less than 0.100. There are two main factors for this small change in SPNE. First, the p_q-values are relatively close together, so mixing IM types to protect a city will not drastically change the SPNE. Secondly, the largest increase in SPNE would arise as a result of protecting a city that was previously unprotected. Returning to the DAD Model 2 full enumeration solution, a majority of the cities are already protected. In fact, the fewest number of cities protected is three, and this only occurs in a few instances. Thus, the only real change is mixing IMs utilized on an already protected city, which as previously mentioned will not cause a large change. While this IM swap technique is not an exact solution method, it does provide an indication that the assumption limiting the defender to at most one IM type employed to protect a city is not too constricting on the defender.

VI. Conclusions

6.1 Summary

With the continual advancement of technology and evolving threats around the world, missile defense remains a key area that accounts for billions of dollars in the U.S. acquisition process alone. As new technologies and weapon systems are introduced, each brings unique capabilities to help combat this threat. While current systems such as the THAAD, Patriot, and AEGIS are capable of providing protection, the focus is beginning to shift towards more cost effect alternatives capable of providing the necessary protection against an overwhelming attack [12]. A defender may need to protect a large, dispersed area, and close attention must be given to properly allocate defensive resources in order to maximize the utility of each system. This is a problem faced not only by the U.S. but by nations around the world. As Goure [13] argues, this requires a coordinated effort that takes full advantage of the capabilities available in any given area. The above factors, along with the ramifications for the failure to deal with such a threat, contribute towards the necessity for the research and advancement of air defense systems and their deployment.

The problem of properly positioning air defense systems may seem simple, but the defender must select a single strategy to implement from an overwhelmingly large number of those available. We observed that solving the problem exactly via enumeration may be plausible for instances of small-size, but the time required increases too sharply to remain tractable for realistically-sized instances. Even by leveraging the assumption of a perfect information game and eliminating defender or attacker strategies at appropriate stages, this method still required several hours to complete. In contrast, Double Oracle offers the advantage of significantly reducing the computation time required at the potential cost of converging to a solution that incorrectly

approximates the SPNE. While this may be concerning, it appears that the optimality gap is not large enough to completely discount the use of Double Oracle as a viable solution method. This is the case whether simulated annealing or tabu search was used to solve the Attacker Oracle subproblem within Double Oracle, as both heuristics have unique features that can be leveraged.

Further, the performance of Double Oracle can be improved by altering its implementation. This was observed by adding new strategies to the initial restricted game, and the success of Double Oracle properly identifying the SPNE in a larger number of instances than under single strategy initialization. In the final test instances with the additional initialization strategies, we note that the average computation time required by Double Oracle utilizing tabu search to solve Attacker Oracle is 0.95 seconds, a significant improvement over implementing full enumeration. For those instances for which Double Oracle failed to properly identify the SPNE, the average relative deviation is less than 2.5%, which equates to an absolute deviation in SPNE value of approximately 0.13. Lastly, in the event that the defender utilizes mixed IM types to protect a city, we have not found this to have a large effect on the change in the expected survival value of the city, assuming the p_q-values are relatively close.

6.2 Future Research

While Double Oracle provides a large improvement over full enumeration and even scales well to solve larger instances, it does come at the potential cost of failing to properly identify the SPNE. Thus, the improvement in the implementation of Double Oracle yields one such area for further work. One element to consider is solving for mixed strategies, or alternative optima, when solving for the best response within Defender and Attacker Oracle. This may increase the computation time required for Double Oracle, with the benefit of exploring more of the strategy space allowing for

an improved performance in predicting the SPNE.

Returning to the network topology, another aspect to examine is considering geographical factors that may be important. If a defense network is designed to provide protection against a known threat, then the location of this threat may be an important factor to consider. For example, assume that all AMs will be launched from one direction (e.g., from the west). In such a scenario, it may be possible for a SAM battery to protect a city to the east that is outside of the coverage radius, as the AM will have to fly over this SAM battery first. This would allow the defender another factor to consider when placing SAM batteries and also help accurately reflect how additional information can be leveraged when selecting a strategy to implement. Such a change would likely warrant a new model formulation to accurately address the different aspects that influence protection capabilities.

