# Balancing Tradeoffs in Security Games: Handling Defenders and Adversaries with Multiple Objectives

by

Matthew Brown

---

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

August 2015

## Acknowledgments

Having now gone through the process, I can unequivocally say that one of the most difficult challenges of writing my thesis was finding the proper words to express my gratitude to those who helped me reach this point. Earning a Ph.D. is often thought of as a solitary, individual process. However, over the last five years, I have learned that the exact opposite is true. The journey that was my Ph.D. was only made possible and, more importantly, made enjoyable by the experiences, conversations, and relationships shared with all of the people whom I have had the privilege of meeting along the way.

I would like to start by thanking my advisor, Professor Milind Tambe, for seeing the potential in me and accepting me as a PhD student. I came to Teamcore without much knowledge of game theory, optimization, or even a clear idea of what it meant to conduct proper research. I learned so much under your tutelage and you provided me sage advice every step of the way including knowing when to give me close guidance as well as when to give me the freedom to explore on my own. Your passion and dedication to research, and use-inspired research in particular, is palpable and something I will strive to emulate in my career. Your commitment to your students is unparalleled and extends long after graduation. The Teamcore family that has been formed over the years stands as a tribute to that commitment. You have opened doors for me that I could not have even imagined five years ago and for that I will be forever grateful.

I would like to thank my committee members: Richard John, Jonathan Gratch, Ewa Deelman, and Dale Kiefer. Your feedback and suggestions were instrumental in guiding the trajectory of this thesis. The unique perspective you each brought to my committee allowed me to better understand my own research and how it fits into the context of related work.

I was fortunate enough to lead the development of two applications for government agencies. First, I would like to thank United States Coast Guard personnel Sam Cheung, Nate Allen, Joe Prado, and Namon Dimitroff for their input and help in making the ARMOR-Fish application a success including a preliminary deployment. Second, I would like to thank Kenneth Fletcher and Jerry Booker from the Transportation Security Administration for their consistent support of the DARMS application and for providing the opportunity to conduct a pilot study in the field.

During my time at USC, I had the privilege of collaborating with many excellent professors and post docs: Chris Kiekintveld, Pradeep Varakantham, Fernando Ordonez, Bo An, Albert Jiang, Francesco Delle Fave, William Haskell, and Arunesh Sinha. I thank you for your guidance and insights on the projects we worked on as well as the patience you showed in your mentoring.

Being in a large research group, I was fortunate enough to share the Ph.D. experience with a number of fellow students who all became my friends: James Pita, Manish Jain, Jason Tsai, Jun-young Kwak, Zhengyu Yin, Rong Yang, Eric Shieh, Thanh Nguyen, Leandro Marcolino, Fei Fang, Chao Zhang, Yundi Qian, Debarun Kar, Benjamin Ford, Haifeng Xu, Amulya Yadav, Aaron Schlenker, Sara Mc Carthy, Yasi Abbasi, and Shahrzad Gholami. I enjoyed our conversations which have ranged from being academically stimulating to deeply profound to utter silly and everything in between. I will always remember sharing offices, traveling to conferences, grinding through conference deadlines, attending group seminars and retreats, as well as the rest of the little bonding moments that made up our lives as Ph.D. students. It was a privilege to work

alongside such a group of bright and talented researchers which is evidenced by all of the great achievements by everyone since leaving Teamcore, and I have no doubt the current students will achieve a similar level of success once their time at USC is over.

Finally, I would like to thank my family for the love and support they have provided me over the years. First and foremost, I would like to thank my parents, Richard and Anita Brown, for always pushing me and believing in me. Your phone calls, letters, and care packages helped to encourage and motivate me, particularly at the moments during my Ph.D. where things were at their most difficult. Additionally, I would like to thank my aunts and uncles Phyllis Schubert, John Casey, Joe Schubert, Al Schubert, Kristin Schubert, Frank Schubert, Greg Fedro, James Brown, Lynn Brown, Rocky Raasch, and Nancy Raasch as well as my cousins Kevin Brownstein, Veronica Schubert, Kyle Schubert, Eddie Schubert, Holly Raasch, Ian Brown, and Trevor Brown. I consider myself to be blessed beyond all measure to be a member of such an amazing and loving family. Everything I have achieved in my life, I have done to make all of you proud.

# Table of Contents

# List of Figures

# Abstract

Stackelberg security games (SSG) have received a significant amount of attention in the literature for modeling the strategic interactions between a defender and an adversary, in which the defender has a limited amount of security resources to protect a set of targets from a potential attack by the adversary. SSGs are at the heart of several significant decision-support applications deployed in real world security domains. All of these applications rely on standard assumptions made in SSGs, including that the defender and the adversary each have a single objective which is to maximize their expected utility. Given the successes and real world impact of previous SSG research, there is a natural desire to push towards increasingly complex security domains, leading to a point where considering only a single objective is no longer appropriate.

My thesis focuses on incorporating multiple objectives into SSGs. With multiple conflicting objectives for either the defender or adversary, there is no one solution which maximizes all objectives simultaneously and tradeoffs between the objectives must be made. Thus, my thesis provides two main contributions by addressing the research challenges raised by considering SSGs with (1) multiple defender objectives and (2) multiple adversary objectives. These contributions consist of approaches for modeling, calculating, and analyzing the tradeoffs between objectives in a variety of different settings. First, I consider multiple defender objectives resulting

from diverse adversary threats where protecting against each type of threat is treated as a separate objective for the defender. Second, I investigate the defender's need to balance between the exploitation of collected data and the exploration of alternative strategies in patrolling domains. Third, I explore the necessary tradeoff between the efficacy and the efficiency of the defender's strategy in screening domains. Forth, I examine multiple adversary objectives for heterogeneous populations of boundedly rational adversaries that no longer strictly maximize expected utility.

The contributions of my thesis provide the novel game models and algorithmic techniques required to incorporate multiple objectives into SSGs. My research advances the state of the art in SSGs and opens up the model to new types of security domains that could not have been handled previously. As a result, I developed two applications for real world security domains that either have been or will be tested and evaluated in the field.

# Chapter 1: Introduction

Security is an ever-present challenge for governments and organizations worldwide. This challenge stems from the fundamental fact that, regardless of the domain, there will always be a limited availability of security resources. As a result, perfect security is never achievable. Therefore, it is critical for decision makers to leverage these limited security resources to the fullest extent possible. Decision makers thus seek principled, mathematical approaches for allocating their security resources to protect against potential adversaries.

Game theory has become a well-established paradigm for modeling security domains which feature complex resource allocation problems. In particular, Stackelberg security games (SSG) have received a significant amount of attention in the literature for modeling such domains [Conitzer and Sandholm, 2006; Kiekintveld et al., 2009; Paruchuri et al., 2008; Jain et al., 2010a]. Security games capture the strategic interactions between a defender (e.g., security agency) and an adversary (e.g., terrorist, criminal). The goal of the defender is to develop a strategy for allocating a limited amount of security resources to protect a set of targets. The adversary is able to observe the defender's strategy and then plans an attack on one of the targets. Given the ability of the adversary to conduct surveillance, the optimal strategy for the defender is an intelligent randomization over resource allocation strategies.

SSGs are at the heart of several significant decision-support applications deployed in the real world. Examples of these applications include ARMOR used at Los Angeles International Airport (LAX) to randomize road checkpoints and canine patrols [Pita et al., 2008], IRIS deployed by the United States Federal Air Marshals Service to assign air marshals to international flights [Tsai et al., 2009], PROTECT utilized by the United States Coast Guard to schedule boat patrols for protecting ports [Shieh et al., 2012], and TRUSTS developed for the Los Angeles Sheriffs Department to generate patrol schedules through the local metro system [Yin et al., 2012].

In all of these applications, both the defender and the adversary are modeled as having a single objective which is to maximize their expected utility. However, as the research on SSGs has advanced, there has been a push towards increasingly complex security domains where the assumption of the players optimizing a single objective may no longer be sufficient. Indeed, allocating resources in virtually any real-world security domain is inherently a multi-objective decision making process. There are any number of quantitative and qualitative considerations that a decision maker could take into account when selecting a strategy to implement. However, previous work reduced the multiple objectives that may have been present in the respective security domains into a single objective either for computational efficiency or simplicity of analysis.

My thesis focuses on modeling more of the complexity present in security domains and addresses the research challenges raised by introducing multiple objectives into security games. The first part of my thesis considers situations where the defender is trying to achieve multiple objectives at the same time. The second part of my thesis considers protecting against heterogeneous populations of boundedly rational adversaries with multiple objectives. The two topics are naturally interconnected but also introduce unique research challenges. The technical contributions of my thesis serve to remove the restriction of only modeling players with a single objective and

(a) Metro Systems


(b) Traffic Patrolling

(c) Aviation Passsenger Screening

(d) Fishery Protection

Figure 1.1: Different domains for Stackelberg security games.

allows for the development of decision aids that construct higher fidelity games models of the underlying domain and offer finer granularity in the resulting analysis.

## 1.1 Multiple Defender Objectives

The first part of my thesis considers defenders with multiple objectives. There are numerous scenarios where the defender would want to optimize multiple objectives when selecting which resource allocation strategy to implement. One such scenario is when the defender needs to protect against a diverse set of adversaries and each adversary poses a unique threat to defender. Given their uniqueness, it may be difficult to know beforehand the exact priority that should be

given to defending against each of these adversaries. An alternative approach is to treat protecting against each threat as an explicit objective for the defender, turning the security game and the underlying resource allocation problem into a multi-objective optimization problem.

In order to capture the fact that the defender is explicitly considering multiple objective during the decision making process, I introduced a new model referred to as a multi-objective security game (MOSG). Instead of receiving a single payoff based on the strategy chosen, the defender now receives a vector of payoffs, i.e., one payoff for each of the multiple objectives. For any well-formed multi-objective optimization problem, there is going to conflict or competition between the objectives. The implication then for security games where the defender has multiple objectives is that no single defender strategy can maximize all of the objectives simultaneously. Thus, it becomes necessary to make compromises and determine how to trade off the performance with respect the multiple objectives. For some domains this may be a straightforward process and weights can be assigned to each objective indicating their relative importance. By specifying the weights a priori, the multi-objective optimization problem can be reduced to a single-objective optimization problem. For other domains this process may impossible or undesirable as either the objective weights are unknown or there is interest in considering the multiple objectives explicitly to gain an understanding of the space of compromise solutions and their relative tradeoffs.

Focusing on security domains with multiple explicit objectives for the defender raises several challenges. Unlike standard single-defender-objective SSGs, which have a single optimal solution in terms of defender payoff, MOSGs have a set of Pareto optimal solutions referred to as the Pareto frontier. A solution is said to be Pareto optimal if and only if there exists no other solution with equal or better performance across all objectives. The defender would only want to consider solutions on the Pareto frontier, as for any other solution there would exist at least one

Pareto optimal solution which yields strictly equal or better performance across all objectives. Therefore, the goal of the defender is to find the solutions which make up the Pareto frontier.

One of the main contributions of the first part of my thesis is Iterative-$\epsilon$-Constraints, a general algorithm for generating the Pareto frontier in multi-objective optimization problems. The Pareto frontier can be generated by solving a sequence of constrained single-objective optimization problems (CSOP), where one primary objective is selected to be maximized while lower bounds are specified for the other secondary objectives. Solving each CSOP produces a Pareto optimal solution and adjusting the lower bound constraints on the secondary objectives generates different solutions on the Pareto frontier. Thus, Iterative-$\epsilon$-Constraints uses an iterative approach in generating the sequence of CSOPs to systematically explore the solution space to find the Pareto frontier. To find the individual solutions that make up the Pareto frontier, I introduced an exact approach for solving a mixed-integer linear programming formulation of each CSOP. Additional contributions include developing heuristics and approximate approaches that achieve speedup by exploiting the structure of MOSGs, increasing the scalability of Iterative-$\epsilon$-Constraints while providing solution quality guarantees on approximating the Pareto frontier.

These insights and technical contributions have been utilized and expanded upon to create two applications. The first application is STREETS (STrategic Randomization with Exploration and Exploitation in Traffic patrol Schedules), which I developed to assist the Singapore Ministry of Home Affairs (MHA) in mitigating reckless driving on the Singapore road network. The idea is to provide a game-theoretic approach for deciding when and where to deploy traffic patrols so as to provide the maximum influence on driver behavior. Given the frequent interaction between the police and drivers, there is a significant amount of data on the times and locations of traffic violations. However, this data is collected when the defender issues citations and thus is inherently

available only for patrolled locations. Therefore, STREETS considers two conflicting objectives: (1) minimizing reckless driving by concentrating patrols on areas with high levels of recorded violations (i.e., exploitation); and (2) maximizing the dispersal of patrols to ensure data is collected from all areas (i.e., exploration). STREETS represents the first use of SSGs to explicitly consider the tradeoff between exploration and exploitation when computing the defender's strategy.

The second application is DARMS (Dynamic Aviation Risk Management System) which provides a new approach for the Transportation Security Administration (TSA) to improve aviation passenger screening security by more directly incorporating risk into their operating procedures. In passenger screening, there is an inherent tradeoff between efficacy (maximizing detection of potential threats) and efficiency (maximizing passenger throughput). The high level idea is that fewer resources should be dedicated to screening lower risk passengers and more resources dedicated to screening higher risk passengers, with the goal of finding the balance between screening efficiency and efficacy. The innovation in DARMS is that the screening for each passenger is conditioned on both the passenger's risk level and flight. I introduced a novel game model, Threat Screening Games (TSGs), to capture the interaction between the TSA and a potential terrorist. The TSG model can solved to determine the level of screening that should be applied to passengers in each flight / risk category pair. A proof of concept for DARMS was completed in March 2015 and will be evaluated in an actual airport as part of a pilot study in December 2015.

## 1.2 Multiple Adversary Objectives

The second part of my thesis considers adversaries with multiple objectives. Additionally, for a wide variety of security domains (particularly those outside of counter-terrorism settings), the

human adversaries facing the defender are not perfect rational or utilizing maximizing as is assumed by classical game theory. Instead, these adversaries can be thought of as being boundedly rational [Simon, 1955], meaning that their decision making process is constrained by a combination of factors such as the availability of accurate information, the cognitive ability to process information, as well as the availability of time in which to make a decision. Considering boundedly rational adversaries with multiple objectives poses a number of significant challenges from both a modeling and an algorithmic perspective.

One of the first modeling challenges is even identifying the objectives of the adversary. Unlike the defender, who can be consulted with to elicit information, little may be known as to what objectives guide the decision making process of the adversary. To address this challenge, I incorporated work on human behavior models, specifically the subjective utility quantal response (SUQR) model [Nguyen et al., 2013]. SUQR suggests that rather than responding to expected utility, adversaries respond to subjective utility, a weighted summation over multiple known objectives. SUQR builds off the Quantal Response (QR) model [McKelvey and Palfrey, 1995] which assumes the existence of latent objectives which impact the decision making process and the weights associated with these latent objectives vary probabilistically. Thus, even with a fixed set of weights for the known objectives, SUQR predicts a distribution over the actions of the adversary, providing a measure of robustness against not explicitly considering the latent objectives. SUQR has been demonstrated to better predict the actions of populations of human subjects when compared against other leading human behavior models including QR [Nguyen et al., 2013].

Even after identifying the objectives that the adversary is likely to be considering, a second modeling challenge for the defender is dealing with the uncertainty over the weights that the adversary assigns to each objective. In many security domains with human adversaries, it is

difficult or impossible for the defender to know the exact adversary they are playing against. This type of uncertainty is typically represented by considering a heterogeneous population of potential adversary types that the defender could encounter. Each of these adversary types is defined by a unique weight vector over the objectives. These unique weights mean that different adversary types will respond differently to the defender's strategy. The defender needs to optimize against all of the adversary responses but this can be difficult if the defender does not know the likelihood of encountering each adversary type. Absent this information, taking a more robust approach for selecting the defender's strategy becomes a reasonable compromise.

To handle scenarios involving heterogeneous populations of boundedly rational adversaries with multiple objectives, I proposed a novel robust maxmin SUQR model which maximizes the worst case defender's payoff against any of the potential adversary responses. Introducing this new model raised an algorithmic challenge in terms of scalability. From a computational perspective, SUQR is both nonlinear and nonconvex, making the defender's optimization problem of solving for the optimal strategy more challenging to solve. The challenges associated with handling one SUQR adversary are exacerbated when the defender must protect against a set of such adversaries simultaneously. This is particularly true for domains where the defender strategy space is exponential in size which is the case for essentially any real world security setting.

The main contribution of the second part of my thesis is MIDAS (MaxImin Defense Against SUQR) which computes robust defender strategies for large-scale SSGs with heterogeneous populations of boundedly rational adversaries with multiple objectives. MIDAS is the first algorithm to address both robustness and scalability simultaneously for such SSGs through a novel combination of a robust maximin formulation and incremental strategy generation. Building off the

insights of [Yang et al., 2012, 2013, 2014; Haskell et al., 2014], MIDAS offers two key innovations: (i) a robust game model that generates defender strategies that hedge against the uncertainty over a heterogeneous population of adversaries and (ii) a tractable mixed-integer linear program formulation approximating the robust game model.

In collaboration with the United States Coast Guard (USCG), MIDAS was used to create an application called ARMOR-Fish for protecting fisheries in the Gulf of Mexico, where illegal fishing seriously threatens the health of local fish stocks. The USCG uses surface and air assets to conduct patrols in order to deter and interdict illegal fishermen entering the exclusive economic zone of the United States from Mexico. By using historical data on illegal fishing sightings and interdictions, I was able to learn and construct a population of SUQR adversary types, i.e., the weights different adversary types assigned to the multiple objectives. ARMOR-Fish was then used to produce aircraft patrol schedules that met the specifications and requirements of the USCG, who began live testing of these patrol schedules in the Gulf of Mexico from July 2014 to September 2014. ARMOR-Fish is currently under review by the USCG for further deployment in the Gulf of Mexico as well as in other fisheries around the nation.

## 1.3 Thesis Overview

The structure of the thesis is organized as follows: Chapter 2 discusses the necessary background material for Stackelberg security games. Chapter 3 reviews the relevant research to provide the proper context for the contributions of the thesis. Chapter 4 considers multiple defender objectives resulting from diverse adversary threats where protecting against each type of threat

is treated as a separate objective for the defender. Chapter 5 investigates the defender's balance between the exploitation of collected data and the exploration of alternative strategies in patrolling domains. Chapter 6 explores the defender's tradeoff between strategy efficacy and strategy efficiency in screening domains. Chapter 7 examines multiple adversary objectives for heterogeneous populations of boundedly rational adversaries. Chapter 8 summarizes the thesis and presents possible directions for future work.

# Chapter 2: Background

## 2.1 Stackelberg Security Games

Stackelberg Security Games (SSGs) [Conitzer and Sandholm, 2006; Kiekintveld et al., 2009; Paruchuri et al., 2008] are composed of two players, a leader and a follower, where the leader (denoted as the defender) must protect a set of targets from the follower (denoted as the adversary). The defender has a finite number of resources $r$ with which to protect the set of targets $T$ against the adversary. A pure strategy for the defender is typically an assignment of the $r$ resources to either patrols or targets (depending on the type of SSG), while a pure strategy for the adversary is typically the target that is to be attacked. Each target $t \in T$ is assigned a set of payoffs $\{R_t^a, P_t^a, R_t^d, P_t^d\}$: $R_t^a$ is the reward earned by an adversary if they successfully attack target $t$, while $P_t^a$ is the penalty received by an adversary for an unsuccessful attack on target $t$. Conversely, if the defender assigns a resource to protect target $t$ and an adversary attacks target $t$, the defender receives a reward $R_t^d$ . If an adversary attacks target $t$ and the defender has not assigned a resource to protect target $t$, the defender receives a penalty $P_t^d$. In order to be a valid SSG, it must hold that $R_t^a > P_t^a$ and $R_t^d > P_d^t$, which means that assigning a resource to cover a target more often is always beneficial for the defender and disadvantageous for the adversary.

Careful planning by the defender is necessary as the amount of available security resources is limited, i.e., $r < |T|$, and not all targets can be covered. As the leader in this Stackelberg game, the defender commits to a strategy first. The adversary is then able to conduct surveillance and thus learn the defender's strategy before selecting their own strategy which is a best response. The standard solution concept for a two-player Stackelberg game is a Strong Stackelberg Equilibrium (SSE), in which the defender selects an optimal strategy based on the assumption that the adversary will choose an optimal response while breaking ties in favor of the defender.

The defender strategy space $A$ contains all valid allocations of the security resources. $A_i$ is the $i^{th}$ defender pure strategy and is an assignment of all the security resources. $A_i$ is represented as a column vector $A_i = \langle A_{it} \rangle^T$, where $A_{it}$ indicates whether target $t$ is covered by $A_i$. For example, in an SSG with 4 targets and 2 resources, $A_i = \langle 1, 1, 0, 0 \rangle$ represents the pure strategy of assigning one resource to target $t_1$ and another to target $t_2$. The optimal resource allocation strategy for the defender will be a mixed (i.e., randomized) strategy over the set of defender pure strategies $A$, as any deterministic defender strategy would easily be exploited by the adversary. The defender's mixed strategy can then be represented as a vector $\mathbf{a} = \langle a_i \rangle$, where $a_i \in [0, 1]$ is the probability of choosing $A_i$. There is also a more compact marginal representation for defender strategies. Let $\boldsymbol{x}$ be the marginal strategy, where $x_t = \sum_{A_j \in \mathcal{A}} a_i A_{ti}$ is the probability that target $t$ is covered. Thus, depending on the particular type of security game, the defender is trying to find either the optimal mixed strategy $\mathbf{a}$ or the optimal marginal strategy $\mathbf{x}$.

There have been many algorithms and models developed to solve SSGs, including DOBSS [Paruchuri et al., 2008] which solves SSGs using a mixed-integer linear program, ASPEN [Jain et al., 2010a] which solves SSGs that contain a greater number of defender resources and larger strategy space, ORIGAMI [Kiekintveld et al., 2009] which provides a polynomial time algorithm

for SSGs that contain no scheduling constraints, along with HUNTER [Yin and Tambe, 2012] and RECON [Yin et al., 2011] which compute robust strategies for security games. However, these algorithms do not apply to SSGs with multiple objectives for either the defender or the adversary.

## 2.2 Human Behavior Models

Classical game theory assumes that all players are perfectly rational and will select the strategies that maximize their expected utilities. However, this is often not a reasonable assumption for security domains with human adversaries. [Yang et al., 2012] was the first to address human adversaries in security games by incorporating the quantal response (QR) model [McKelvey and Palfrey, 1995] from the behavioral economics literature. QR predicts a probability distribution over adversary actions where actions with higher expected utility have a greater chance of being chosen. By anticipating possible adversary deviation from the optimal action, strategies computed with QR are more robust to uncertainty in human decision making. [Jiang et al., 2013a] generalized the QR model to be robust against all adversary models satisfying monotonicity (i.e., higher expected utility actions are selected more frequently than lower expected utility actions), but this approach struggles to scale up to larger security games.

[Nguyen et al., 2013] extended the QR model by proposing that humans respond to subjective utility, a weighted summation over multiple objectives (such as avoiding defender coverage, seeking adversary reward, and avoiding adversary penalty), when making decisions. [Nguyen et al., 2013] proposes the subjective utility quantal response (SUQR) model which was shown to outperform QR in human subject experiments. As result, most subsequent research on boundedly rational human adversaries in security games has focused on the SUQR model.

An SUQR adversary type $\omega$ can be represented as a the weight vector $\omega = \{\omega_1, \omega_2, \omega_3\}$ which encodes the relative importance of $x_t$, $R_t^a$, and $P_t^a$, respectively, in the decision making process of the adversary. Recall that the SUQR model selects a probability distribution over adversary actions rather than deterministically selecting the utility maximizing adversary action. Given defender strategy $\boldsymbol{x}$, the probability that adversary $\omega$ will attack target $t$ is

$$q_t\left(\omega \mid \boldsymbol{x}\right) = \frac{e^{\omega_1 x_t + \omega_2 R_t^a + \omega_3 P_t^a}}{\sum_{t'} e^{\omega_1 x_{t'} + \omega_2 R_{t'}^a + \omega_3 P_{t'}^a}}.$$

If an adversary chooses to attack target $t$, then for a given defender strategy $\boldsymbol{x}$, the defender's expected utility is defined as

$$U_t\left(\boldsymbol{x}\right) = x_t R_t^d + \left(1 - x_t\right) P_t^d.$$

For a known adversary type $\omega$, the defender's optimization problem is then

$$\max_{\boldsymbol{x}} F\left(\boldsymbol{x} \mid \omega\right) \triangleq \sum_t U_t\left(\boldsymbol{x}\right) q_t\left(\omega \mid \boldsymbol{x}\right),$$

which can be solved to find the optimal defender marginal strategy $\boldsymbol{x}$.

# Chapter 3: Related Work

## 3.1 Stackelberg Security Games

Stackelberg security games (SSGs) have received a significant amount of attention in the literature [Basilico et al., 2009; Dickerson et al., 2010; Korzhyk et al., 2011b,a; Letchford and Conitzer, 2013; Letchford et al., 2012; Letchford and Vorobeychik, 2013]. Early work in this area was not explicitly focused on security but rather on developing the necessary theoretic and algorithmic concepts necessary to solve general Stackelberg games. [von Stengel and Zamir, 2004] first explored the commitment to mixed (i.e., randomized) strategies in Stackelberg games. [Conitzer and Sandholm, 2006] introduced the first general approach for solving Stackelberg games known as Multiple LPs which solves a linear program for every pure strategy of the adversary. Improving upon Multiple LPs, DOBSS [Paruchuri et al., 2008] uses a mixed-integer linear program to solve for the leader's strategy in general Stackelberg games with a single optimization problem. Additionally, DOBSS represented the first optimal approach for solving Bayesian Stackelberg games, where the leader may face one of multiple follower types.

[Kiekintveld et al., 2009] formalized the Stackelberg security game model and presented the ORIGAMI and ERASER algorithms. ORIGAMI provided a polynomial time algorithm for solving SSGs with no resource constraints, e.g., spatio-temporal constraints if the resource must

conduct patrols. Meanwhile, ERASER provided a compact representation of the defender strategy space for multiple resources, improving the ability to scale up to larger SSGs. Additionally, ERASER was able to handle the type of resources constraints that were not considered by ORIGAMI. ASPEN [Jain et al., 2010a] further enhanced scalability by utilizing a branch-and-price approach that considers only the most relevant defender pure strategies to incrementally solve for the optimal solution, thereby significantly improving the efficiency of solving SSGs.

The extensive literature on SSGs has resulted in a number of decision-support applications including ARMOR [Pita et al., 2008], IRIS [Tsai et al., 2009], GUARDS [Pita et al., 2011], PROTECT [Shieh et al., 2012], TRUSTS [Yin et al., 2012] and RaPtoR [Varakantham et al., 2013]. All of these applications were developed to suggest resource allocation strategies for protecting physical infrastructure such as airports, ports, and metro systems. However, all of these decision aids only consider a single objective for both the defender and the adversary. Additionally, there is an assumption that the adversary is perfectly rational and selects the strategy that maximizes expected utility given the strategy of the defender.

A recent trend in SSGs is looking to the significant volume of research dedicated to developing computational models of human behavior to help relax the strong assumption that the adversary is a perfectly rational utility maximizer. [Simon, 1955] introduced the concept of bounded rationality where the decision maker may not have the time or resources to compute the optimal strategy and thus deviates from strictly maximizing utility. Unlike perfect rationality, Luce's Choice Axiom [Luce, 1959] proposes that strategy selection is a probabilistic process as opposed to a deterministic process. The Quantal Reponse (QR) model [McKelvey and Palfrey, 1995] builds off of that proposition to suggest that people probabilistically respond to expected utility, where strategies with higher expected utility are selected more often than strategies with lower

expected utilities. The selection of suboptimal strategies is parameterized by an estimate of the decision maker's rationality level and is motivated by the existence of other latent objectives influencing the decision making process. Taking inspiration from the Lens model [Brunswik, 1952] and Multi-Attribute Utility Theory [Keeney and Raiffa, 1976], the Subjective Utility Quantal Response model (SUQR) [Nguyen et al., 2013] expands upon QR by explicitly considering multiple weighted objectives in the decision making process (while still allowing for the existence of other latent objectives). Subsequent research [Cui and John, 2014; Kar et al., 2015] has analyzed which set of objectives should be included in the SUQR model.

Incorporating human behavioral models into SSGs represents an important progression that has been demonstrated to improve the performance of defender strategies in both simulations and human subject experiments [Pita et al., 2010; Yang et al., 2012, 2013; Nguyen et al., 2013; Cui and John, 2014]. By introducing stochasticity in strategy selection, behavioral models such as QR and SUQR are able to better predict the actions of real human adversaries and thus lead the defender to choose strategies that perform better in practice. Utilizing these types of boundedly rational human behavioral models raises two fundamental research challenges that previous work in SSGs has tried to address separately: scalability and robustness.

While perhaps counter-intuitive, modeling adversaries which behave suboptimally (from an expected utility perspective) makes the defender's optimization problem computationally more difficult to solve. Both QR and SUQR are non-linear non-convex models which present challenges particularly when considering large-scale security domains. This issue of scalability for SSGs with boundedly rational adversaries has received attention in the literature. [Yang et al., 2012] presented a mixed-integer linear program approximation for QR, improving tractability. Additionally, [Yang et al., 2013] introduces a cutting planes approach which handle resource

constraints and uses a master-slave formulation to iteratively solve for the optimal strategy. However, [Yang et al., 2012, 2013] both only consider a single boundedly rational adversary.

However, in many domains the defender could encounter multiple types of boundedly rational human adversaries. Thus, a separate line of SSG research has focused on achieving robustness against uncertainty in the defender's model of the adversary. [Yang et al., 2014] proposed a Bayesian approach which learns a Gaussian distribution over adversary types but has two potential drawbacks. First, the assumption that the adversary types are normally distributed is difficult to justify in practice. Second, even if the adversaries are normally distributed, a large amount of data is needed to learn the Gaussian distribution. Alternatively, [Haskell et al., 2014] introduced a maximin approach which does not use a distribution over the adversary types. Instead, the defender chooses a strategy that maximizes the worst-case performance over a set of adversary types. Primarily interested in robustness, these approaches for handling multiple boundedly rational adversaries cannot handle large-scale SSGs with complex resource constraints.

My thesis serves to merge these two research threads for the first time by simultaneously addressing scalability and robustness while handling heterogeneous populations of boundedly rational adversaries with multiple objectives. Each thread alone is impractical for important real-world security domains such as environmental crime. Large-scale SSGs with complex resource constraints and multiple boundedly rational adversary types present a number of modeling and computational challenges. However, overcoming these challenges is critical as they are precisely the characteristics that define many real-world security domains.

## 3.2 Multi-Objective Optimization

Expanding beyond a single objective for either the defender or the adversary turns the process of strategy selection into a multi-objective optimization problem. The techniques for solving multi-objective optimization problems can be broken down into three categories [Hwang and Masud, 1979]: *a priori*, *interactive*, and *a posteriori* methods. This classification is determined by the phase in which the decision maker expresses their preferences.

If the preferences of the decision maker are known *a priori* [Steuer, 1989; Zadeh, 1963] then this information can be incorporated into the solution process by assigning each objective a weight according to its relative importance. This weighted summation technique [Chankong and Haimes, 1983] effectively turns a multi-objective optimization problem into a single-objective optimization problem which implies the existence of a single optimal solution. However, it is often difficult for the decision maker to both know and articulate their preferences, especially if prior knowledge as to the shape of the solution space is limited. Bayesian security games [Paruchuri et al., 2008] are solved using this formulation with the weights representing the probability distribution over adversary types. Another issue is that not all preferences over multiple objectives can be expressed as simple weighted summations, more complex preferences may be desired.

*Interactive* methods [Alves and Clmaco, 2007; Luque et al., 2009; Tappeta and Renaud, 1999] involve alternating between computation and dialogue phases. In the computation phase, a set of solutions are computed and presented to the decision maker. In the dialogue phase, the decision maker is asked about their preferences over the set of solutions. The decision maker can thus guide the search process with their responses toward a preferable solution. By using preference elicitation, only a subset of the Pareto frontier needs to be generated and reviewed. The drawback

is that the decision maker never has the opportunity to view the entire Pareto frontier at once and could potentially miss out on a more preferable solution. In addition, solutions must be computed in an online manner which requires synchronization between the system and the decision maker.

Finally, there will be instances where the preferences of the decision maker are only known *a posteriori*. In this situation, the entire Pareto frontier (or a representative subset) is generated and presented to the decision maker. While this approach is the most expensive computationally, it provides the most information, enabling the decision maker to make a more informed decision as tradeoffs between objectives can be observed directly. The three most common *a posteriori* approaches are weighted summation [Kim and de Weck, 2005], evolutionary algorithms [Coello et al., 2007], and the $\epsilon$-constraint method [Haimes et al., 1971].

When weighted summation [Chankong and Haimes, 1983] and its successors are used as a generative approach, the true weights of the decision maker are not known. Thus, it is necessary to sample many different combinations of weights in order to generate the Pareto frontier. Solving for one assignment of weights, $w$, produces a Pareto optimal solution. Since the weight vector is an artificial construct which may not have any real meaning in the optimization problem, it is difficult to know how to update the weights in order to generate different solutions on the Pareto frontier. Another limitation of weighted summation is that it is only guaranteed to find Pareto-optimal solutions in the convex region of the Pareto frontier. The weighted $p$-power method [Lightner and Director, 1981] and the weighted minimax method [Li et al., 1999] were introduced as improved versions of weighted summation capable of handling nonconvex problems.

Another approach for generating the Pareto frontier which has seen significant application [Abido, 2003; Giuliano and Johnston, 2008; Toffolo and Lazzaretto, 2002] is multi-objective evolutionary algorithms (MOEA) [Deb, 2001]. This class of algorithms is inspired by biological

concepts such as reproduction, mutation, recombination, and selection. A population of candidate solutions is maintained and evolved over multiple generations, where the likelihood of survival for individual solutions is determined by a fitness function. A key advantage of evolutionary algorithms such as NSGA-II [Deb et al., 2002], SPEA-2 [Zitzler et al., 2001], and GDE3 [Kukkonen and Lampinen, 2005] is that there is no need to solve optimization problems as the assignment of decision variables are passed down genetically from generation to generation. However, due to the stochastic nature of evolutionary algorithms, the solutions returned by these approaches are not Pareto-optimal but rather approximate solutions. Additionally, it is not possible to bound this level of approximation, making evolutionary algorithms unsuitable for the security domains on which are the focus of this thesis, where quality guarantees are critical.

The third approach is the $\epsilon$-constraint method in which the Pareto frontier is generated by solving a sequence of constrained single-objecitve optimization problems (CSOP). One objective is selected as the primary objective to be maximized while lower bound constraints are added for the other secondary objectives. By varying the constraints, different solutions on the Pareto frontier can be generated. The original $\epsilon$-constraint method [Chankong and Haimes, 1983] discretizes the objective space and solves a CSOP for each grid point. This approach is computationally expensive since it exhaustively searches the high-dimensional space formed by the secondary objectives. There has been work to improve upon the original $\epsilon$-constraint method. In [Laumanns et al., 2006], an adaptive constraint variation scheme is proposed which is able make use of information obtained from previously computed subproblems. However, the exponential complexity of $\mathcal{O}(k^{n-1})$, where $k$ is the number of solutions in the Pareto frontier and $n$ is the number of objectives, limits its application as the Pareto frontier can be large or even continuous for many real world multi-objective optimization problems. Another approach, the augmented $\epsilon$-constraint

method [Mavrotas, 2009] reduces computation by using infeasibility information from previously solved CSOPs. However, this approach returns a predefined number of points and thus cannot bound the level of approximation for the Pareto frontier.

Security domains demand *both* efficiency as well as solution quality guarantees when providing decision support. Given these requirements, my thesis provides the first approach for solving SSGs with multiple defender objectives by utilizing and improving upon the $\epsilon$-constraint method through the following innovations: (1) using a recursive, tree-based algorithm to search the objective space instead of a predefined grid, (2) dynamically generating CSOPs using adaptive constraints from previously computed CSOPs, and (3) exploiting infeasibility information to avoid unnecessary computation. These innovations result in only needing to solve $\mathcal{O}(nk)$ CSOPs and additionally serve to provide approximation bounds on missing Pareto optimal solutions.

# Chapter 4: Multiple Defender Objectives (Diverse Adversary Types)

Game theory is an increasingly important paradigm for modeling security domains which feature complex resource allocation [Basilico et al., 2009; Conitzer and Korzhyk, 2011]. Security games, an important class of attacker-defender Stackelberg games, are at the heart of several significant deployed decision-support applications. Such systems include ARMOR at the Los Angeles International Airport (LAX) [Pita et al., 2008], IRIS deployed by the US Federal Air Marshals Service [Tsai et al., 2009], GUARDS developed for the US Transportation Security Administration [An et al., 2011a], and PROTECT used at the Port of Boston by the US Coast Guard [An et al., 2011a].

While multiple objectives may have been present in these domains, the games are modeled as having the defender optimizing a single objective as the necessary solution concepts did not exist. However, there are domains where the defender has to consider multiple objectives simultaneously. For example, the Los Angeles Sheriff's Department (LASD) needs to protect the city's metro system from ticketless travelers, common criminals, and terrorists.[1] From the perspective of LASD, each one of these attacker types presents a unique threat. Fare evaders are directly responsible for lost revenue by not purchasing the appropriate tickets, criminals can commit crimes against property and persons which undermine the perceived safety of the metro system, and

[1] http://sheriff.lacounty.gov

terrorists can inflict massive casualties, causing long-term system-wide disruptions, and spreading fear through the general public. Given that preventing these threats yield different types of benefit, protecting against each type of attacker could correspond to an objective for LASD.

With a diverse set of attacker types, selecting a security strategy is a significant challenge as no single strategy can maximize all of the objectives. Thus, tradeoffs must be made as increasing protection against one attacker type may increase the vulnerability to another attacker type. However, it is not clear how LASD should weigh the objectives when determining the security strategy to use. One could attempt to establish methods for converting the benefits of protecting against each attacker type into a single objective. However, this process can become convoluted when attempting to compare abstract notions such as safety and security with concrete concepts such as ticket revenue.

Bayesian security games [An et al., 2011a; Conitzer and Sandholm, 2006; Jain et al., 2010b; Kiekintveld et al., 2009; Paruchuri et al., 2008] have been used to model domains where the defender is facing multiple attacker types. The threats posed by the different attacker types are weighted according to the relative likelihood of encountering that attacker type. However, there are three potential factors limiting the applicability of Bayesian security games: (1) the defender may not have information on the probability distribution over attacker types, (2) it may be impossible or undesirable to directly compare the defender rewards for different attacker types, and (3) only one solution is given, hiding the trade-offs between the objectives from the end user.

We propose a new game model, multi-objective security games (MOSG), which combines game theory and multi-objective optimization. Such a model is suitable for domains like the LASD metro system, as the threats posed by the attacker types (ticketless travelers, criminals,

and terrorists) are treated as different objective functions which are not aggregated, thus eliminating the need for a probability distribution over attacker types. Unlike Bayesian security games which have a single optimal solution, MOSGs may have a set of Pareto optimal (non-dominated) solutions which is referred to as the Pareto frontier. By presenting the Pareto frontier to the end user, they are able to better understand the structure of their problem as well as the tradeoffs between different security strategies. As a result, end users are able to make a more informed decision on which strategy to enact. For instance, LASD has suggested that rather than having a single option handed to them, they would be interested in being presented with a set of alternative strategies from which they can make a final selection. Overall, there has been a growing trend towards multi-objective decision making in a wide variety of areas, including transportation [Brauers et al., 2008] and energy [Pohekar and Ramachandran, 2004]. We are pursuing along in the same direction but now from a game-theoretic perspective.

Our key contributions include (i) Iterative-$\epsilon$-Constraints, an algorithm for generating the Pareto frontier for MOSGs by producing a sequence of constrained single-objective optimization problems (CSOP); (ii) an exact approach for solving a mixed-integer linear program (MILP) formulation of a CSOP (which also applies to multi-objective optimization in more general Stackelberg games); (iii) heuristics that exploit the structure of security games to speed up solving the MILPs; and (iv) an approximate approach for solving CSOPs, which greatly increases the scalability of our approach while maintaining quality guarantees. Additionally, we provide analysis of the complexity and completeness for all of our algorithms, detailed experimental results evaluating the effect of MOSG properties and algorithm parameters on performance, as well as several techniques for visualizing the Pareto frontier.