Heterogeneous Air Defense Battery Location: A Game Theoretic Approach

2d Lt Nicholas T. Boardman
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

Objective

- Identify the optimal locations for a heterogeneous set of air defense batteries for a defender to protect a set of cities, given an attacker's best response to launch a limited number of ballistic missiles and the batteries' collective optimal response to the attack, as limited by interceptor magazine depths, ranges, and probability of interception

Modelling Framework

- Formulate a two-player, three-stage Defender-Attacker-Defender, sequential move, complete information game
  - Stage 1: Defender places SAM batteries
  - Stage 2: Attacker launches all missiles
  - Stage 3: Defender launches interceptors

Related Literature

- Maximum Expected Covering Location Problem
- Weapon Target Assignment Problem
- Double Oracle Algorithm
- Selected Heuristics

Trilevel Math Programming Formulation

N	The set of all cities
F	The set of possible SAM locations
T	The set of IM types
v_j	The value of city j
p_q	Probability a single interceptor successfully intercepts an attacker missile
m_q	The number of SAM batteries with interceptors of type q
n	The total number of attacker missiles
c_q	The number of interceptors per SAM battery
A^q	A coverage matrix indicating whether a SAM battery is capable of protecting a city

d_i^q	Defender's placement of SAM Batteries
w_j	Attacker's allocation of attacker missiles
ψ_j^q	Defender's decision to protect a city
x_{ij}^q	Defender's utilization of interceptors

$$\max_{d} \min_{w} \max_{x,\psi} \sum_{j\in N} v_j \sum_{q\in T} \psi_j^q p_q^{w_j} \quad \text{Expected Surviving City Value}$$

Subject to

$$\sum_{i\in F} d_i^q = m_q, \quad \forall q \in T, \qquad \sum_{j\in N} w_j = n, \qquad \text{SAM and attacker missile constraints}$$

$$\sum_{j\in N} x_{ij}^q \le c_q d_i^q, \quad \forall i \in F, q \in T,$$

$$\sum_{i\in F} d_i^q x_{ij}^q \le w_j \psi_j^q, \quad \forall j \in N, q \in T, \qquad \text{Interceptor utilization}$$

$$\sum_{i\in F} d_i^q x_{ij}^q \ge w_j \psi_j^q, \quad \forall j \in N, q \in T,$$

$$\sum_{q\in T} d_i^q \le 1, \quad \forall i \in F,$$

$$\sum_{q\in T} \psi_j^q \le 1, \quad \forall j \in N, \qquad \text{SAM protection limitations}$$

$$\psi_j^q \in \{0,1\}, \quad \forall j \in N,$$

$$d_i^q \in \{0,1\}, \quad \forall i \in F, q \in T,$$

$$w_j \in \mathbb{Z}^+, \quad \forall j \in N,$$

$$x_{ij}^q \in \mathbb{Z}^+, \quad \forall i \in F, j \in N, q \in T. \qquad \text{Variable restrictions}$$

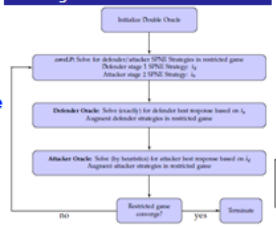
Challenges

- Trilevel formulation cannot be solved via direct optimization
- Enumeration is not a viable solution method for realistically sized instances

Methodologies Examined

1. Full Enumeration
2. Reduced Enumeration w/o dominated strategies
3. Double Oracle (DO) Algorithm

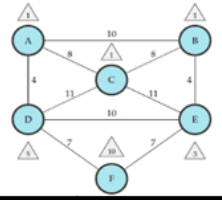
DO Algorithmic Flowchart



Implementation variations

1. Heuristics to solve attacker oracle subproblem: simulated annealing (DO/SA) and tabu search (DO/TS)
2. Initial game initialization: single vs. multiple strategies for each player

Test Instance Topology



Set of Test Instances

- Designed experiment on n, m_q, c_q, and p_q values
- Yielded 52 test instances for each solution method and variation

Solution Method	Computation Time Min	Max
Full Enumeration	15 seconds	8.7 hours
Reduced Enumeration	12 seconds	6.3 hours
DO/SA	1.35 seconds	39 seconds
DO/TS	0.17 seconds	3.39 seconds

- Full and Reduced enumeration are not tractable for large size instances
- Double Oracle greatly reduces computation time, is not guaranteed to be optimal, but had enhanced performance with multiple strategy initialization

DO/SA

	Below Optimal	Attain Optimal	Above Optimal
# of instances	11	31	10
Avg Absolute Deviation	-0.083		0.023
Max Absolute Deviation	-0.236		0.064