The structure of this chapter is as follows: Section 4.1 motivates our research by providing a detailed description of the LASD domain. Section 4.2 formally introduces the MOSG model as well as multi-objective optimization concepts such as the Pareto frontier and Pareto optimality. Section 4.3 introduces the Iterative-$\epsilon$-Constraints algorithm for solving a series of CSOPs to generate the Pareto frontier. Section 4.4 presents the MILP formulation for solving each CSOP. Section 4.5 proposes heuristics which can be used to constrain our MILP formulation, including three algorithms (ORIGAMI-M, ORIGAMI-M-BS, and DIRECT-MIN-COV) for computing on lower bounds defender coverage. Section 4.6 introduces an approximate algorithm (ORIGAMI-A) for solving CSOPs based on the defender coverage heuristics. Section 4.7 provides experimental results for all of our algorithms and heuristics as well as analysis on the properties of the MOSG model. Section 4.8 discusses a number of approaches for visualizing the Pareto frontier as a step in the decision making process for selecting a security policy to implement. We conclude this chapter and outline future research directions in Section 4.9.

## 4.1 Motivating Domain

There are a variety of real-world security domains in which the defender has to consider multiple, and potentially conflicting, objectives when deciding upon a security policy. In this section, we focus on the one specific example of transportation security, in which LASD is responsible for protecting the Los Angeles metro system, shown in Figure 4.1.[2] The metro system consists of 70 stations and maintains a weekday ridership of over 300,000 passengers. The LASD is primarily concerned with protecting the metro system from three adversary types: ticketless travelers, criminals, and terrorists. A significant number of the rail stations feature barrier-free entrances that

[2] http://www.metro.net/riding_metro/maps/images/rail_map.pdf

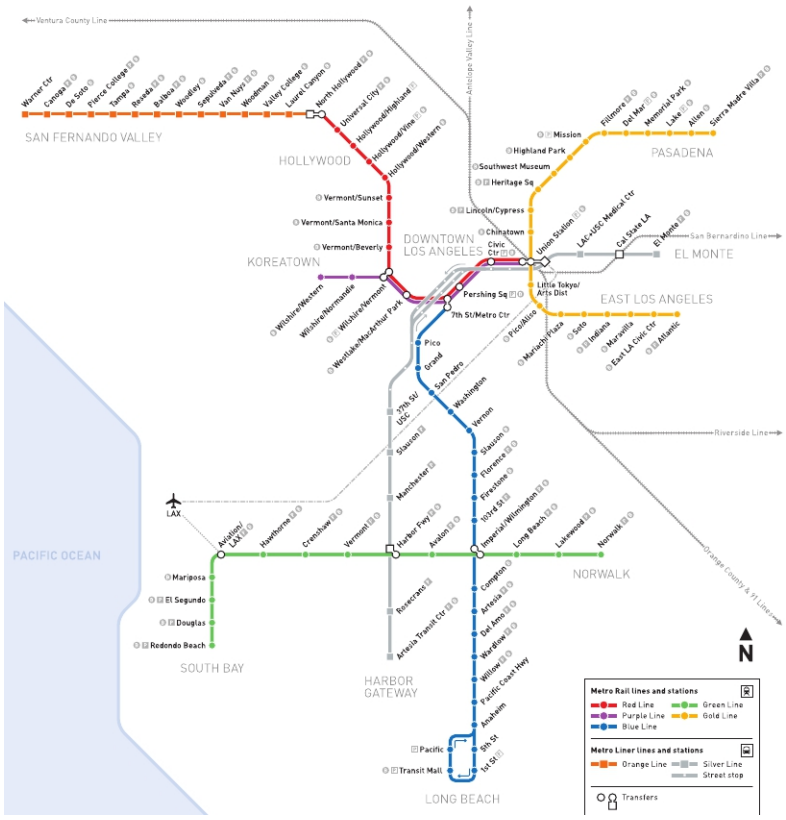



Figure 4.1: Map of the Los Angeles rail system.

do not employ static security measures such as metal detectors or turnstiles. Instead randomized patrols and inspections are utilized in order to verify that passengers have purchased a valid ticket as well as to generally maintain security of the system. Thus, LASD must make decisions on how best to allocate their available security resources as well as on how frequently to visit each station.

Each of the three adversary types are distinct and present a unique set of challenges which may require different responses by LASD. For example, each adversary may have different preferences over the stations they choose to target. Ticketless travelers may choose to fare evade at busier stations thinking that the larger crowds decrease the likelihood of having their ticket

checked. Whereas, criminals may prefer to commit crimes at less frequented stations, as they believe the reduced crowds will result in a smaller security presence. Finally, terrorists may prefer to strike stations which hold economic or cultural significance, as they believe that such choice of targets can help achieve their political goals.

LASD may also have different motivation for preventing the various adversary types. It is estimated that fare evasion costs the Los Angeles metro system over $5 million in lost revenue each year [Iseki et al., 2008]. Deploying security policies that target ticketless travelers can help to recuperate a portion of this lost revenue as it implicitly encourages passengers to purchase tickets. Pursuing criminals will reduce the amount of property damage and violent crimes, increasing the overall sense of passenger safety. In 2010, 1216 "part one crimes" were reported on the metro system, which includes homicide, rape/attempted rape, assault, robbery, burglary, grand theft, and petty theft.[3] Most significantly, the rail system experienced its first and only slaying when a man was fatally stabbed on the subway in August 2011. Finally, due to the highly sensitive nature of the information, statistics regarding the frequency and severity of any terrorist threats targeting the transit system are not made available to the public. However, the city of Los Angeles is well known to be a high priority target given the much publicized foiling of attempted terrorist attacks at LAX in 2000 and 2005. Additionally, trains and subway systems are common targets for terrorism, as evidenced by the devastating attacks on Madrid in 2004 and London in 2005. Thus, despite the relatively low likelihood of a terrorist attack, security measures designed to prevent and mitigate the effects of terrorism must always remain a priority, given the substantial number of lives at risk.

[3] http://thesource.metro.net/2011/09/21/statistics-on-crime-on-metro-buses-and-trains/

LASD is required to simultaneously consider all of the threats posed by the different adversary types in order to design effective and robust security strategies. Thus, defending against each adversary type can be viewed as an objective for LASD. While these objectives are not strictly conflicting (e.g. checking tickets at a station may lead to a reduction in crime), focusing security measures too much on one adversary may neglect the threat posed by the others. As LASD has finite resources with which to protect all of the stations in the city, it is not possible to protect all stations against all adversaries at all times. Therefore, strategic decisions must be made such as where to allocate security resources and for how long. These allocations should be determined by the amount of benefit they provide to LASD. However, if protecting against different adversaries provides different, incomparable benefits to LASD, it may be unclear how to specify such a decision as maximizing a single objective for automated analysis (as in ARMOR and similar systems). Instead, a more interactive process whereby the decision support system presents possible solutions to the decision-makers for further analysis and human judgment may be preferable

For a domain such as the Los Angeles metro system, an MOSG model could be of use, as it can capture the preferences and threats of the adversary types as well as the benefit to LASD of preventing these threats. Solving the MOSG produces a set of candidate solutions with each solution corresponding to a security policy and a set of expected payoffs for LASD, one for each adversary. Thus, different solutions can be compared to better understand the trade-offs between the different objectives. LASD can then select the security policy they feel most comfortable with based on the information they have available. For this type of evaluation process to occur, we must be able to both generate and visualize the Pareto frontier. Our research focuses primarily on developing efficient algorithms for solving MOSGs and generating the Pareto frontier (Sections 4.3 through 4.6), but we also touch on issues relating to visualization (Section 4.8).

## 4.2 Multi-Objective Security Games

A multi-objective security game (MOSG) is a multi-player game between a defender and $n$ attacker types.[4] The defender tries to prevent attacks by covering targets $T = \{t_1, t_2, \ldots, t_{|T|}\}$ using $m$ identical resources which can be distributed in a continuous fashion amongst the targets. The MOSG model adopts the Stackelberg framework in which the defender acts first by committing to a strategy that the attackers are able to observe and best respond. The defender's strategy can be represented as a coverage vector $\mathbf{c} \in C$ where $c_t$ is the amount of coverage placed on target $t$ and represents the probability of the defender successfully preventing any attack on $t$ [Kiekintveld et al., 2009]. This formulation assumes that the covering of each target costs the same amount of resources, specifically one defender resource. It is this assumption that allows for the equivalence between the amount of resources placed on a target and the probability of that target being covered. Thus, given a budget of $m$ resources, the defender could choose to fully protect $m$ targets. However, given the Stackelberg paradigm, such a deterministic strategy would perform poorly, as the attackers can easily select one of the targets that are known to be unprotected. Therefore, the defender has incentive to consider mixed strategies where resources are allocated to a larger set of partially protected targets. While an attacker is still able to observe this mixed strategy, when the MOSG is actually played there is uncertainty on the attacker's part as to whether a target will be covered or not. More formally, $C = \{\langle c_t \rangle | 0 \leq c_t \leq 1, \sum_{t \in T} c_t \leq m\}$ describes the defender's strategy space. The mixed strategy for attacker type $i$, $\mathbf{a}_i = \langle a_i^t \rangle$, is a vector where $a_i^t$ is the probability of attacking $t$.

[4] The defender may actually face multiple attackers of different types, however, these attackers are not coordinated and hence the problem we address is different than in [Korzhyk et al., 2011b].

$U$ defines the payoff structure for an MOSG, with $U_i$ defining the payoffs for the security game played between the defender and attacker type $i$. $U_i^{c,d}(t)$ is the defender's utility if $t$ is chosen by attacker type $i$ and is fully covered ($c_t = 1$). If $t$ is uncovered ($c_t = 0$), the defender's penalty is $U_i^{u,d}(t)$. The attacker's utility is denoted similarly by $U_i^{c,a}(t)$ and $U_i^{u,a}(t)$. A property of security games is that $U_i^{c,d}(t) > U_i^{u,d}(t)$ and $U_i^{u,a}(t) > U_i^{c,a}(t)$ which means that placing more coverage on a target is always beneficial for the defender and disadvantageous for the attacker [Kiekintveld et al., 2009]. For a strategy profile $\langle \mathbf{c}, \mathbf{a}_i \rangle$ for the game between the defender and attacker type $i$, the expected utilities for both agents are given by:

$$U_i^d(\mathbf{c}, \mathbf{a}_i) = \sum_{t \in T} a_i^t U_i^d(c_t, t), \quad U_i^a(\mathbf{c}, \mathbf{a}_i) = \sum_{t \in T} a_t U_i^a(c_t, t)$$

where $U_i^d(c_t, t) = c_t U_i^{c,d}(t) + (1 - c_t) U_i^{u,d}(t)$ and $U_i^a(c_t, t) = c_t U_i^{c,a}(t) + (1 - c_t) U_i^{u,d}(t)$ are the payoff received by the defender and attacker type $i$, respectively, if target $t$ is attacked and is covered with $c_t$ resources.

The standard solution concept for a two-player Stackelberg game is Strong Stackelberg Equilibrium (SSE) [von Stengel and Zamir, 2004], in which the defender commits first to an optimal strategy based on the assumption that the attacker will be able to observe this strategy and then choose an optimal response, breaking ties in favor of the defender. We denote $U_i^d(\mathbf{c})$ and $U_i^a(\mathbf{c})$ as the payoff received by the defender and attacker type $i$, respectively, when the defender uses the coverage vector $\mathbf{c}$ and attacker type $i$ attacks the best target while breaking ties in favor of the defender.

With multiple attacker types, the defender's utility (objective) space can be represented as a vector $U^d(\mathbf{c}) = \langle U_i^d(\mathbf{c}) \rangle$. An MOSG defines a multi-objective optimization problem:

$$\max_{\mathbf{c} \in C} \left( U_1^d(\mathbf{c}), \ldots, U_n^d(\mathbf{c}) \right)$$

We associate a different objective with each attacker type because, as pointed out in Section 4.1, protecting against different attacker types may yield types of payoff to the defender which are not directly comparable. This is in contrast to Bayesian security games, which uses probabilities to combine the objectives into a single weighted objective, making the assumption about identical units of measure for each attacker type.

Solving such multi-objective optimization problems is a fundamentally different task than solving a single-objective optimization problem. With multiple objectives functions there exist tradeoffs between the different objectives such that increasing the value of one objective decreases the value of at least one other objective. Thus for multi-objective optimization, the traditional concept of optimality is replaced by Pareto optimality.

**Definition 1.** *(Dominance). A coverage vector* $\mathbf{c} \in C$ *is said to* ***dominate*** $\mathbf{c}' \in C$ *if* $U_i^d(\mathbf{c}) \geq U_i^d(\mathbf{c}')$ *for all* $i = 1, \ldots, n$ *and* $U_i^d(\mathbf{c}) > U_i^d(\mathbf{c}')$ *for at least one index* $i$*.*

**Definition 2.** *(Pareto Optimality) A coverage vector* $\mathbf{c} \in C$ *is* ***Pareto optimal*** *if there is no other* $\mathbf{c}' \in C$ *that dominates* $\mathbf{c}$*. The set of non-dominated coverage vectors is called* ***Pareto optimal solutions*** $C^*$ *and the corresponding set of objective vectors* $\Omega = \{U^d(\mathbf{c}) | \mathbf{c} \in C^*\}$ *is called the* ***Pareto frontier****.*

This chapter gives algorithms to find Pareto optimal solutions in MOSGs. For many multi-objective optimization problems, the Pareto frontier contains a large or even infinite number of

solutions. In these situations, it is necessary to generate a subset of Pareto optimal solutions that can approximate the true Pareto frontier with quality guarantees. The methods we present in this chapter are a starting point for further analysis and additional preference elicitation from end users, all of which depends on fast approaches for generating the Pareto frontier. This analysis can include creating visual representations of the Pareto frontier, a topic discussed in Section 4.8.

## 4.3 Iterative-$\epsilon$-Constraints

Using the $\epsilon$-constraint method, we translate a multi-objective optimization problem into the following constrained single-objective optimization problem (CSOP) by transforming all but one of the optimizations into a set of constraints $\mathbf{b}$.

$$\begin{aligned} \max_{\mathbf{c} \in C} \quad & U_1^d(\mathbf{c}) \\ & U_2^d(\mathbf{c}) \geq b_2 \\ & U_3^d(\mathbf{c}) \geq b_3 \\ & \cdots \\ & U_n^d(\mathbf{c}) \geq b_n \end{aligned}$$

This allows for the use of standard optimization techniques to solve for a single Pareto optimal solution, which is a vector of payoffs $\mathbf{v} = (U_1^d(\mathbf{c}), \ldots, U_n^d(\mathbf{c}))$. The Pareto frontier is then generated by solving multiple CSOPs produced by modifying the constraints in $\mathbf{b}$.

This section presents Iterative-$\epsilon$-Constraints (Algorithm 1), an algorithm for systematically generating a sequence of CSOPs for an MOSG. After each CSOP is generated, it is passed to

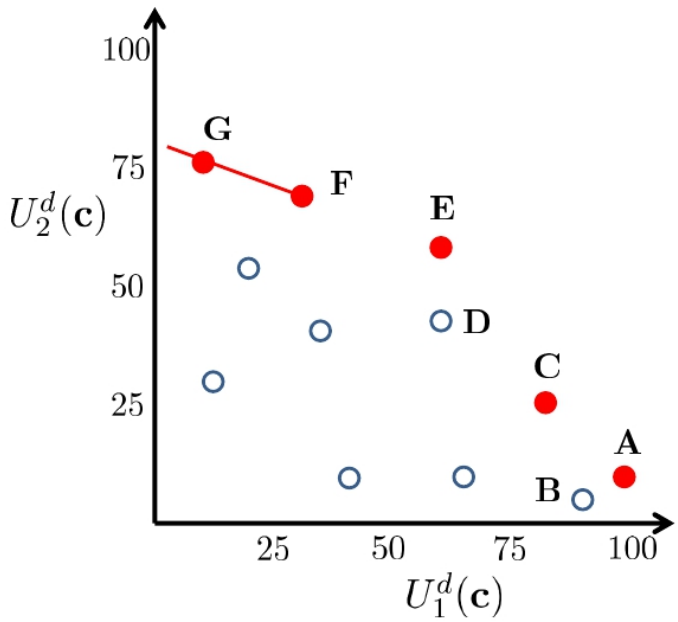



Figure 4.2: Pareto frontier for a bi-objective MOSG.

a solver $\Phi$ and if a solution is found that information is used to generate additional CSOPs. In Section 4.4, we present a MILP approach which guarantees the Pareto optimality of each CSOP solution. While in Section 4.6, we introduce a faster, approximate approach for solving CSOPs.

### 4.3.1 Algorithm for Generating CSOPs

Iterative-$\epsilon$-Constraints uses the following four key ideas: (1) The Pareto frontier for an MOSG can be found by solving a sequence of CSOPs. For each CSOP, $U_1^d(\mathbf{c})$ is selected as the primary objective, which will be maximized. Lower bound constraints $\mathbf{b}$ are then added for the secondary objectives $U_2^d(\mathbf{c}), \ldots, U_n^d(\mathbf{c})$. (2) The sequence of CSOPs can be iteratively generated by exploiting previous Pareto optimal solutions and applying Pareto dominance. (3) It is possible for a CSOP to have multiple coverage vectors $\mathbf{c}$ that maximize $U_1^d(\mathbf{c})$ and satisfy $\mathbf{b}$. Thus, lexicographic maximization is needed to ensure that the CSOP solver $\Phi$ only returns Pareto optimal solutions. (4) It may be impractical (even impossible) to generate all Pareto optimal points if the

frontier contains a large number of points or is continuous. Therefore, a parameter $\epsilon$ is used to discretize the objective space, trading off solution efficiency versus the degree of approximation in the generated Pareto frontier.

We now present a simple MOSG example with two objectives and $\epsilon = 5$. Figure 4.2 shows the objective space for the problem as well as several points representing the objective payoff vectors for different defender coverage vectors. In this problem, $U_1^d$ will be maximized while $b_2$ constrains $U_2^d$, meaning that the utility of the second objective $U_2^d$ should be no less than $b_2$. The initial CSOP is unconstrained (i.e., $b_2 = -\infty$), thus the solver $\Phi$ will maximize $U_1^d$ and return solution *A*=(100,10). Based on this result, we know that any point $\mathbf{v} = \{v_1, v_2\}$ (e.g., *B*) in the objective space is not Pareto optimal if $v_2 < 10$, as it would be dominated by *A*. We then generate a new CSOP, updating the bound to $b_2 = 10 + \epsilon$. Solving this CSOP with $\Phi$ produces solution *C*=(80, 25) which can be used to generate another CSOP with $b_2 = 25 + \epsilon$. Both *D*=(60,40) and *E*=(60,60) satisfy $b_2$ but only *E* is Pareto optimal. Lexicographic maximization ensures that only *E* is returned and dominated solutions are avoided (details in Section 4.4). The method then updates $b_2 = 60 + \epsilon$ and $\Phi$ returns *F*=(30,70), which is part of a continuous region of the Pareto frontier from $U_2^d = 70$ to $U_2^d = 78$. The parameter $\epsilon$ causes the method to select a subset of the Pareto optimal points in this continuous region. In particular this example returns *G*=(10,75) and in the next iteration ($b_2 = 80$) finds that the CSOP is infeasible and terminates. The algorithm returns a Pareto frontier of *A*, *C*, *E*, *F*, and *G*.

Iterative-$\epsilon$-Constraints systematically updates a set of lower bound constraints $\mathbf{b}$ to generate the sequence of CSOPs. Each time we solve a CSOP, a portion of the $n-1$ dimensional space formed by the secondary objectives is marked as searched with the rest divided into $n-1$ subregions (by updating $\mathbf{b}$ for each secondary objective). These $n-1$ subregions are then recursively

searched by solving $n-1$ CSOPs with updated bounds. This systematic search forms a branch and bound search tree with a branching factor of $n-1$. As the depth of the tree increases, the CSOPs are more constrained, eventually becoming infeasible. If a CSOP is found to be infeasible, no child CSOPs are generated because they are guaranteed to be infeasible as well. The algorithm terminates when all of the leaf nodes in the search tree are infeasible, meaning the entire secondary objective space has been searched.

**Algorithm 1:** Iterative-$\epsilon$-Constraints($\mathbf{b} = \{b_2, \ldots, b_n\}$)

```
if b ∉ previousBoundsList then
    append(previousBoundsList, b) ;
    c ← Φ(b) ;
    if c is a feasible solution then
        v ← {U_1^d(c), ..., U_n^d(c)};
        for 2 ≤ i ≤ n do
            b' ← b;
            b'_i ← v_i + ε ;
            if b' ≱ s, ∀s ∈ infeasibleBoundsList then
                Iterative-ε-Constraints(b') ;
    else append(infeasibleBoundsList, b) ;
```

Figure 4.3 shows the type of search tree generated by Iterative-$\epsilon$-Constraints. In this simple example, there are three objectives and thus the search tree has a branching factor of 2. The number at the top of each node represents the order in which the nodes were processed. Along each branch, we show information about $\mathbf{b}$ and $\mathbf{v}$ being passed down from parent to child. This information is used to create the set of lower bound constraints for the child CSOP which is then passed to the solver $\Phi$. In total, seven CSOPs are computed with three feasible CSOPs (Iterations 1, 2, and 4) and four infeasible CSOPs (Iterations 3, 5, 6, and 7). Figure 4.4 shows the process taking place within a CSOP with four objectives, where a vector $\mathbf{v}$ of $n-1$ objective lower bounds is used to formulate the constraints of a CSOP which maximizes the remaining, primary

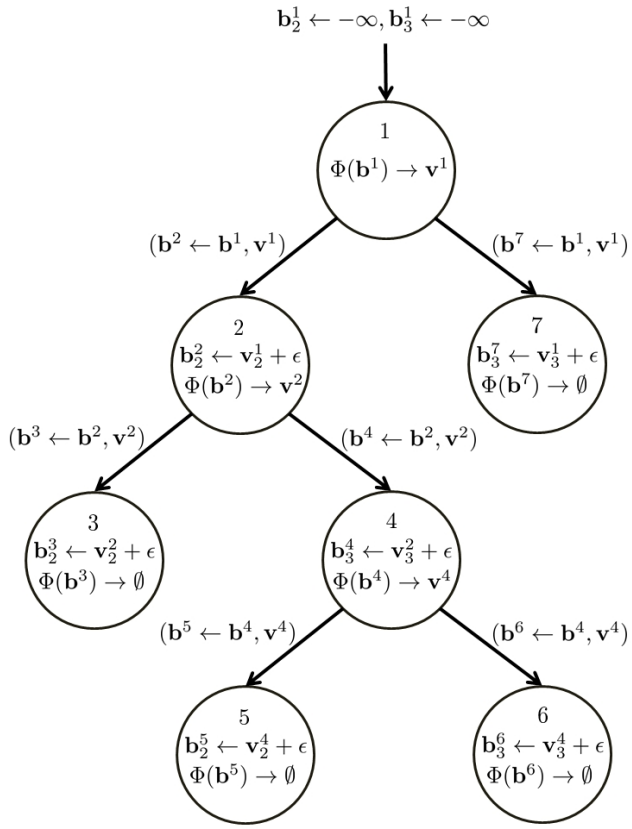



Figure 4.3: Example Iterative-$\epsilon$-Constraints search tree for three objectives.

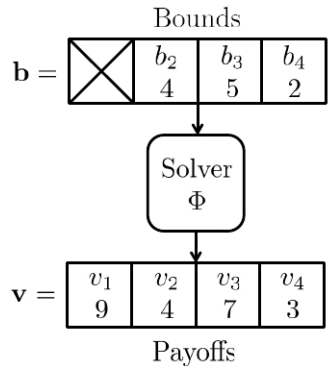



Figure 4.4: Internal process for an example CSOP with four objectives.

objective. This CSOP is then passed to CSOP solver $\Phi$ which produces a vector $\mathbf{v}$ of $n$ objective payoff values.

### 4.3.2 Search Tree Pruning

By always going from less constrained CSOPs to more constrained CSOPs, Iterative-$\epsilon$-Constraints is guaranteed to terminate. However, there are several issues which can cause the algorithm to be inefficient. The first issue is redundant computation caused by multiple CSOPs having identical sets of lower bound constraints. When this occurs, the set of child CSOPs generated for each duplicate parent CSOP would also be identical. Given the recursive nature of the algorithm, these duplicate CSOPs can result in an exponential increase in the number of CSOPs that are solved. This issue can be addressed by recording the lower bound constraints for all previous CSOPs in a list called $\mathrm{previousBoundsList}$ and pruning any new CSOP which matches an element in this list. The second issue is the unnecessary computation of CSOPs which are known to be infeasible based on previously computed CSOPs. This can be achieved by recording the lower bound constraints for all CSOPs previously found to be infeasible in a list called $\mathrm{infeasibleBoundsList}$ and pruning any new CSOP for which all lower bounds constraints are greater than or equal to the lower bound constraints of a CSOP in the list. These two heuristics form the baseline pruning rules that are used when evaluating Iterative-$\epsilon$-Constraints in Section 4.7.

It is possible to further exploit the concept of Pareto dominance in order to create a more effective pruning heuristic. For example, it is possible for two sets of lower bound constraints, $\mathbf{b}^1$ and $\mathbf{b}^2$, to result in the same vector of objective payoffs $\mathbf{v}$. This situation is obviously undesirable not only due to the time spent on the CSOPs corresponding to $\mathbf{b}^1$ and $\mathbf{b}^2$ but also because both CSOPs will have a full set of child CSOPs that need to be processed. While generating some duplicate solutions is unavoidable, steps can be taken to reduce their occurrence. Solving a CSOP

creates a mapping of constraints to payoffs, $\Phi(\mathbf{b}) \rightarrow \mathbf{v}$. Each such mapping provides useful information as it creates a dominated region in which no additional CSOPs need to be solved. Specifically, if we have a mapping $\Phi(\mathbf{b}) \rightarrow \mathbf{v}$, then we can prune any CSOP corresponding to $\mathbf{b}'$ such that $\mathbf{b}' \geq \mathbf{b}$ and $\mathbf{b}' \leq \mathbf{v}$. This is the case because for any such $\mathbf{b}'$ the payoffs found by solving the CSOP are guaranteed to be $\mathbf{v}$. Since $\mathbf{b}' \geq \mathbf{b}$, $\mathbf{b}'$ is inherently at least as constrained as $\mathbf{b}$. Given that the CSOP is a maximization problem, if $\mathbf{b}$ maps to $\mathbf{v}$ then a more constrained problem $\mathbf{b}' \leq \mathbf{v}$ must also map to $\mathbf{v}$. Thus, in Iterative-$\epsilon$-Constraints, we can record all of the constraint-payoff mappings in $solutionsMap$. Then before attempting to solve a CSOP corresponding to $\hat{\mathbf{b}}$, we first check to see if $\hat{\mathbf{b}}$ resides within any of the dominated regions defined by any of the mappings in $solutionsMap$. We compare this more sophisticated pruning rule to the baseline pruning rule in Section 4.7.5.

### 4.3.3 Approximation Analysis

When the Pareto frontier contains a large or infinite number of points, it may be undesirable or impossible to produce the entire Pareto frontier. Thus, the set of solutions returned in such situations is an approximation of the true Pareto frontier. In this section, we prove that the solutions found by Iterative-$\epsilon$-Constraints are Pareto optimal, if $\Phi$ is exact, and then provide formal bounds on the level of approximation in the generated Pareto frontier. We refer to the full Pareto frontier as $\Omega$ and the set of solutions found by Iterative-$\epsilon$-Constraints as $\Omega_\epsilon$.

**Theorem 3.** *Solutions in $\Omega_\epsilon$ are non-dominated, i.e., $\Omega_\epsilon \subseteq \Omega$.*

*Proof.* Let $\mathbf{c}^*$ be the coverage vector such that $U^d(\mathbf{c}^*) \in \Omega_\epsilon$ and assume that it is dominated by a solution from a coverage vector $\bar{\mathbf{c}}$. That means $U_i^d(\bar{\mathbf{c}}) \geq U_i^d(\mathbf{c}^*)$ for all $i = 1, \ldots, n$ and for some $j$, $U_j^d(\bar{\mathbf{c}}) > U_j^d(\mathbf{c}^*)$. This means that $\bar{\mathbf{c}}$ was a feasible solution for the CSOP for which $\mathbf{c}^*$

was found to be optimal. Furthermore, the first time the objectives differ, the solution $\bar{\mathbf{c}}$ is better and should have been selected in the lexicographic maximization process. Therefore $\mathbf{c}^* \notin \Omega_\epsilon$ which is a contradiction. □

We have just shown that each solution in $\Omega_\epsilon$ is indeed Pareto optimal. However, the use of $\epsilon$ introduces a degree of approximation in the generated Pareto frontier. Specifically, by not generating the full Pareto frontier, we are approximating the shape of $\Omega$. One immediate question is to characterize the efficiency loss caused by this approximation. Here we define a bound to measure the largest efficiency loss as a function of $\epsilon$:

$$\rho(\epsilon) = \max_{\mathbf{v} \in \Omega \setminus \Omega_\epsilon} \min_{\mathbf{v}' \in \Omega_\epsilon} \max_{1 \leq i \leq n} (v_i - v'_i)$$

This approximation measure is widely used in multi-objective optimization (e.g. [Bringmann et al., 2011]). It computes the maximum distance between any point $\mathbf{v} \in \Omega \setminus \Omega_\epsilon$ on the frontier to its "closest" point $\mathbf{v}' \in \Omega_\epsilon$ computed by our algorithm. Here, the distance between two points is the maximum difference of different objectives.

**Theorem 4.** $\rho(\epsilon) \leq \epsilon$.

*Proof.* It suffices to prove this theorem by showing that for any $\mathbf{v} \in \Omega \setminus \Omega_\epsilon$, there is at least one point $\mathbf{v}' \in \Omega_\epsilon$ such that $v'_1 \geq v_1$ and $v'_i \geq v_i - \epsilon$ for $i > 1$.

Algorithm 2 recreates the sequence of CSOP problems generated by Iterative-$\epsilon$-Constraints by ensuring the bounds $\mathbf{b} \leq \mathbf{v}$ throughout. Since Algorithm 2 terminates when we do not update $\mathbf{b}$, this means that $v'_i + \epsilon > v_i$ for all $i > 1$. Summarizing, the final solution $\mathbf{b}$ and $\mathbf{v}' = U^d(\Phi(\mathbf{b}))$ satisfy $\mathbf{b} \leq \mathbf{v}$ and $v'_i > v_i - \epsilon$ for all $i > 1$. Since $\mathbf{v}$ is feasible for the CSOP with bound $\mathbf{b}$, but $\Phi(\mathbf{b}) = \mathbf{v}' \neq \mathbf{v}$ then $v'_1 \geq v_1$. ■ □

Given Theorem 4, the maximum distance for every objective between any missed Pareto optimal point and the closest computed Pareto optimal point is bounded by $\epsilon$. Therefore, as $\epsilon$ approaches 0, the generated Pareto frontier approaches the complete Pareto frontier in the measure $\rho(\epsilon)$. For example if there are $k$ discrete solutions in the Pareto frontier and the smallest distance between any two is $\delta$ then setting $\epsilon = \delta/2$ will make $\Omega_\epsilon = \Omega$. In this case, since each solution corresponds to a non-leaf node in our search tree, the number of leaf nodes is no more than $(n-1)k$. Thus, our algorithm will solve at most $\mathcal{O}(nk)$ CSOPs. This is a significant improvement over [Laumanns et al., 2006], which solves $\mathcal{O}(k^{n-1})$ CSOPs as a result of recomputing each cell in an adaptive grid every time a solution is found. Our approach limits recomputing regions of objective space through our pruning heuristics and by moving from less constrained to more constrained CSOPs.

**Algorithm 2:** For $\mathbf{v} \in \Omega \setminus \Omega_\epsilon$, find $\mathbf{v}' \in \Omega_\epsilon$ satisfying $v'_1 \geq v_1$ and $v'_i \geq v_i - \epsilon$ for $i > 1$

```
Let b be the constraints in the root node, i.e., b_i = −∞ for i > 1 ;
repeat
    c ← Φ(b), v' ← U^d(c), b' ← b;
    for each objective i > 1 do
        if v'_i + ε ≤ v_i then
            b_i ← v'_i + ε ;
            break;
until b = b';
return Φ(b) ;
```

## 4.4 MILP Approach

In Section 4.3, we introduced a high level search algorithm for generating the Pareto frontier by producing a sequence of CSOPs. In this section we present an exact approach for defining and solving a mixed-integer linear program (MILP) formulation of a CSOP for MOSGs. In Section

4.5, we go on to show how heuristics that exploit the structure and properties of security games can be used to improve the efficiency of our MILP formulation.

As stated in Section 4.3, to ensure the Pareto optimality of solutions, lexicographic maximization is required to sequentially maximize all the objective functions while still respecting the constraints in $\mathbf{b}$. Thus, for each CSOP we must solve $n$ MILPs, where each MILP is used to maximize one objective. For the $\lambda^{th}$ MILP in the sequence, the variable $d_\lambda$ is maximized, which represents the defender's payoff for security game / objective $\lambda$. This MILP is constrained by having to maintain the maximized values $d_j^*$ for $1 \leq j < \lambda$ found by previous MILPs in the sequence as well as satisfy lower bound constraints $b_k$ for $\lambda < k \leq n$ corresponding to the remaining uncomputed MILPs in the sequence.

We present our MILP formulation for a CSOP for MOSGs in Figure 4.5. This is similar to the MILP formulations for security games presented in [Kiekintveld et al., 2009] and elsewhere with the exception of the key Equations 4 and 5. Equation 1 is the objective function, which maximizes the defender's payoff for objective $\lambda$, $d_\lambda$. In Equations 2 and 3, $M$ is a large constant relative to the maximum payoff value for any objective. Equation 2 defines the defender's expected payoff $d_i$ for each objective $i$ based on the target selected by attacker type $i$. The constraint places an upper bound of $U_i^d(c_t, t)$ on $d_i$, but only for the attacked target. For every other target, $M$ on the right hand side causes the constraint to be arbitrarily satisfied.

Similarly, Equation 3 defines the expected payoff $k_i$ for attacker type $i$ based on the target selected for attack. The first part of the constraint specifies that $k_i - U_i^a(c_t, t) \geq 0$, which implies that $k_i$ must be at least as large as the maximal payoff for attacking any target. The second part forces $k_i - U_i^d(c_t, c) \leq 0$ for the target selected by attacker type $i$. If the selected target is not maximal, this constraint is violated.

$$
\begin{aligned}
\max \quad & d_\lambda && (4.1)\\
1 \le i \le n, \forall t \in T: \quad & d_i - U_i^d(c_t, t) \le M(1 - a_i^t) && (4.2)\\
1 \le i \le n, \forall t \in T: \quad & 0 \le k_i - U_i^a(c_t, t) \le M(1 - a_i^t) && (4.3)\\
1 \le j < \lambda: \quad & d_j = d_j^* && (4.4)\\
\lambda < k \le n: \quad & d_k \ge b_k && (4.5)\\
1 \le i \le n, \forall t \in T: \quad & a_i^t \in \{0, 1\} && (4.6)\\
\forall j \in A: \quad & \textstyle\sum_{t \in T} a_i^t = 1 && (4.7)\\
\forall t \in T: \quad & 0 \le c_t \le 1 && (4.8)\\
& \textstyle\sum_{t \in T} c_t \le m && (4.9)
\end{aligned}
$$

Figure 4.5: Lexicographic MILP formulation for a CSOP.

Taken together, Equations 1-3 imply that the strategies for both the defender and attacker type $\lambda$ are best-responses with respect to each other. However, the same cannot be said about the defender's strategy with respect to all of the other attacker types because the defender's payoffs for those objectives are not included in the objective function. It is for this reason that lexicographic maximization is necessary, ensuring that defender strategy is the best response with respect to all attacker types and the constraints in $\mathbf{b}$.

Equation 4 constrains the feasible region to solutions that maintain the values of objectives maximized in previous iterations of the lexicographic maximization. Equation 5 guarantees that the lower bound constraints in $\mathbf{b}$ will be satisfied for all objectives which have yet to be optimized.

If a mixed strategy is optimal for the attacker, then so are all the pure strategies in the support of that mixed strategy. Thus, we only consider the pure strategies of the attacker [Paruchuri et al., 2008]. Equations 6 and 7 constrain attackers to pure strategies that attack a single target. Equations (8) specifies that the coverage for each target $c_t$ is in the range [0,1]. Finally, Equation 9 ensures the amount of defender coverage used is no greater than the total number of defender resources, $m$.

| Variable | Definition | Dimension |
|---|---|---|
| $\lambda$ | Current Objective | $-$ |
| $m$ | Number of Defender Resources | $-$ |
| $n$ | Number of Attacker Types | $-$ |
| $Z$ | Huge Positive Constant | $-$ |
| $T$ | Set of Targets | $\|T\|$ |
| $\mathbf{a}$ | Attacker Coverage $a_j^t$ | $n \times \|T\|$ |
| $\mathbf{b}$ | Objective Bounds $b_j$ | $(n-1) \times 1$ |
| $\mathbf{c}$ | Defender Coverage $c_t$ | $\|T\| \times 1$ |
| $\mathbf{d}$ | Defender Payoff $d_j$ | $n \times 1$ |
| $\mathbf{d}^*$ | Maximized Defender Payoff $d_j^*$ | $n \times 1$ |
| $\mathbf{k}$ | Attacker Payoff $k_j$ | $n \times 1$ |
| $U^d$ | Defender Payoff Structure $U_j^d(c_t, t)$ | $n \times \|T\|$ |
| $U^a$ | Attacker Payoff Structure $U_j^a(c_t, t)$ | $n \times \|T\|$ |

Figure 4.6: MILP formulation definitions for a CSOP.

As noted earlier, this MILP is a modified version of the optimization problem formulated in [Kiekintveld et al., 2009] and is specific for security games. Similar modifications can be made to more generic Stackelberg games, such as those used for the Decomposed Optimal Bayesian Stackelberg Solver (DOBSS) [Paruchuri et al., 2008], giving a formulation for generalized multi-objective Stackelberg games beyond security games.

## 4.5 Improving MILP Efficiency

Once the MILP has been formulated as specified in Section 4.4, it can be solved using an optimization software package such as CPLEX. It is possible to increase the efficiency of the MILP formulation by using heuristics to constrain the decision variables. A simple example of a general heuristic which can be used to achieve speedup is placing an upper bound on the defender's payoff for the primary objective. Assume $d_1$ is the defender's payoff for the primary objective in the parent CSOP and $d_1'$ is the defender's payoff for the primary objective in the child CSOP. As each CSOP is a maximization problem, it must hold that $d_1 \geq d_1'$ because the child CSOP is

more constrained than the parent CSOP. Thus, the value of $d_1$ can be passed to the child CSOP to be used as an upper bound on $d'_1$.

In addition to placing bounds on the defender payoff, it is possible to constrain the defender coverage in order to improve the efficiency of our MILP formulation. Thus, we introduce three approaches for translating constraints on defender payoff into constraints on defender coverage. These approaches (ORIGAMI-M, ORIGAMI-M-BS, and DIRECT-MIN-COV) achieve this translation by computing the minimum coverage needed to satisfy a set of lower bound constraints $\mathbf{b}$ such that $U_i^d(\mathbf{c}) \geq b_i$, for $1 \leq i \leq n$. This minimum coverage is then added to the MILP in Figure 4.5 as constraints on the variable $\mathbf{c}$, reducing the feasible region and leading to significant speedup as verified in experiments.