DO/TS

	Below Optimal	Attain Optimal	Above Optimal
# of instances	9	39	4
Avg Absolute Deviation	-0.141		0.080
Max Absolute Deviation	-0.720		0.172

Conclusions

- Double Oracle (DO/TS) properly (and rapidly) identifies the optimal strategy for both players in 75% of test instances
- Average absolute deviation is very small when double oracle does fail: approximately 0.13

Future Work

- Consider larger-sized test instances and alternative benchmarking
- Incorporate radar placement and acquisition of attacker missiles
- Incorporate attacker missile tracks vis-à-vis SAM intercept opportunities

Bibliography

1. Ahuja, Ravindra K., Arvind Kumar, Krishna C. Jha, and James B. Orlin. 2007. Exact and Heuristic Algorithms for the Weapon-Target Assignment Problem. *Operations Research*, **55**(6), 1136–1146.

2. Arce, Daniel G., Dan Kovenock, and Briand Roberson. 2012. Weakest-Link Attacker-Defender Games with Multiple Attack Technologies. *Naval Research Logistics (NRL)*, **59**(6), 457–469.

3. Bier, Vicki M., Santiago Oliveros, and Larry Samuelson. 2007. Choosing What to Protect: Strategic Defensive Allocation Against an Unknown Attacker. *Journal of Public Economic Theory*, **9**(4), 563–587.

4. Brown, Gerald, Matthew Carlyle, Douglas Diehl, Jeffrey Kline, and Kevin Wood. 2005. A Two-Sided Optimization for Theater Ballistic Missile Defense. *Operations Research*, **53**(5), 745–763.

5. Brown, Gerald, Matthew Carlyle, Javier Salmerón, and Kevin Wood. 2006. Defending Critical Infrastructure. *Interfaces*, **36**(6), 530–544.

6. Church, Richard, and Charles ReVelle. 1974. The maximal covering location problem. *Papers in Regional Science*, **32**(1), 101–118.

7. Daskin, Mark S. 1983. A maximum expected covering location model: formulation, properties and heuristic solution. *Transportation Science*, **17**(1), 48–70.

8. Gady, Franz-Stefan. 2015. *US and Japan Successfully Test Ballistic Missile Killer*. `http://thediplomat.com/2015/06/us-and-japan-successfully-test-ballistic-missile-killer/`. Accessed 25 June 2015.

9. Galati, David G., and Marwan Simaan. 2007. Effectiveness of the Nash strategies in competitive multi-team target assignment problems. *Aerospace and Electronic Systems, IEEE Transactions on*, **43**(1), 126–134.

10. Glover, Fred. 1990. Tabu search: A tutorial. *Interfaces*, **20**(4), 74–94.

11. Golalikhani, Mohsen, and Jun Zhuang. 2011. Modeling Arbitrary Layers of Continuous-Level Defenses in Facing with Strategic Attackers. *Risk Analysis*, **31**(4), 533–547.

12. Goure, Daniel. 2015a. *Directed Energy Can Defeat Massed Missile Salvoes*. `http://www.realcleardefense.com/articles/2015/07/11/directed_energy_can_defeat_massed_missile_salvoes_108212.html`. Accessed 16 July 2015.

13. Goure, Daniel. 2015b. *The Futrue is in Regional Missile Defense Architectures*. http://www.realcleardefense.com/articles/2015/12/30/the_future_is_in_regional_missile_defense_architectures_108843.html. Accessed 4 January 2016.

14. Guéret, Christelle, Christian Prins, and Marc Sevaux. 1999. Applications of optimization with Xpress-MP. *contract*, 00034.

15. Halvorson, Erik, Vincent Conitzer, and Ronald Parr. 2009. Multi-Step Multi-Sensor Hider-Seeker Games. *Pages 159–166 of: Proceedings of the International Joint Conferences on Artificial Intelligence*, vol. 9.

16. Han, Chan Y., Brian J. Lunday, and Matthew J. Robbins. (In press). A Game Theoretic Model for the Optimal Disposition of Integrated Air Defense Missile Batteries. *INFORMS Journal on Computing*.

17. Haphuriwat, Naraphorn and Vicki M. Bier. 2011. Trade-offs Between Target Hardening and Overarching Protection. *European Journal of Operational Research*, **213**(1), 320–328.