### 4.5.1 ORIGAMI-M

ORIGAMI-M (Algorithm 3), is a modified version of the ORIGAMI algorithm [Kiekintveld et al., 2009] and borrows many of its key concepts. The "M" in the algorithm name refers to the fact that ORIGAMI-M is designed for security games with multiple objectives. At a high level, ORIGAMI-M starts off with an empty defender coverage vector $\mathbf{c}$, a set of lower bound constraints $\mathbf{b}$, and $m$ defender resources. The goal is to update $\mathbf{c}$ such that it uses the minimum amount of defender resources to satisfy the constraints in $\mathbf{b}$. If a constraint $b_i$ is violated, i.e., $U_i^d(\mathbf{c}) < b_i$, ORIGAMI-M updates $\mathbf{c}$ by computing the minimum additional coverage necessary to satisfy $b_i$. Since we focus on satisfying the constraints one objective at a time, the constraints for other objectives that were satisfied in previous iterations may become unsatisfied again. The reason is that the additional coverage may alter the targets selected for attack by one or more

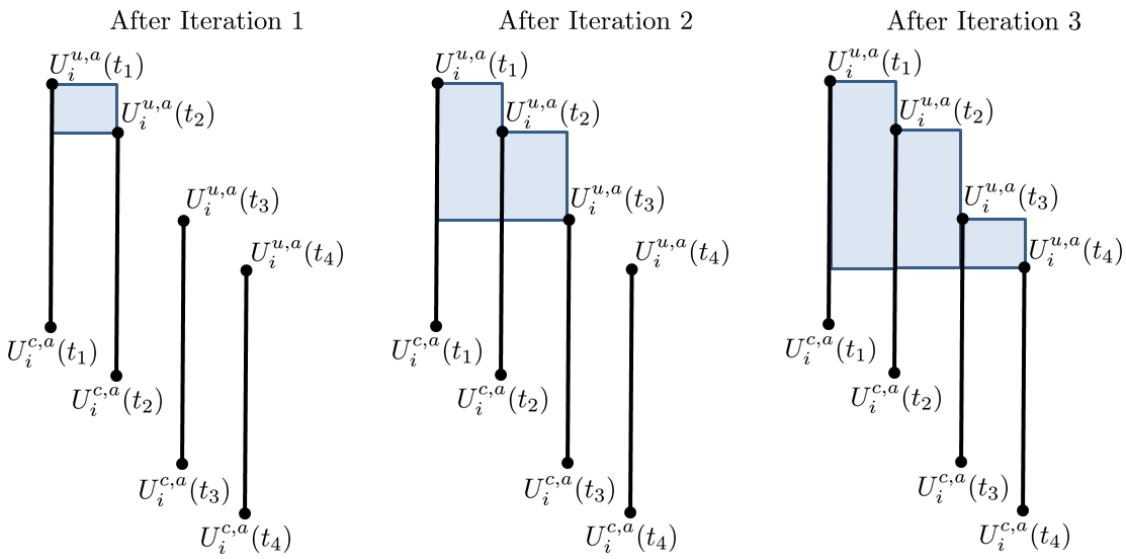



Figure 4.7: Example of ORIGAMI-M incrementally expanding the attack set by increasing coverage.

attacker types, possibly reducing the defender's payoff for those objectives below their once satisfied constraints. Therefore, all of the constraints in $\mathbf{b}$ must be checked repeatedly until there are no violated constraints. If all $m$ defender resources are exhausted before $\mathbf{b}$ is satisfied, then the CSOP is infeasible.

The process for calculating the minimum coverage for a single constraint $b_i$ is built on two assumption of security games [Kiekintveld et al., 2009]: (1) the attacker chooses the target that is optimal with respect its own payoffs; (2) if multiple targets are optimal, the attacker breaks ties by choosing the target that yields the highest defender payoff. The first property intuitively establishes that the attacker is a fully rational decision maker. The second property may seem less intuitive given the adversarial nature of the defender and the attacker. In theory, the player acting first in a Stackelberg game may force the adversary to play specific inducible actions in the follower's optimal set of actions by the threat of a slight perturbation of the optimal strategy, as described in [von Stengel and Zamir, 2004]. In practice, the assumption that the attacker breaks

ties in favor of the defender has been used in a number of real-world applications of Stackelberg security games. There has been work to remove these assumptions with models that consider uncertainty about the attacker, such as the imperfect rationality of human decision making [Pita et al., 2009; Yang et al., 2012]. However, we focus on the base model with standard assumptions for our initial multi-objective work and leave extensions for handling these types of uncertainty to future work.

The set of optimal targets for attacker type $i$, given coverage $\mathbf{c}$, is referred to as the attack set, $\Gamma_i(\mathbf{c})$. Accordingly, adding coverage on target $t \notin \Gamma_i$ does not affect the attacker type $i$'s strategy or payoff. Thus, if $\mathbf{c}$ does not satisfy $b_i$, we only consider adding coverage to targets in $\Gamma_i$. $\Gamma_i$ can be expanded by increasing coverage such that the payoff for each target $t \in \Gamma_i$ is equivalent to the payoff for the target $t' \notin \Gamma_i$ with the highest payoff as defined by $U_i^a(c_{t'}, t')$. Adding an additional target to the attack set can only benefit the defender since the defender receives the optimal payoff among targets in the attack set.

Figure 4.7 shows a simple example of ORIGAMI-M with four targets. The vertical axis is the payoff for attacker type $i$, $U_i^a(\mathbf{c})$, while each target $t$ is represented as the range $[U_i^{c,a}(t), U_i^{u,a}(t)]$. The blue rectangles depict the amount of coverage placed on each target. Before Iteration 1, the targets are sorted in descending order according to $U_i^a(\mathbf{c})$, resulting in the ordering $t_1 > t_2 > t_3 > t_4$ as well as $\Gamma_i = \{t_1\}$. After Iteration 1, enough coverage has been added to $t_1$ that $U_i^a(c_1, t_1) = U_i^{u,a}(t_2)$, meaning $\Gamma_i$ has been expanded to include $t_2$. In Iteration 2, coverage is placed on both $t_1$ and $t_2$ in order to push attacker type $i$'s payoff for these targets down to $U_i^{u,a}(t_3)$, adding $t_3$ to $\Gamma_i$. The process is again repeated in Iteration 3 with coverage now being added to $t_1$, $t_2$, and $t_3$ until $t_4$ can be induced into $\Gamma_i$.

**Algorithm 3:** ORIGAMI-M($\mathbf{b}$)

```
c ← empty coverage vector ;
while b_i > U_i^d(c) for some bound b_i do
    sort targets T in decreasing order of value by U_i^a(c_t, t);
    covLeft ← m − Σ_{t∈T} c_t;
    next ← 2;
    while next ≤ |T| do
        addedCov[t] ← empty coverage vector;
        if max_{1≤t<next} U_i^{c,a}(t) > U_i^a(c_next, t_next) then
            x ← max_{1≤t<next} U_i^{c,a}(t);
            noninducibleNextTarget ← true;
        else
            x ← U_i^a(c_next, t_next);
        for 1 ≤ t < next do
            addedCov[t] ← (x − U_i^{u,a}(t)) / (U_i^{c,a}(t) − U_i^{u,a}(t)) − c_t;
        if Σ_{t∈T} addedCov[t] > covLeft then
            resourcesExceeded ← true;
            ratio[t] ← 1 / (U_i^{u,a}(t) − U_i^{c,a}(t)), ∀1 ≤ t < next;
            addedCov[t] = (ratio[t] · covLeft) / (Σ_{1≤t≤next} ratio[t]), ∀1 ≤ t < next;
        if U_i^d(c + addedCov) ≥ b_i then
            c′ ← MIN-COV(i, c, b, next);
            if c′ ≠ null then
                c ← c′;
            else
                c ← c + addedCov;
            break;
        else if resourcesExceeded ∨ noninducibleNextTarget then
            return infeasible;
        else
            c ← c + addedCov;
            covLeft −= Σ_{t∈T} addedCov[t];
            next++;
    if next = |T| + 1 then
        if covLeft > 0 then
            c ← MIN-COV(i, c, b, next);
            if c = null then
                return infeasible;
        else
            return infeasible;
return c ;
```

The idea for ORIGAMI-M is to expand the attack set $\Gamma_i$ until $b_i$ is satisfied. Targets are added to $\Gamma_i$ in descending order according to attacker payoff, $U_i^a(c_t, t)$, which requires sorting

the list of targets (Line 3). The attack set $\Gamma_i$ initially contains only the first target in this sorted list, while the variable $\mathrm{next}$ represents the size that the attack set will be expanded to. In order to add the $\mathrm{next}$ target to $\Gamma_i$, the attacker's payoff for all targets in $\Gamma_i$ must be reduced to $U_i^a(c_{\mathrm{next}}, t_{\mathrm{next}})$ (Line 12). However, it might not be possible to do this. Once a target $t$ is fully covered by the defender, there is no way to decrease the attacker's payoff below $U_i^{c,a}(t)$. Thus, if $\max_{1 \leq t < \mathrm{next}} U_i^{c,a}(t) > U_i^a(c_{\mathrm{next}}, t_{\mathrm{next}})$ (Line 8), then it is impossible to induce attacker type $i$ to choose target $t_{\mathrm{next}}$. In that case, we can only reduce the attacker's payoff for targets in the attack set to $\max_{1 \leq t < \mathrm{next}} U_i^{c,a}(t)$ (Line 9) and set the $\mathrm{noninducibleNextTarget}$ flag (Line 10). Then for each target $t \in \Gamma_i$, we compute the amount of additional coverage, $\mathrm{addedCov}[t]$, necessary to reach the required attacker payoff (Line 14). If the total amount of additional coverage exceeds the amount of remaining coverage (Line 15), denoted by variable $\mathrm{covLeft}$, then the $\mathrm{resourcesExceeded}$ flag is set (Line 16) and $\mathrm{addedCov}$ is recomputed with each target in $\Gamma_i$ being assigned a ratio of the remaining coverage so as to maintain the attack set (Line 18).

Once the $\mathrm{addedCov}$ vector has been computed, we check to see if $\mathbf{c} + \mathrm{addedCov}$ satisfies $b_i$ (Line 19). If it does, there may exist a coverage $\mathbf{c}'$ which uses less defender resources and still satisfies $b_i$. To determine if this is the case, we developed a subroutine called MIN-COV, described in detail below, to compute $\mathbf{c}'$ (Line 20). If $\mathbf{c}' = null$, then $\mathbf{c} + \mathrm{addedCov}$ is the minimum coverage which satisfies $b_i$ (Line 24), otherwise $\mathbf{c}'$ is the minimum coverage (Line 22). In either case, $\mathbf{c}$ is updated to the new minimum coverage and then compared against $\mathbf{b}$ to check for violated constraints (Line 2).

If $\mathbf{c} + \mathrm{addedCov}$ does not satisfy $b_i$, we know that further expansion of the attack set is necessary. Thus, $\mathbf{c}$ is updated to include $\mathrm{addedCov}$ (Line 29), the amount of coverage in $\mathrm{addedCov}$

is deducted from the running total of remaining coverage $\mathrm{covLeft}$ (Line 30), and $\mathrm{next}$ is incremented (Line 31). However, if either the $\mathrm{resourcesExceeded}$ or $\mathrm{noninducibleNextTarget}$ flag have been set (Line 26), then further expansion of the attack set is not possible. In this situation, $b_i$ as well as the CSOP are infeasible and ORIGAMI-M terminates. If the attack set is expanded to include all targets (Line 32), i.e., $\mathrm{next} = |T|+1$, then it may be possible to satisfy $b_i$ if there is still defender resources remaining. Thus, we update $\mathbf{c}$ to the output generated by calling MIN-COV. If $\mathbf{c} = null$, then $b_i$ is unsatisfiable and ORIGAMI-M returns $\mathrm{infeasible}$, otherwise $\mathbf{c}$ is the minimum coverage.

If $\mathbf{c}^*$ is the coverage vector returned by ORIGAMI-M then Equation (8) of our MILP formulation can be replaced with $c_t^* \leq c_t \leq 1, \forall t \in T$. If, instead, ORIGAMI-M returns $infeasible$ then there is no feasible solution that satisfies $\mathbf{b}$ and thus there is no need to attempt solving the CSOP with $\Phi$.

When MIN-COV (Algorithm 4) is called, we know that the coverage $\mathbf{c}$ induces an attack set of size $\mathrm{next}-1$ and does not satisfy $b_i$, while $\mathbf{c}+\mathrm{addedCov}$ induces an attack set of size $\mathrm{next}$ and satisfies $b_i$. Thus, MIN-COV is designed to determine if there exists a coverage $\mathbf{c}^*$ that uses more coverage than $\mathbf{c}$ and less coverage than $\mathbf{c}+\mathrm{addedCov}$ while still satisfying $b_i$. This determination can be made by trying to induce a satisfying attack on different targets and comparing the resulting coverage vectors. As $\mathbf{c} + \mathrm{addedCov}$ is the minimum coverage needed to induce an attack set of size $\mathrm{next}$, we only need to consider attacks on the first $\mathrm{next}-1$ targets. Thus, for each target $t_j$, $1 \leq j < \mathrm{next}$ (Line 5), we generate the coverage vector $\mathbf{c}'$ that induces an attack on $t_j$ and yields a defender payoff of at least $b_i$. MIN-COV returns $\mathbf{c}^*$ (Line 26), which represents the $\mathbf{c}'$ that uses the least amount of defender resources while satisfying $b_i$. The variable $\mathrm{minResources}$ denotes

**Algorithm 4:** MIN-COV$(i, \mathbf{c}, \mathbf{b}, \text{next})$

```
1  Input: Game index i, initial coverage c, lower bound b, size of expanded attack set next;
2  c* ← null;
3  minResources ← m;
4  baseCov ← Σ_{t∈T} c_t;
5  for 1 ≤ j < next do
6      feasible ← true;
7      c' ← c ;
8      c'_j ← (b_i − U_i^{u,a}(t_j)) / (U_i^{c,a}(t_j) − U_i^{u,a}(t_j));
9      c'_j ← max(c'_j, c_j);
10     if c'_j > 1 then
11         break;
12     covSoFar ← baseCov + c'_j − c_j;
13     for 1 ≤ k ≤ |T| do
14         if j ≠ k ∧ U_i^a(c'_{t_k}, t_k) > U_i^a(c'_{t_j}, t_j) then
15             c'_k = (U_i^a(c'_{t_j}, t_j) − U_i^{u,a}(t_k)) / (U_i^{c,a}(t_k) − U_i^{u,a}(t_k));
16             if c'_k < c_k ∨ c'_k > 1 then
17                 feasible ← false;
18                 break;
19             covSoFar += c'_k − c_k;
20             if covSoFar ≥ minResources then
21                 feasible ← false;
22                 break;
23     if feasible then
24         c* ← c';
25         minResources ← covSoFar ;
26 return c*
```

the amount of coverage used by the current minimum coverage and is initialized to $m$, the total number of defender resources.

For each coverage $\mathbf{c}'$, we initialize $\mathbf{c}'$ with $\mathbf{c}$ (Line 7) and then compute the coverage $c_j$ on target $t_j$ needed to yield a defender payoff of $b_i$ (Line 8). We can never remove any coverage that has already been placed, so we ensure that $c'_j \geq c_j$ (Line 9). If $c'_j > 1$, then no valid coverage of $t_j$ could satisfy $b_i$ and thus there is no need to compute $\mathbf{c}'$ for $t_j$. Otherwise, we update the coverage for every other target $t_k$, $1 \leq k \leq |T|$ $j \neq k$. Placing $c'_j$ coverage on $t_j$ yields an attacker payoff $U_i^a(c'_j, t_j)$. Since our goal is to induce an attack on $t_j$, we must ensure that the attacker payoff

for every $t_k$ is no greater than for $t_j$, i.e., $U_i^a(c'_j, t_j) \geq U_i^a(c'_k, t_k)$, by placing additional coverage (Line 15). If either $c'_k < c_k$ or $c'_k > 1$ (Line 16) then no feasible coverage $\mathbf{c}'$ exists for $t_j$. The variable $\mathrm{covSoFar}$ tracks the amount of resources used by $\mathbf{c}'$, if at any point this value exceeds $\mathrm{minResources}$ then $\mathbf{c}'$ for $t_j$ cannot be the minimum defender coverage (Line 20).

If the coverage for all targets $t_k$ is updated successfully then we know that: (1) $\mathbf{c}'$ satisfies $b_i$ and (2) $\mathbf{c}'$ is the current minimum coverage. For (1), we have ensured $t_j$ is in the attack set $\Gamma_i$. By the properties of security games, the attacker will select the target $t \in \Gamma_i$ that yields the highest defender payoff. Thus, in the worst case from the defender's perspective, $t = t_j$ and gives the defender a payoff of at least $b_i$. Since $\mathrm{covSoFar}$ is compared to $\mathrm{minResources}$ everytime the coverage for a target is updated, (2) is inherently true if all targets have been updated. Having found a new minimum coverage, we update $\mathbf{c}^* \leftarrow \mathbf{c}'$ and $\mathrm{minResources} \leftarrow \mathrm{covSoFar}$.

### 4.5.2 Binary Search ORIGAMI-M

The ORIGAMI-M algorithm expands the attack set $\Gamma_i$ one target at a time until either the current lower bound constraint is satisfied or determined to be infeasible. If the satisfying attack set is large, it may become computationally expensive to incrementally expand and evaluate the satisfiability of $\Gamma_i$. Thus, we introduced a modified version of ORIGAMI-M called ORIGAMI-M-BS (Algorithm 5)which uses binary search to find the minimum coverage vector $\mathbf{c}$ which satisfies the lower bound constraints in $\mathbf{b}$. Intuitively, for a violated constraint $i$, we are performing binary search to find the size of the smallest attack set which satisfies the lower bound constraint $b_i$. The natural range for the size of $\Gamma_i$ is between 1 and $|T|$, therefore we use the respective bounds $\mathrm{lower} = 0$ and $\mathrm{upper} = |T| + 1$ for our binary search. The size of the attack set to be evaluated is determined by $\mathrm{next} = (\mathrm{upper} + \mathrm{lower})/2$. We record the size of the smallest satisfying attack

set with $\mu$, which is initially set to $|T|+1$. The coverage vector corresponding to the smallest satisfying attack set is $\mathbf{c}^+$ and is initialized to $null$.

For an attack set of a given size, the procedure for placing coverage on targets is identical to the procedure in ORIGAMI-M. The set of targets is sorted in descending order according to attacker payoff, $U_i^a(c_t, t)$ (Line 3). Then it is necessary to compute the vector of additional coverage, $\text{addedCov}$, that must be added to the first $\text{next}-1$ targets so that $\Gamma_i$ is expanded to include $t_{\text{next}}$. There are three possible scenarios when evaluating an attack set: (1) An attack set of size $\text{next}$ cannot be induced due to either an insufficient amount of defender resources (Line 19) or a noninducible target (Line 12). Therefore, the smallest satisfying attack set must be smaller than size $\text{next}$ so we update $\text{upper}=\text{next}$ (Line 24). (2) An attack set of size $\text{next}$ can be induced but it does not satisfy the lower bound constraint $b_i$. Thus, we know that if a satisfying attack set exists it must be larger than size $\text{next}$ so we update $\text{lower}=\text{next}$ (Line 31). (3) An attack set of size $\text{next}$ can be induced and satisfies the lower bound constraint $b_i$ (Line 25). While the current attack set is a satisfying attack set, it may be possible to find a smaller attack set which also satisfies $b_i$. Thus, we update $\text{upper}=\text{next}$ (Line 26) and if the current attack set is the smallest satisfying attack set found so far we update $\mathbf{c}^+ = \mathbf{c}+\text{addedCov}$ (Line 27) and $\mu=\text{next}$ (Line 28).

The binary search loop is repeated while $\text{upper}-\text{lower}>1$ (Line 9). After loop termination, if $\mathbf{c}^+=null$ and $\text{upper}<|T|+1$ (Line 32), then the constraint $b_i$ is not satisfiable and the CSOP is infeasible (Line 39). We know this because $\text{upper}$ is updated whenever an attack set either satisfies $b_i$ (Line 26), exceeds the available resources (Line 24), and/or contains a noninducible target (Line 24). Thus, $\text{upper} < |T|+1$ would indicate that at least one attack set was found to exceed defender resources or contain a noninducible target, but no satisfying attack set was

**Algorithm 5:** ORIGAMI-M-BS($\mathbf{b}$)

```
c ← empty coverage vector ;
while b_i > U_i^d(c) for some bound b_i do
    sort targets T in decreasing order of value by U_i^a(c_t, t);
    covLeft ← m − Σ_{t∈T} c_t;
    lower ← 0;
    upper ← |T| + 1;
    μ ← |T| + 1;
    c^+ ← null;
    while upper − lower > 1 do
        next = (upper + lower)/2;
        addedCov[t] ← empty coverage vector;
        if max_{1≤t<next} U_i^{c,a}(t) > U_i^a(c_next, t_next) then
            x ← max_{1≤t<next} U_i^{c,a}(t);
            noninducibleNextTarget ← true;
        else
            x ← U_i^a(c_next, t_next);
        for 1 ≤ t < next do
            addedCov[t] ← (x − U_i^{u,a}(t)) / (U_i^{c,a}(t) − U_i^{u,a}(t)) − c_t;
        if Σ_{t∈T} addedCov[t] > covLeft then
            resourcesExceeded ← true;
            ratio[t] ← 1 / (U_i^{u,a}(t) − U_i^{c,a}(t)), ∀1 ≤ t < next;
            addedCov[t] = ratio[t]·covLeft / Σ_{1≤t≤next} ratio[t], ∀1 ≤ t < next;
        if resourcesExceeded ∨ noninducibleNextTarget then
            upper = next;
        if U_i^d(c + addedCov) ≥ b_i then
            upper = next;
            if next < μ then
                c^+ ← c + addedCov;
                μ ← next;
        else
            lower = next;
    if c^+ ≠ null ∨ upper = |T| + 1 then
        c' ← MIN-COV(i, c, b, μ);
        if c' ≠ null then
            c ← c';
        else
            c ← c^+;
    else
        return infeasible;
return c;
```

found given that $\mathbf{c}^+ = null$. However, if $\mathbf{c}^+ = null$ and $\text{upper} = |T|+1$, then it is still possible that a coverage satisfying $b_i$ exists because it means the attack set has been expanded to the full set of targets and there is still remaining coverage. In this situation, as well as when $\mathbf{c}^+ \neq null$, MIN-COV is called to produce a coverage $\mathbf{c}'$ (Line 33). If $\mathbf{c}' \neq null$, then $\mathbf{c}'$ is the minimum coverage which satisfies $b_i$ and we update $\mathbf{c} \leftarrow \mathbf{c}'$ (Line 35). Otherwise, the coverage $\mathbf{c}^+$ found during the binary search is the minimum coverage and we update $\mathbf{c} \leftarrow \mathbf{c}^+$ (Line 37). The updated $\mathbf{c}$ is then checked for violated constraints (Line 2 ) and the entire process is repeated until either all constraints are satisfied or $\mathbf{b}$ is determined to be infeasible.