18. Hausken, Kjell. 2011. Protecting complex infrastructures against multiple strategic attackers. *International Journal of Systems Science*, **42**(1), 11–29.

19. Jain, Manish, Vincent Conitzer, and Milind Tambe. 2013. Security scheduling for real-world networks. *Pages 215–222 of: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*. International Foundation for Autonomous Agents and Multiagent Systems.

20. Jr, Sydney J Freedberg. 2015. *Save Our Seoul: Can Lasers & Rail Guns Protect Korea?* http://breakingdefense.com/2015/05/save-our-seoul-can-lasers-rail-guns-protect-korea/. Accessed 12 May 2015.

21. Kirkpatrick, Scott, C. Daniel Gelatt, and Mario P. Vecchi. 1983. Optimization by simulated annealing. *Science*, **220**(4598), 671–680.

22. Korzhyk, Dmytro, Vincent Conitzer, and Ronald Parr. 2011. Security games with multiple attacker resources. *Page 273 of: IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22.

23. McMahan, H. Brendan, Geoffrey J. Gordon, and Avrim Blum. 2003. Planning in the presence of cost functions controlled by an adversary. *Pages 536–543 of: International Conference on Machine Learning*.

24. Myerson, Roger B. 1999. Nash equilibrium and the history of economic theory. *Journal of Economic Literature*, 1067–1082.

25. Nash, John. 1951. Non-cooperative games. *Annals of Mathematics*, 286–295.

26. National Air and Space Intelligence Center. 2013. *Ballistic & Cruise Missile Threat.*

27. Raytheon. 2015. *Standard Missile-3 is the World's Only Ballistic Missile Killer Deployable on Land or at Sea.* `http://www.raytheon.com/capabilities/products/sm-3/`. Accessed 30 June 2015.

28. Scaparra, Maria P., and Richard L. Church. 2008. A bilevel mixed-integer program for critical infrastructure protection planning. *Computers & Operations Research*, **35**(6), 1905–1923.

29. Selten, Reinhard. 1975. Reexamination of the perfectness concept for equilibrium points in extensive games. *International Journal of Game Theory*, **4**(1), 25–55.

30. Tsai, Jason, Thanh H. Nguyen, Nicholas Weller, and Milind Tambe. 2014. Game-Theoretic Target Selection in Contagion-Based Domains. *The Computer Journal*, **57**(6), 893–905.

31. United States Department of Defense. 2014. *Program Acquisition Cost by Weapon System.*

32. United States Department of Defense. 2015. *Program Acquisition Cost by Weapon System.*

33. United States Joint Chiefs of Staff. 2012. *Joint Publication 3-01: Countering Air and Missile Threats.*

34. Yang, Rong, Christopher Kiekintveld, Fernando Ordonez, Milind Tambe, and Richard John. 2011. Improving resource allocation strategy against human adversaries in security games. *Page 458 of: IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704–0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
24–03–2016	Master's Thesis	Sept 2014 — Mar 2016

4. TITLE AND SUBTITLE

Heterogeneous Air Defense Battery Location: A Game Theoretic Approach

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Boardman, Nicholas T., 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-16-M-091

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

In the air defense context of a missile-and-interceptor engagement, a challenge for the defender is that surface to air interceptor missile batteries often must be located to protect high-value targets dispersed over a vast area, subject to an attacker observing the disposition of batteries prior to developing and implementing an attack plan. To model this scenario, we formulate a two-player, three-stage, perfect information, sequential move, zero-sum game. The resulting trilevel math programming formulation cannot be solved via direct optimization and it is not suitable to solve via full enumeration for realistically-sized instances. We instead utilize the game tree search technique Double Oracle, within which we embed alternative heuristics to solve an important subproblem for the attacker. Whereas full enumeration required up to 8.6 hours to solve the largest instance considered, our superlative implementation of Double Oracle terminates in a maximum of 3.39 seconds over the set of instances and properly identifies the optimal SPNE strategies in 75% of our test instances. Regarding those instances for which Double Oracle failed, we note that the relative deviation is less than 2.5% from optimal, on average, yielding promise as a solution method to solve realistically-sized instances.

15. SUBJECT TERMS

air defense, Double Oracle, subgame perfect equilibrium, heuristics

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			LTC Brian J Lunday, AFIT/ENS
U	U	U	UU	86	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636, x4624;brian.lunday@afit.edu