### 4.5.3 Direct MIN-COV

Both ORIGAMI-M and ORIGAMI-M-BS rely on the MIN-COV subroutine which is called when the smallest satisfying attack set is found. However, it is not necessary to first compute the satisfying attack set before calling MIN-COV. The only benefit of precomputing the attack set is to reduce the number of coverage vectors that must be computed in MIN-COV. The minimum coverage for satisfying $\mathbf{b}$ can be computed directly using MIN-COV, if we set the size of the attack set to be $|T| + 1$. In this way, MIN-COV will generate, for every target $t$, the coverage necessary to induce a satisfying attack on $t$. These coverages will be compared and the smallest, feasible, satisfying coverage will be selected. Thus, we introduced DIRECT-MIN-COV (Algorithm 6) which bypasses computing the smallest satisfying attack set and uses MIN-COV to compute the minimum coverage $\mathbf{c}$ needed to satisfy $\mathbf{b}$. Additionally, due to every target being considered for an attack there is no need to sort the targets by $U_i^a(c_t, t)$, as in ORIGAMI-M and ORIGAMI-M-BS. In all three algorithms (ORIGAMI-M, ORIGAMI-M-BS, and DIRECT-MIN-COV), MIN-COV is called only once for each violated constraint, the only difference being the number of

coverage vectors computed. Despite DIRECT-MIN-COV having to generate more coverages via MIN-COV than either ORIGAMI-M or ORIGAMI-M-BS, the intuition is that there could be potential computational savings in not having to first compute $\Gamma_i$. As we show in Section 4.7, the fastest algorithm for computing lower bounds on the defender coverage depends on the specific properties of the MOSG such as the number of resources and targets.

**Algorithm 6:** DIRECT-MIN-COV($\mathbf{b}$)

```
1 c ← empty coverage vector ;
2 while b_i > U_i^d(c) for some bound b_i do
3     c ←MIN-COV(i, c, b, |T| + 1);
4     if c = null then
5         return infeasible;
6 return c ;
```

## 4.6 Approximate Approach

In the previous section, we showed heuristics to improve the efficiency of our MILP approach. However, solving MILPs, even when constrained, is computationally expensive. Thus, we present ORIGAMI-A (Algorithm 7), an extension to these heuristics which eliminates the computational overhead of MILPs for solving CSOPs. The key idea of ORIGAMI-A is to translate a CSOP into a feasibility problem which can be solved using any one of the three algorithms described in Section 4.5. We will use $\Psi$ to refer to whichever algorithm (ORIGAMI-M, ORIGAMI-M-BS, or DIRECT-MIN-COV) is used as the subroutine in ORIGAMI-A. A series of feasibility problems is generated using binary search in order to approximate the optimal solution to the CSOP. This decomposition of the CSOP provides computational savings as we have developed efficient algorithms for solving the individual feasibility problems. Each of the three algorithms that can be used as a subroutine ($\Psi$) in ORIGAMI-A are polynomial in the number of targets,

while the number of calls to $\Psi$ by ORIGAMI-A is bounded by $\mathcal{O}(n \log r)$, where $r$ denotes the length of the range formed by the objective values. Thus, ORIGAMI-A is polynomial in the size of the MOSG, while solving even a single iteration of lexicographic maximization for the exact MILP formulation is NP-hard, based on the result from [Conitzer and Sandholm, 2006] which proved the computational complexity of Bayesian security games. As a result, this algorithmic approach is much more efficient and the level of approximation between the computed solution and the Pareto optimal solution can be bounded.

**Algorithm 7:** ORIGAMI-A$(\mathbf{b}, \alpha)$

```
1  c ← empty coverage vector;
2  b_1^+ ← min_{t∈T} U_1^{u,d}(t);
3  b^+ ← {b_1^+} ∪ b ;
4  for 1 ≤ i ≤ n do
5      lower ← b_i^+;
6      upper ← max_{t∈T} U_i^{c,d}(t);
7      while upper − lower > α do
8          b_i^+ ← (upper+lower)/2;
9          c' ← Ψ(b^+);
10         if c' = violated then
11             upper ← b_i^+;
12         else
13             c ← c';
14             lower ← b_i^+;
15     b_i^+ ← U_i^d(c);
16 return c ;
```

The subroutine $\Psi$ is used to compute the minimum coverage vector necessary to satisfy a set of lower bound constraints $\mathbf{b}$. As our MILP approach is an optimization problem, lower bounds are specified for the secondary objectives but not the primary objective. We can convert this optimization problem into a feasibility problem by creating a new set of lower bounds constraints $\mathbf{b}^+$ by adding a lower bound constraint $b_1^+$ for the primary objective to the constraints $\mathbf{b}$. We set $b_1^+ = \min_{t \in T} U_1^{u,d}(t)$, the lowest defender payoff for leaving a target uncovered. Now instead of

finding the coverage $\mathbf{c}$ which maximizes $U_1^d(\mathbf{c})$ and satisfies $\mathbf{b}$, we use $\Psi$ to determine if there exists a coverage vector $\mathbf{c}$ such that $\mathbf{b}^+$ is satisfied.

ORIGAMI-A finds an approximately optimal coverage vector $\mathbf{c}$ by using $\Psi$ to solve a series of feasibility problems. This series is generated by sequentially performing binary search on the objectives starting with initial lower bounds defined in $\mathbf{b}^+$. For objective $i$, the lower and upper bounds for the binary search are, respectively, $b_i^+$ and $\max_{t \in T} U_1^{c,d}(t)$, the highest defender payoff for covering a target. At each iteration, $\mathbf{b}^+$ is updated by setting $b_i^+ = (upper + lower)/2$ and then passed as input to $\Psi$. If $\mathbf{b}^+$ is found to be feasible, then the lower bound is updated to $b_i^+$ and $\mathbf{c}$ is updated to the output of $\Psi$, otherwise the upper bound is updated to $b_i^+$. This process is repeated until the difference between the upper and lower bounds reaches the termination threshold, $\alpha$. Before proceeding to the next objective, $b_i^+$ is set to $U_i^d(\mathbf{c})$ in case the binary search terminated on an infeasible problem. After searching over each objective, ORIGAMI-A will return a coverage vector $\mathbf{c}$ such that $U_1^d(\mathbf{c}^*) - U_1^d(\mathbf{c}) \leq \alpha$, where $\mathbf{c}^*$ is the optimal coverage vector for a CSOP defined by $\mathbf{b}$.

The solutions found by ORIGAMI-A are no longer Pareto optimal. Let $\Omega_\alpha$ be the objective space of the solutions found by ORIGAMI-A. We can bound its efficiency loss using the approximation measure $\rho(\epsilon, \alpha) = \max_{\mathbf{v} \in \Omega} \min_{\mathbf{v}' \in \Omega_\alpha} \max_{1 \leq i \leq n}(v_i - v_i')$.

**Theorem 5.** $\rho(\epsilon, \alpha) \leq \max\{\epsilon, \alpha\}$.

*Proof.* Similar to the proof of Theorem 4, for each point $\mathbf{v} \in \Omega$, we can use Algorithm 2 to find a CSOP with constraints $\mathbf{b}$ which is solved using ORIGAMI-A with coverage $\mathbf{c}$ such that (1) $b_i \leq v_i$ for $i > 1$ and (2) $v_i' \geq v_i - \epsilon$ for $i > 1$ where $\mathbf{v}' = U^d(\mathbf{c})$.

Assume that the optimal coverage is $\mathbf{c}^*$ for the CSOP with constraints $\mathbf{b}$. It follows that $U_1^d(\mathbf{c}^*) \geq v_1$ since the coverage resulting in point $\mathbf{v}$ is a feasible solution to the CSOP with constraints $\mathbf{b}$. ORIGAMI-A will terminate if the difference between lower bound and upper bound is no more than $\alpha$. Therefore, $v_1' \geq U_1^d(\mathbf{c}^*) - \alpha$. Combining the two results, it follows that $v_1' \geq v_1 - \alpha$.

Therefore, for any point missing in the frontier $\mathbf{v} \in \Omega$, we can find a point $\mathbf{v}' \in \Omega_\alpha$ such that 1) $v_1' \geq v_1 - \alpha$ and $v_i' \geq v_i - \epsilon$ for $i > 1$. It then follows that $\rho(\epsilon, \alpha) \leq \max\{\epsilon, \alpha\}$. ■ □

## 4.7 Evaluation

The purpose of this section is to analyze how the choice of approach and properties of MOSGs impact both the runtime and solution quality of Iterative-$\epsilon$-Constraints. We perform this evaluation by running the full algorithm in order to generate the Pareto frontier for randomly-generated MOSGs. For our experiments, the defender's covered payoff $U_i^{c,d}(t)$ and attacker's uncovered payoff $U_i^{u,a}(t)$ are uniformly distributed integers between 1 and 10, for all targets. Conversely, the defender's uncovered payoff $U_i^{u,d}(t)$ and attacker's covered payoff $U_i^{c,a}(t)$ are uniformly distributed integers between -1 and -10, for all targets. Unless otherwise mentioned, the default setup for each experiment is 3 objectives, 25 targets, $\epsilon = 1.0$, and $\alpha = 0.001$. The amount of defender resources $m$ is fixed at 20% of the number of targets. ORIGAMI-M is the default subroutine used in ORIGAMI-A. For experiments comparing multiple formulations, all formulations were tested on the same set of MOSGs. A maximum cap on runtime for each sample is set at 1800 seconds. We solved our MILP formulations using CPLEX version 12.1. The results were averaged over 30 trials and include error bars showing standard error.

### 4.7.1 Runtime Analysis

This section evaluates how different factors (e.g., the number of targets) impact the time needed to generate the Pareto frontier using five different formulations. We refer to the baseline MILP formulation as MILP-B. The MILP formulation adding a bound on the defender's payoff for the primary objective is MILP-P. MILP-M uses ORIGAMI-M to compute bounds on defender coverage. MILP-P can be combined with MILP-M to form MILP-PM. The algorithmic approach using ORIGAMI-A will be referred to by name. For analyzing the effect of the number of targets on runtime, we evaluate all five formulations for solving CSOPs. We then select ORIGAMI-A and the fastest MILP formulation, MILP-PM, to evaluate the effect of the remaining factors.

#### 4.7.1.1 Effect of the Number of Targets

This section presents results showing the efficiency of our different formulations as the number of targets is increased. In Figure 4.8, the x-axis represents the number of the targets in the MOSG. The y-axis is the number of seconds needed by Iterative-$\epsilon$-Constraints to generate the Pareto frontier using the different formulations for solving CSOPs. Our baseline MILP formulation, MILP-B, has the highest runtime for each number of targets we tested. By adding an upper bound on the defender payoff for the primary objective, MILP-P yields a runtime savings of 36% averaged over all numbers of targets compared to MILP-B. MILP-M uses ORIGAMI-M to compute lower bounds for defender coverage, resulting in a reduction of 70% compared to MILP-B. Combining the insights from MILP-P and MILP-M, MILP-PM achieves an even greater reduction of 82%. Removing the computational overhead of solving MILPs, ORIGAMI-A is the most efficient formulation with a 97% reduction. For 100 targets, ORIGAMI-A requires 4.53 seconds to generate the Pareto frontier, whereas the MILP-B takes 229.61 seconds, a speedup of greater than

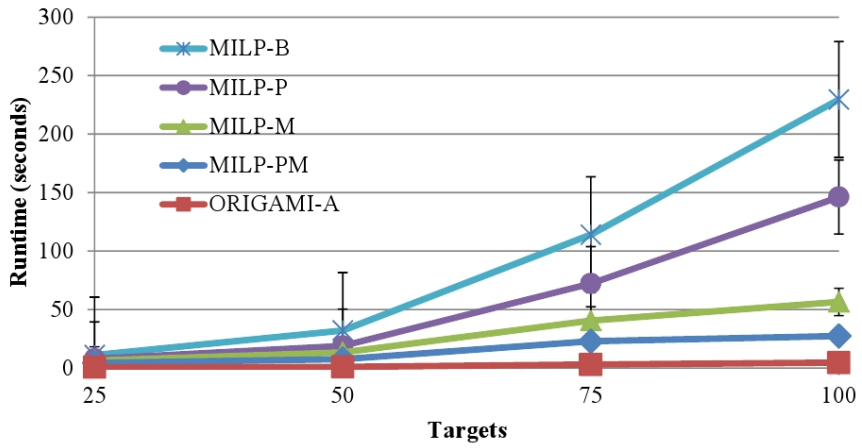



Figure 4.8: Effect of target scale up on the runtime of Iterative-$\epsilon$-Constraints with different CSOP solvers.

50 times. Even compared to fastest MILP formulation, MILP-PM at 27.36 seconds, ORIGAMI-A still achieves a 6 times speedup. Additionally, since a small $\alpha$ value is used (0.001), there is only negligible loss in solution quality. A more detailed analysis of solution quality is presented in Section 4.7.3. T-test yields p-value $< 0.001$ for all comparisons of different formulations when there are 75 or 100 targets.

We conducted an additional set of experiments to determine how both MILP-PM and ORIGAMI-A scale up for an order of magnitude increase in the number of targets by testing on MOSGs with between 200 and 1000 targets. Based on the trends seen in Figure 4.9, we can conclude that ORIGAMI-A significantly outperforms MILP-PM for MOSGs with large number of targets. Therefore, the number of targets in an MOSG is not a prohibitive bottleneck for generating the Pareto frontier using ORIGAMI-A.

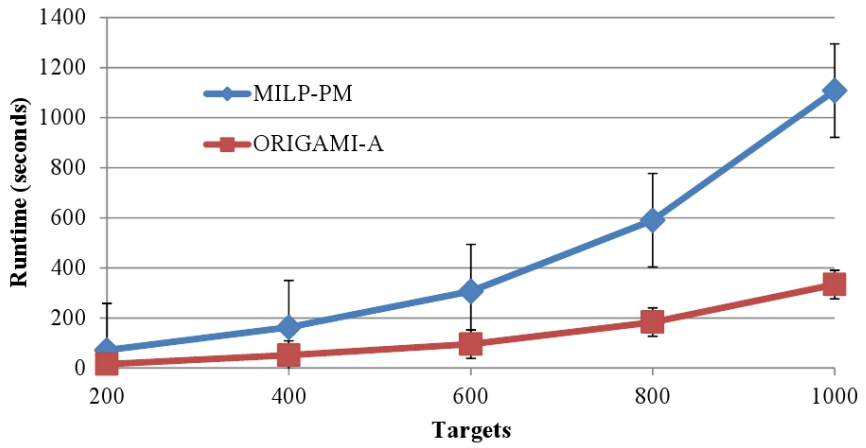



Figure 4.9: Effect of additional target scale up on the runtime of Iterative-$\epsilon$-Constraints with the most efficient exact CSOP solver (MILP-PM) and the approximate CSOP solver (ORIGAMI-A).

#### 4.7.1.2 Effect of the Number of Objectives

Another key factor on the efficiency of Iterative-$\epsilon$-Constraints is the number of objectives which determines the dimensionality of the objective space that Iterative-$\epsilon$-Constraints must search. We ran experiments for MOSGs with between 2 and 6 objectives. For these experiments, we fixed the number of targets at 10. Figure 4.10 shows the effect of scaling up the number of objectives. The x-axis represents the number of objectives, whereas the y-axis indicates the average time needed to generate the Pareto frontier. For both MILP-PM and ORIGAMI-A, we observe an exponential increase in runtime as the number of objectives is scaled up. For both approaches, the Pareto frontier can be computed in under 5 seconds for 2 and 3 objectives. At 4 objectives, the runtime increases to 126 seconds for MILP-PM and 28 seconds for ORIGAMI-A. With 5 objectives, the separation between the two algorithm increases with respective runtimes of 917 and 669 seconds, with 7 trials with MILP-PM and 6 trials with ORIGAMI-A timing out after 1800 seconds. Whereas, with 6 objectives neither approach is able to generate the Pareto frontier before the runtime cap of 1800 seconds. The reason for this exponential runtime increase is two-fold.

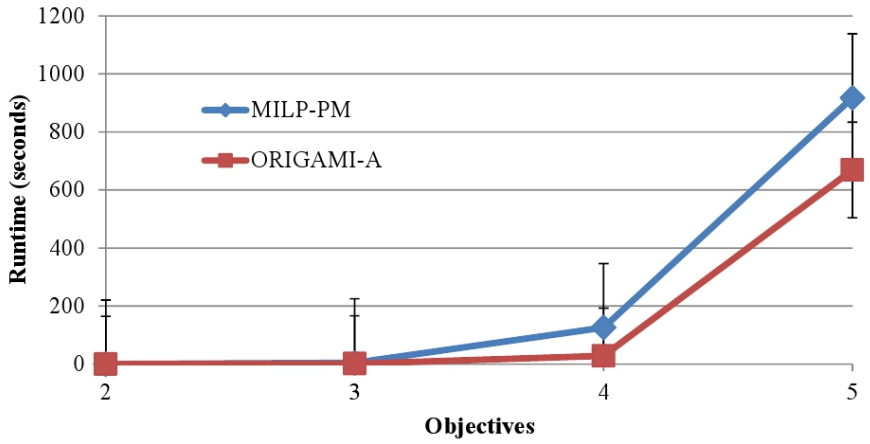



Figure 4.10: Effect of objective scale up on the runtime of Iterative-$\epsilon$-Constraints.

First, there is an increase in the number of generated solutions because the Pareto frontier now exists in a higher dimensional space. Second, each solution on the Pareto frontier takes longer to generate because the lexicographic maximization needed to solve a CSOP requires additional iterations. These results show that the number of objectives, and not the number of targets, is the key limiting factor in solving MOSGs.

#### 4.7.1.3 Effect of Epsilon

A third critical factor on the running time of Iterative-$\epsilon$-Constraints is the value of the $\epsilon$ parameter which determines the granularity of the search process through the objective space. In Figure 4.11, results are shown for $\epsilon$ values of 0.1, 0.25, 0.5, and 1.0. Both MILP-PM and ORIGAMI-A see a sharp increase in runtime as the value of $\epsilon$ is decreased due to the rise in the number of CSOPs solved. For example, with $\epsilon = 1.0$ the average Pareto frontier consists of 49 points, whereas for $\epsilon = 0.1$ that number increases to 8437. Due to the fact that $\epsilon$ is applied to the $n - 1$ dimensional objective space, the increase in the runtime resulting from decreasing $\epsilon$ is exponential

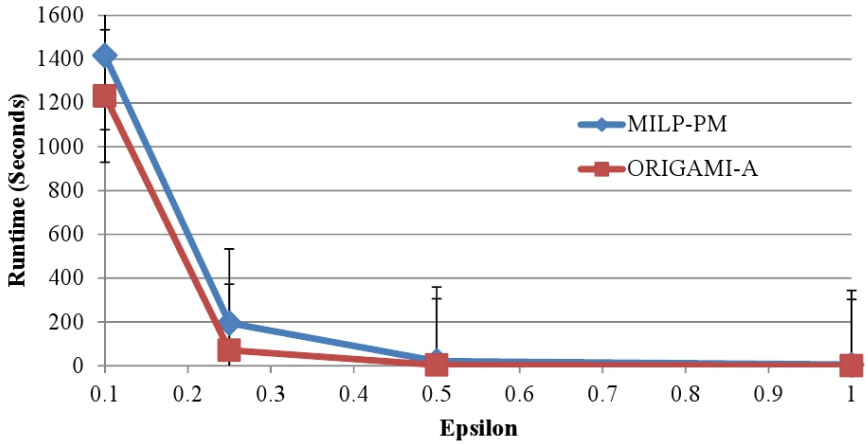



Figure 4.11: Effect of epsilon on the runtime of Iterative-$\epsilon$-Constraints.

in the number of secondary objectives. Thus, using small values of $\epsilon$ can be computationally expensive, especially if the number of objectives is large.

### 4.7.2 Objective Similarity Analysis

In previous experiments, all payoffs were sampled from a uniform distribution resulting in independent objective functions. However, it is possible that in a security setting, the defender could face multiple attacker types which share certain similarities, such as the same relative preferences over a subset of targets.

#### 4.7.2.1 Effect of Objective Distribution

As the objective payoffs become similar, there is less conflict between the objectives. Less conflict means there is a reduction in the possible tradeoff between objectives, as it becomes increasingly likely that multiple objectives will be maximized simultaneously. As a result, the Pareto frontier is made up of fewer solutions, which means it can be generated more efficiently by Iterative-$\epsilon$-Constraints.

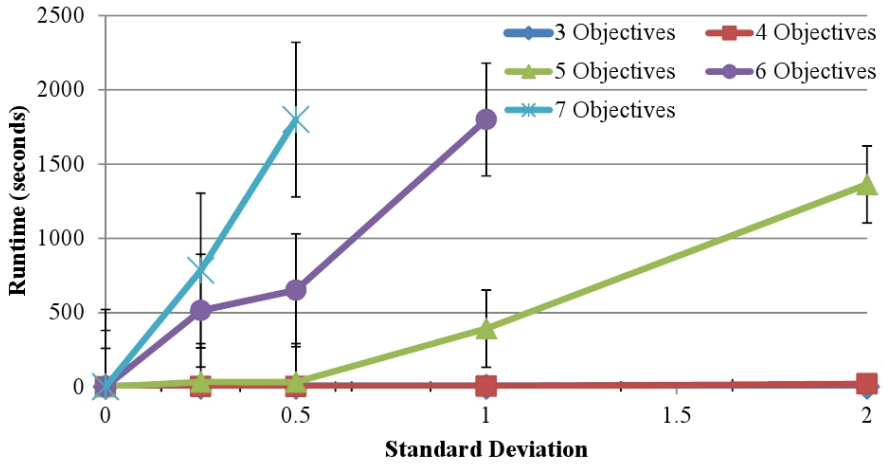



Figure 4.12: Effect of objective similarity on the runtime of Iterative-$\epsilon$-Constraints using ORIGAMI-A for a varying number of objectives.

To evaluate the effect of objective similarity on runtime, we used a single security game to create a Gaussian function with standard deviation $\sigma$ from which all the payoffs for an MOSG are sampled. Figure 4.12 shows the results for using ORIGAMI-A to solve MOSGs with between 3 and 7 objectives using $\sigma$ values of 0, 0.25, 0.5, 1.0, and 2.0. For $\sigma = 0$, the payoffs for all security games are the same, resulting in Pareto frontier consisting of a single point. In this extreme example, the number of objectives does not impact the runtime. However, as the number of objectives increases, less dissimilarity between the objectives is needed before the runtime starts increasing dramatically. For 3 and 4 objectives, the amount of similarity has negligible impact on runtime. With 5 objectives, a significant runtime increase is observed, going from an average of 32 seconds at $\sigma = 0.25$ to 1363 seconds at $\sigma = 2.0$. This effect is further amplified as the number of objectives is increased. At 6 objectives, Iterative-$\epsilon$-Constraints is unable to finish within the 1800 second time limit with $\sigma > 1.0$, while the same is true for 7 objectives with $\sigma > 0.5$. We conclude that it is possible to scale to larger number of objectives if there is similarity, as defined in this section, between the attacker types.

#### 4.7.2.2 Effect of Objective Clustering

In Section 4.7.2.1, the payoffs for each objective function are sampled from the same Gaussian distribution. This implies that all of the objective functions are related in their structure. However, there could be situations where one or more objectives are similar but other objectives are independently distributed. In this model, the set of related objectives can be viewed as forming a cluster while the remaining objectives are divergent from this cluster. A cluster is defined by two parameters. The first parameter is the number of objectives in the cluster as compared to the number of divergent objectives. A cluster size of 4 means that all of the objectives are in the cluster and thus all similar. In contrast, a cluster size of 1 implies that all objective functions are independently distributed. The second parameter is the value of $\sigma$ which is the standard deviation defining the Gaussian distribution from which the objectives in the cluster are drawn, i.e., the degree of similarity between the related objectives. In Figure 4.13, we show the runtime results for MOSGs with 4 objectives for different cluster sizes and values of $\sigma$. We observe a trend in which the average runtime rises as the value of $\sigma$ is increased. This is a logical result as larger values of $\sigma$ mean that there is greater dissimilarity between the objectives within the cluster. When the cluster size is between 2 and 4, increasing $\sigma$ always results in an increase in the runtime. When the cluster contains only 1 objective, the runtimes for all values of $\sigma$ are similar because all objectives are independently distributed.

Another trend we would expect to observe is that as the size of the cluster decreases, the runtime would increase as fewer objectives are similar and more are independently distributed. However, this trend only holds for $\sigma = 0$, when all of the objectives within the cluster are exactly identical. For $\sigma > 0$, we observe a substantially different runtime trend. With $\sigma = 1$ and $\sigma = 2$,

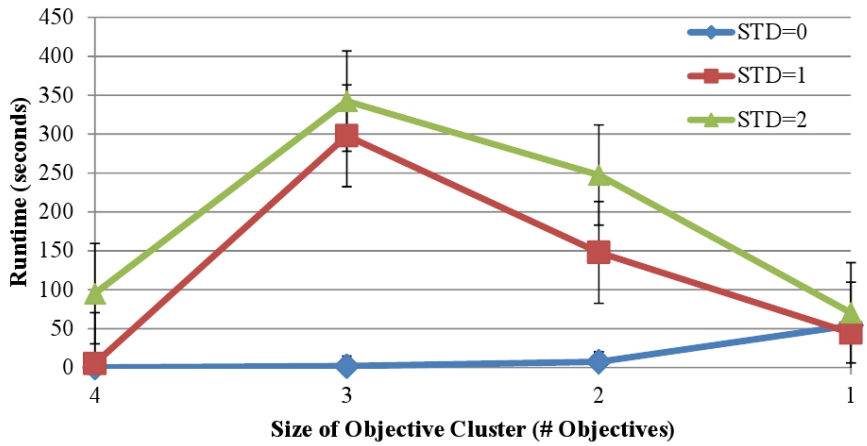



Figure 4.13: Effect of objective clustering size on the runtime of Iterative-$\epsilon$-Constraints using ORIGAMI-A for varying levels of intra-cluster Gaussian distribution.

the runtime starts low for clusters of size 4 and then increases dramatically when the size of the cluster is reduced to 3. Beyond 3 objectives, the runtime begins to decrease along with the cluster size until the runtime becomes similar for all values of $\sigma$ at cluster size 1. It is counterintuitive that the worst runtimes are achieved with three similar objectives and one independently distributed objective. Upon close analysis of the experiment output files, the increase in runtime is the result of solving more CSOPs and having a larger Pareto frontier. In Figure 4.14, we can see that a comparison of the number of solutions in the Pareto frontier closely resembles the trends seen in the comparison of runtimes. Thus, one possible hypothesis could be that having three somewhat related objectives and one independently distributed objective allows for greater tradeoff between the objective payoffs than four independently distributed objectives.

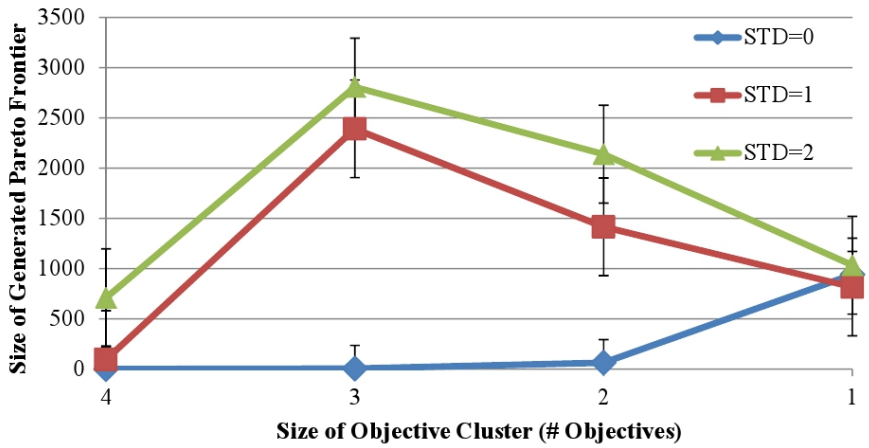



Figure 4.14: Effect of objective clustering on size of the Pareto frontier generated by Iterative-$\epsilon$-Constraints using ORIGAMI-A for varying levels of intra-cluster Gaussian distribution.

### 4.7.3 Solution Quality Analysis

#### 4.7.3.1 Effect of Epsilon

If the Pareto frontier is continuous, only a subset of that frontier can be generated. Thus, it is possible that one of the Pareto optimal points not generated by Iterative-$\epsilon$-Constraints would be the most preferred solution, were it presented to the end user. In Section 4.3.3, we proved that the maximum utility loss for each objective resulting from this situation could be bounded by $\epsilon$. We conducted experiments to empirically verify our bounds and to determine if the actual maximum objective loss was less than $\epsilon$.

Ideally, we would compare the Pareto frontier generated by Iterative-$\epsilon$-Constraints to the true Pareto frontier. However, the true Pareto frontier may be continuous and impossible for us to generate, thus we simulate the true frontier by using $\epsilon = 0.001$. Due to the computational cost associated with such a value of $\epsilon$, we fix the number of objectives to 2. Figure 4.15 shows the results for $\epsilon$ values of 0.25, 0.5, 0.75, and 1.0. The x-axis represent the value of $\epsilon$, whereas the y-axis represents the maximum objective loss when comparing the generated Pareto frontier to

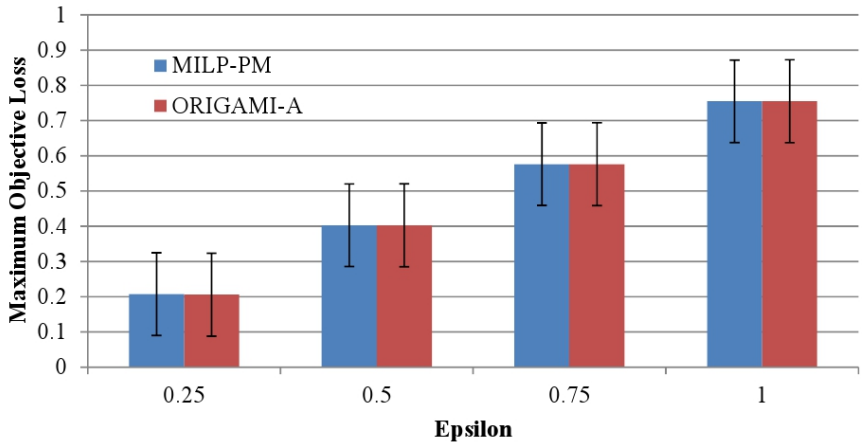



Figure 4.15: Effect of epsilon on solution quality of the Pareto frontier generated by Iterative-$\epsilon$-Constraints using MILP-PM and ORIGAMI-A compared against a Pareto frontier generated by MILP-PM using $\epsilon = 0.001$.

the true Pareto frontier. We observe that the maximum objective loss is less than $\epsilon$ for each value of $\epsilon$ tested. At $\epsilon = 1.0$, the average maximum objective loss is only 0.75 for both MILP-PM and ORIGAMI-A. These results verify that the bounds for our algorithms are correct and that in practice we are able to generate a better approximation of the Pareto frontier than the bounds would suggest.

#### 4.7.3.2 Comparison against Uniform Weighting

We introduced the MOSG model, in part, because it eliminates the need to specify a probability distribution over attacker types a priori. However, even if the probability distribution is unknown it is still possible to use the Bayesian security game model with a uniform distribution. We conducted experiments to show the potential benefit of using MOSG over Bayesian security games in such cases. We computed the maximum objective gain produced by using a point in the Pareto frontier generated by Iterative-$\epsilon$-Constraints as opposed to the Bayesian solution. If $\mathbf{v}'$ is the solution to a uniformly weighted Bayesian security game then the equation for maximum objective

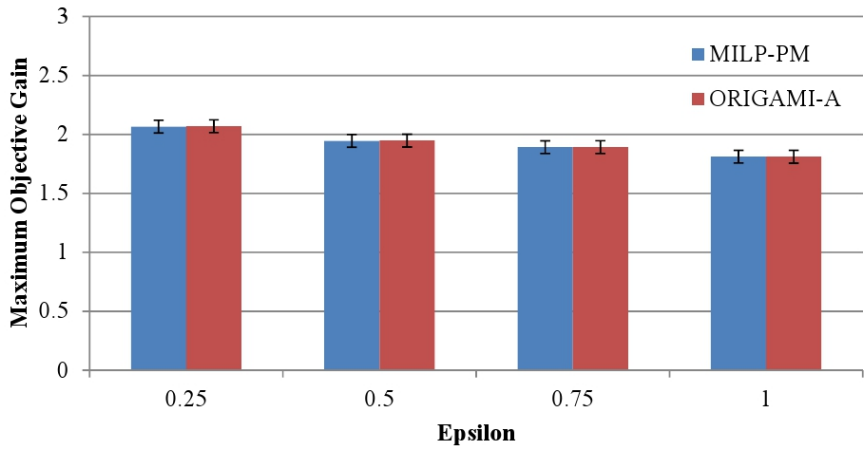



Figure 4.16: Effect of epsilon on the benefit of the Pareto frontier generated by Iterative-$\epsilon$-Constraints using MILP-PM and ORIGAMI-A over the single solution generated by a uniformly weighted Bayesian security game.

loss is $\max_{v \in \Omega_\epsilon} \max_i (v_i - v'_i)$. Figure 4.16 shows the results for $\epsilon$ values of 0.25, 0.5, 0.75, and 1.0. At $\epsilon = 1.0$, the maximum objective gain was 1.81 for both MILP-PM and ORIGAMI-A. Decreasing $\epsilon$ all the way to 0.25 increases the maximum objective gain by less than 15% for both algorithms. These results suggests that $\epsilon$ has limited impact on maximum objective gain, which is a positive result as it implies that solving an MOSG with a large $\epsilon$ can still yield benefits over a uniform weighted Bayesian security game.

### 4.7.4 Constraint Computation Analysis

A key property of the ORIGAMI-M algorithm is that it computes the minimum coverage satisfying a vector $\mathbf{b}$ of lower bound constraints by attempting to satisfy one constraint at a time until no violated constraints remain. In the process of computing the additional coverage needed to satisfy the current constraint it is possible that previously satisfied constraints could become violated. It is important to understand the frequency with which this phenomenon occurs as it can have serious implications for the efficiency of the algorithm. Thus, we performed experiments

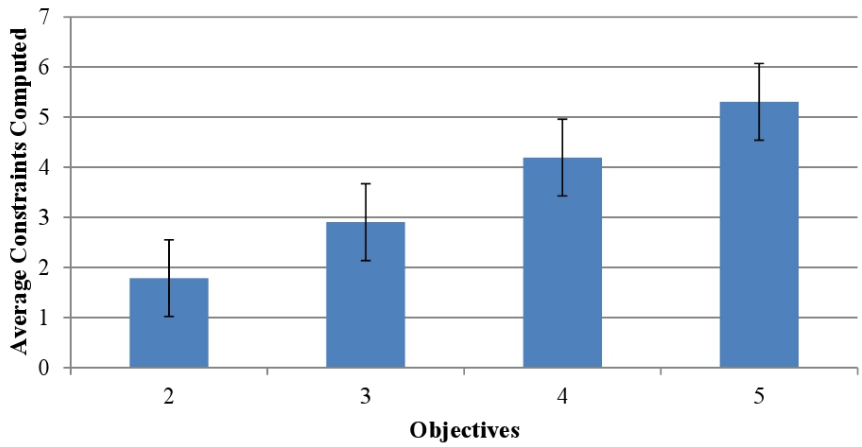



Figure 4.17: Effect of objective scale up on the number of constraints computed per call to ORIGAMI-M for Iterative-$\epsilon$-Constraints using ORIGAMI-A.

which recorded the number of constraints that had to be satisfied for each call to ORIGAMI-M. The number of constraints is inherently linked to the number of objectives, thus we tested how the number of constraints computed was affected when scaling up the number of objectives. Figure 4.17 shows the average number of computed constraints for MOSGs with between 2 and 5 objectives and 10 targets. With 2 objectives, the number of constraints computed is 1.78, implying that on average ORIGAMI-M finds the minimal coverage with one pass through the constraints. Additionally, it means that there are situations where solving the first constraint results in a coverage which also satisfies the second constraint. For MOSGs with 5 objectives, the average number of computed constraints is 5.3 which again implies that ORIGAMI-M mostly requires just one pass through the constraints. However, it also indicates that there are instances where previously satisfied constraints become violated and must be recomputed. Fortunately, these violated constraints appear to be infrequent and do not seem to produce a cascading effect of additional violated constraints. These results suggest that ORIGAMI-M is able to efficiently compute the minimum coverage and is capable of scaling up to larger number of objectives.

### 4.7.5 Improved Pruning

In Section 4.3.2, we introduced two sets of pruning rules to improve the efficiency of Iterative-$\epsilon$-Constraints. As shown in Section 4.7.1.2, the number of objectives is one of the key contributors to runtime when solving MOSGs. Thus, in order to perform a comparison, we evaluated each set of pruning heuristics as the number of objectives is increased. In Figure 4.18, we show results which demonstrate the impact of the improved pruning heuristic. The x-axis represents the number of objectives in the MOSG, while the y-axis represents the average runtime for Iterative-$\epsilon$-Constraints to compute the Pareto frontier. For MOSGs with 2 or 3 objectives, there is little difference in the average runtimes between the original and improved pruning heuristics. When the number of objectives is increased to 4, the benefit of the improved pruning heuristic emerges, reducing the average runtime from 34.5 to 23.1 seconds. At 5 objectives the improved pruning heursitic results in significant computational savings, reducing the average runtime by almost 28% (813.8 versus 588.7 seconds). Even with the improved set of pruning heuristics, Iterative-$\epsilon$-Constraintsis still not able to finish in under the 1800 second time limit. These results indicate that by further exploiting the concept of Pareto dominance, it is possible obtain modest runtime improvements.

### 4.7.6 ORIGAMI-A Subroutine Analysis

The ORIGAMI-A algorithm relies on ORIGAMI-M to compute the minimum coverage necessary to satisfy a set of lower bound constraints. ORIGAMI-M is a critical subroutine which is called multiple times for each CSOP, thus making efficiency paramount. In Figure 4.9, we showed the ability of ORIGAMI-M to scale up to large number of targets. However, any improvement to the subroutine used by ORIGAMI-A could lead to significant computation savings. Thus, in this

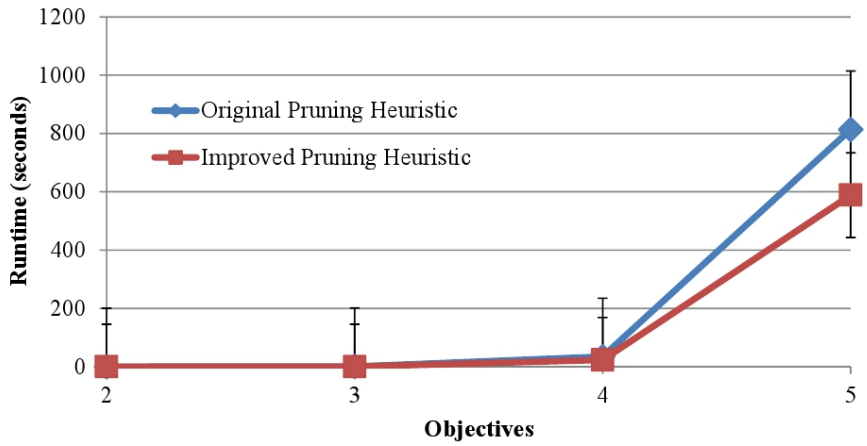



Figure 4.18: Effect of pruning heuristic on the runtime of Iterative-$\epsilon$-Constraints using ORIGAMI-A for a varying number of objectives.

section, we describe two approaches that either modify or replace ORIGAMI-M in an attempt to improve the efficiency of ORIGAMI-A.

#### 4.7.6.1 Comparing the Effect of the Number of Targets

In Figure 4.19, we compare the ability of both ORIGAMI-M-BS and DIRECT-MIN-COV to scale up the number of targets as opposed to ORIGAMI-M. We evaluated the three algorithms for MOSGs with between 200 and 1000 targets. The x-axis indicates the number of targets in the MOSG, whereas the y-axis represents the average time needed to generate the Pareto frontier. The runtime results for ORIGAMI-M-BS are counterintuitive, as the inclusion of binary search fails to provide any improvement over ORIGAMI-M. In fact, for every number of targets tested the runtime for ORIGAMI-M-BS is greater than ORIGAMI-M. The difference in runtime between the two algorithms remains essentially constant at 2 seconds for each number of targets tested.

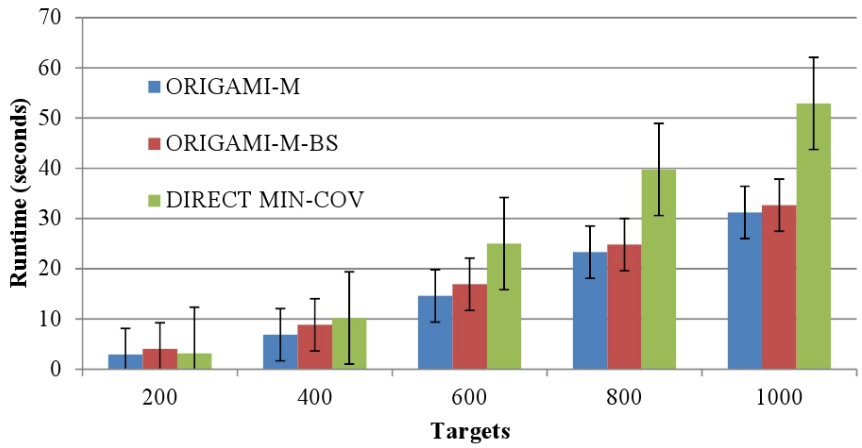



Figure 4.19: Effect of ORIGAMI-A subroutine on the runtime of Iterative-$\epsilon$-Constraints for a varying number of targets.

This result suggests that despite having different formulations, ORIGAMI-M and ORIGAMI-M-BS are evaluating a similar number of attack sets. Additionally, the runtimes for DIRECT-MIN-COV are worse than either ORIGAMI-M or ORIGAMI-M-BS for every number of targets tested, except for ORIGAMI-M-BS at 200 targets. As the number of targets is increased, the disparity between the runtimes for the two ORIGAMI-M algorithms and DIRECT-MIN-COV widens.

#### 4.7.6.2 Comparing the Effect of the Ratio of Defender Resources to Targets

We sought to better understand why neither of the two new proposed algorithms were able to improve upon the performance of ORIGAMI-M. In particular, we wanted to determine why incrementally expanding the attack set (ORIGAMI-M) was faster than performing binary search (ORIGAMI-M-BS), even for MOSGs with 1000 targets.

For all of our experiments, the ratio of defender resources to targets was fixed at $\frac{m}{|T|} = 0.2$. Intuitively, the higher this ratio is, the larger the average size of the attack set will be. With relatively more resources, the defender can place additional coverage so as to induce the attacker into

considering a larger number of targets. Thus, the small $\frac{m}{|T|}$ ratio that we had been using previously meant the average size of the attack set would also be small. This greatly favors ORIGAMI-M which expands the attack set one target at time and returns as soon as it has found a satisfying attack set. In contrast, ORIGAMI-M-BS always evaluates $\log n$ attack sets regardless of the $\frac{m}{|T|}$ ratio. To evaluate the effect of $\frac{m}{|T|}$ on the performance of our three algorithms, we conducted experiments on MOSGs with 400 targets and $\frac{m}{|T|}$ ratios ranging between 0.2 and 0.8. In Figure 4.20, we show the results for this set of experiments. The x-axis indicates the $\frac{m}{|T|}$ ratio, whereas the y-axis indicates the average time to generate the Pareto frontier. A clear pattern emerges from these results: (1) if $\frac{m}{|T|} < 0.5$ then the ordering of the algorithms from most to least efficient is ORIGAMI-M, ORIGAMI-M-BS, DIRECT-MIN-COV; (2) if $\frac{m}{|T|} \geq 0.5$ then the ordering is reversed to DIRECT-MIN-COV, ORIGAMI-M-BS, ORIGAMI-M. What is interesting is that ORIGAMI-M-BS is never the optimal algorithm. If $\frac{m}{|T|}$ is small then it is better to incrementally expanding the attack set using ORIGAMI-M, whereas when $\frac{m}{|T|}$ is large it is more efficient to not precompute the smallest satisfying attack set as in DIRECT-MIN-COV. This result suggests that the optimal subroutine for ORIGAMI-A is dependent on the underlying properties of the MOSG and thus could vary from domain to domain.

Additionally, there is a discernible trend across all three algorithms as the value of $\frac{m}{|T|}$ is varied. Specifically, the average runtime as a function of $\frac{m}{|T|}$ resembles a bell curve centered at $\frac{m}{|T|} = 0.6$. This is a result of the combinatorial nature of placing coverage on targets. Therefore, when $\frac{m}{|T|} = 0.2$ there are significantly more targets than defender resources and there is only so much that can be done to prevent attacks. Since there are fewer ways to configure the coverage, the Pareto frontier contains fewer solutions. At the other extreme, when $\frac{m}{|T|} = 0.8$ the amount of defender resources is essentially equivalent to the number of targets. It is then possible to generate

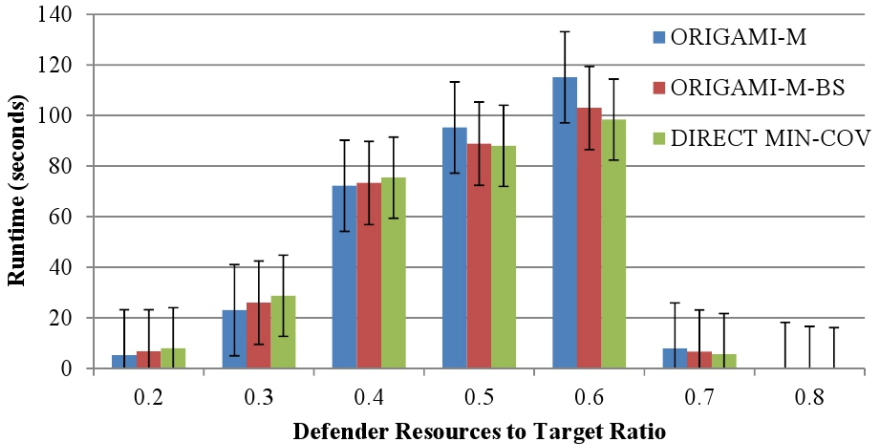



Figure 4.20: Effect of ORIGAMI-A subroutine on the runtime of Iterative-$\epsilon$-Constraints for varying resource-target ratios.

a coverage which maximizes all objectives simultaneously, leading to a Pareto frontier consisting of a single solution. Then as $\frac{m}{|T|}$ approaches $0.6$ from either direction the runtime increases as there are more ways to place coverage and thus more solutions in the Pareto frontier. Due to the large number of possible defender coverages to consider, each individual CSOP also takes longer to solve, which is a phenomenon that has also been observed in single objective security games as described in [Jain et al., 2012].

## 4.8 Visualization

The goal of our research is to provide decision support for decision-makers faced with multi-objective optimization problems. As mentioned previously, solving a multi-objective optimization problem involves generating the Pareto frontier. Once the Pareto frontier has been obtained, it must still be presented to the end user who then selects one of the candidate solutions based on their preferences, background knowledge, etc. One challenge associated with multi-objective optimization is how to present information about the Pareto frontier to the user so as to best facilitate

their decision-making process. The most naïve approach is to present the contents of the Pareto frontier in a tabular format. However, this approach suffers from one significant drawback, a lack of visualized spatial information. A table cannot convincingly convey the shape and structure of the Pareto frontier as well as the tradeoff between different objectives and solutions. Thus, visualization is an important component for presenting the Pareto frontier to the user.

In Section 4.1, we highlighted the Los Angeles rail system as a motivating domain for MOSGs. To recall, the LASD is responsible for protecting 70 stations in the rail system against three potential attacker types: ticketless travelers, criminals, and terrorists. We use the LASD domain as a case study to compare different methods for visualization in security domains, which is only possible using our algorithms for calculating the Pareto frontier.

We model the LASD domain as an MOSG with 3 objectives, 70 targets, and 14 defender resources. Iterative-$\epsilon$-Constraints with $\epsilon = 1.0$ was then used to generate the Pareto frontier which contained 100 solutions. It is this Pareto frontier that we use to compare the different visualization techniques.

### 4.8.1 Euclidean Plots

The elements of the Pareto frontier exist in an $n$-dimensional space, where $n$ is the number of objectives. Visualizing the Pareto frontier for $n = 2$ is intuitive as solutions can be represented in two-dimensional Euclidean space, as shown in Figure 4.2, by the payoffs obtained for each objective. This approach allows the tradeoff between the two objectives to be directly observed in a comprehensible form. An advantage of using Euclidean plots is that because the solutions are represented as points, the plots can display a large number of solutions without overwhelming the user. For $n = 3$ the Pareto frontier can still be plotted in Euclidean space. In Figure 4.21,

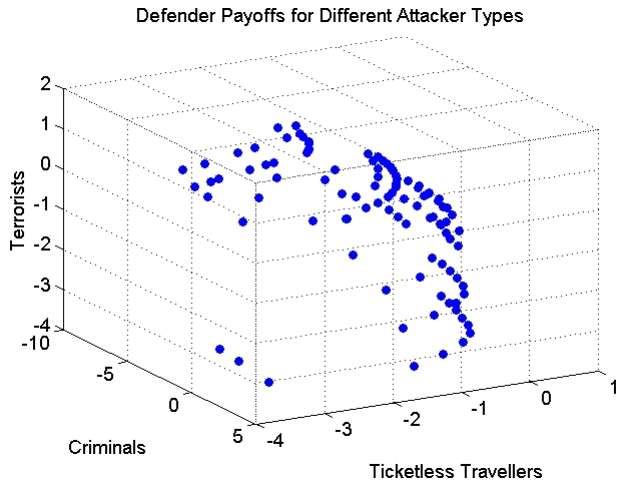



Figure 4.21: Euclidean plot of the Pareto frontier for the LASD domain.

the sample Pareto frontier from the LASD domain is visualized in three-dimensional Euclidean space. This example illustrates one of the drawbacks of using a Euclidean plot for $n = 3$. It is difficult to evaluate the tradeoffs in payoff for defending against ticketless travelers, criminals, and terrorists based on a single figure. Thus, interactive components such as animation or figure manipulation become necessary and present an additional barrier to the user's understanding.

### 4.8.2 Scatter Plots

One of the standard methods for visualizing the Pareto frontier is the scatter plot matrix [van Wijk and van Liere, 1993], where $n$ dimensions are visualized using $\binom{n}{2}$ two dimensional scatter plots, in which each pair of dimensions has a scatter plot showing their relation. With each scatter plot, the end user is able to gain a fundamental understanding of the tradeoffs between the payoffs for the two objectives. Similar to Euclidean plots, scatter plots are capable of efficiently displaying a large number of solutions. One extension on the standard bi-objective scatter plot is the addition of a third color dimension [Lotov et al., 2004], resulting in $\binom{n}{3}$ possible scatter plots. This color dimension can be represented as either a continuous gradient or as a discrete

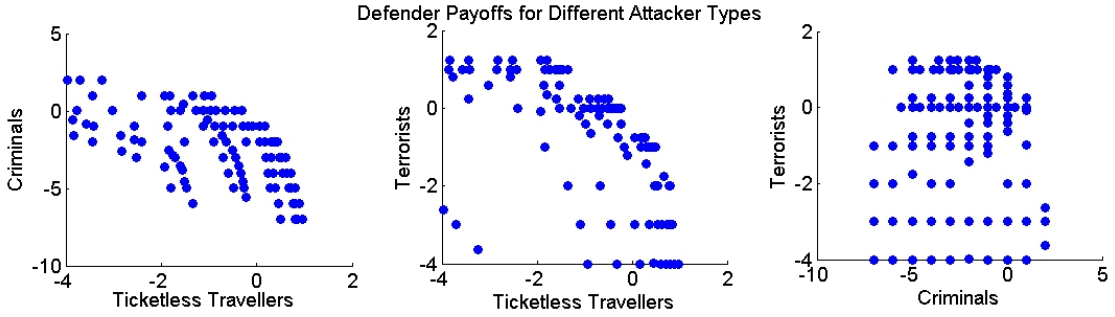



Figure 4.22: Bi-objective scatter plot matrix of the Pareto frontier for the LASD domain.

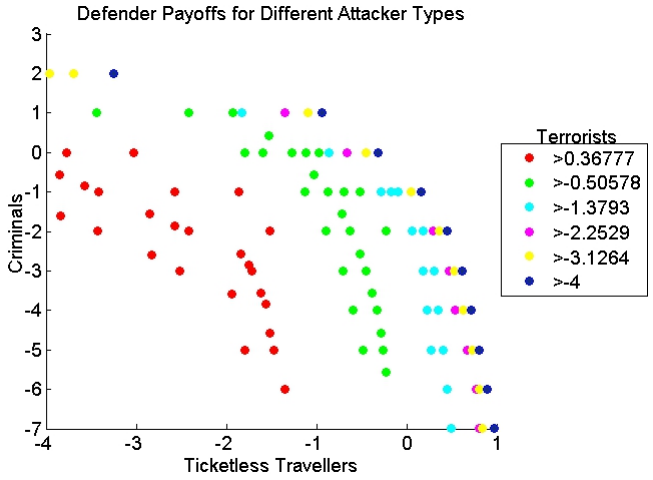



Figure 4.23: Tri-objective scatter plot matrix for the Pareto frontier for the LASD domain.

set of colors mapping to specific segments of the possible objective values. Examples of both bi-objective and tri-objective (with discrete coloring) scatter plots for the LASD domain can be seen in Figures 4.22 and 4.23, respectively. For the LASD domain, the tri-objective scatter plot matrix is preferable because the entire Pareto frontier can be visualized in a single figure, rather than the three figures required for the bi-objective scatter plot matrix. This eliminates the need for the end user to synthesize data between multiple scatter plots in order to obtain the global perspective. For both approaches, the decision making process becomes more difficult as the number of objectives is increased due to the polynomial number of scatter plots that must be generated.

### 4.8.3 Parallel Coordinates

Parallel Coordinates [Inselberg, 1997] is another common approach used for visualizing the Pareto frontier. In this approach, $n$ parallel lines are used to represent the range of values for each objective. A Pareto-optimal solution is displayed as a polyline that intersects each parallel line at the point corresponding to the payoff received for that objective. Figure 4.24 shows the Pareto frontier for the LASD domain using the Parallel Coordinates approach. The main advantage of Parallel Coordinates is that the entire Pareto frontier, regardless of dimensionality, can be presented in a single figure. This eliminates any issues associated with having to process data from multiple sources. However, due to the usage of polylines rather than points, the Pareto frontier can become incomprehensible to the user if the number of solutions in the Pareto frontier is large. This is an issue for the LASD domain because the Pareto frontier consists of 100 candidate solutions, making it difficult to distinguish each individual solution. The number of Pareto optimal solutions can be influenced during processing by adjusting the value of $\epsilon$ as well as during post-processing by employing a filter to prevent certain solutions from being displayed. However, the number of solutions may need to be dramatically reduced before the Pareto frontier becomes comprehensible.

### 4.8.4 Overall Trends

There is currently no one-size-fits-all visualization approach, the appropriate technique must be determined for each domain based on factors such as the number of objectives and the size of the Pareto frontier. For example, scatter plot matrices are better suited to situations where the dimensionality of the Pareto frontier is low but the number of solutions it contains is high, whereas

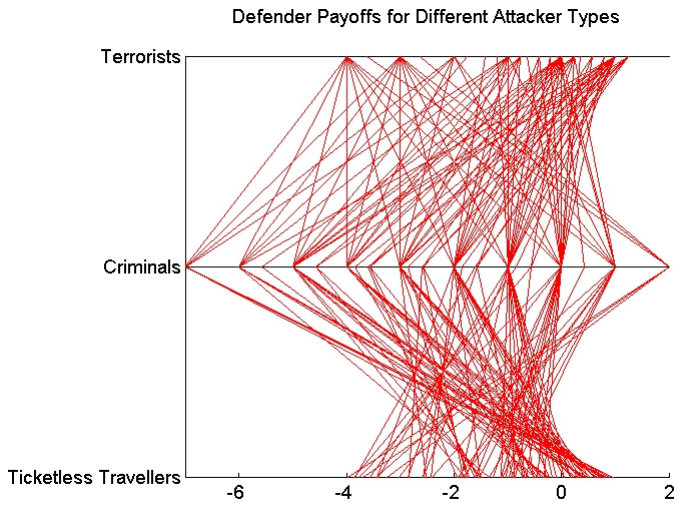



Figure 4.24: Parallel coordinates representation of the Pareto frontier for the LASD domain.

Parallel Coordinates is better suited to situations with high dimensionality but fewer candidate solutions.

Based on the properties of the domain, we conclude that tri-objective scatter plot is the best approach for visualizing the Pareto frontier of the LASD MOSG because it allows for the most compact and coherent visual representation. It captures the entire Pareto frontier in a single figure which should be intuitive even for non-technical decision makers. By generating and visualizing the Pareto frontier in this way, LASD can gain a significant amount of knowledge about their domain and the tradeoffs that exist between different security strategies. This can be more insightful than finding a single solution, even if it were generated using well thought out weightings for the objectives. Finally, since the tri-objective scatter plot does not rely on animation or manipulation, information about the Pareto frontier can be disseminated easily to large groups and included in printed reports.

We have demonstrated the ability to visualize the Pareto frontier for the LASD domain which has 3 objectives. As the dimensionality of the objective space increases, the Pareto frontier naturally becomes more complex and difficult to understand. However, for most multi-objective optimization problems the total number of objectives is relatively small ($n \leq 5$). Even for domains which require large number of objectives, it may be possible to reduce the dimensionality of the Pareto frontier in order to focus the decision making process only on the most salient objectives. Dimension reduction is possible in two situations: (1) some objectives are insignificant in that their range of Pareto-optimal values is small; (2) there exists a strong correlation between multiple objectives. This reduction is typically performed using machine learning techniques with the most common approach being Principal Component Analysis (PCA) [Jolliffe, 2002]. So if, in the future, LASD requires a higher fidelity model with more attacker types, it may become necessary to use such dimension reduction techniques in order to visualize the Pareto frontier.

## 4.9 Chapter Summary

We draw upon insights from game theory and multi-objective optimization to introduce a new model, multi-objective security games (MOSG), for domains where security forces must balance multiple objectives. Instead of a single optimal solution, MOSGs have a set of Pareto-optimal (non-dominated) solutions, known as the Pareto frontier, which represents the space of trade offs between the objectives. A single Pareto optimal solution can be found by solving a CSOP for a given set of constraints $\mathbf{b}$. The Pareto frontier is then generated by solving multiple CSOPs produced by modifying the constraints in $\mathbf{b}$. The contributions presented in this chapter include: (i) an algorithm, Iterative-$\epsilon$-Constraints, for generating the sequence of CSOPs; (ii) an exact

approach for solving an MILP formulation of a CSOP; (iii) heuristics that achieve speedup by exploiting the structure of security games to further constrain the MILP; (iv) an approximate approach for solving a CSOP built off those same heuristics, increasing the scalability of our approach with quality guarantees. Additional contributions of this chapter include proofs on the level of approximation, detailed experimental evaluation of the proposed approaches and heuristics, as well as a discussion on techniques for visualizing the Pareto frontier.

Now that we have demonstrated that generating and analyzing the Pareto frontier is a viable solution concept for multi-objective security games, we plan to further extend our MOSG model in the future. One possible direction to explore is having multiple objectives for the attacker. This could model situations where the attacker explicitly considers multiple criteria when selecting a target, such economic significance, political significance, cost to attack, etc. As a result, the problem becomes even more difficult for the defender, as it is unknown what process the attacker is using to weigh the objectives in order to select a target. Such an extension may require the development of new solution concepts that rely on robust optimization techniques. Another possible direction to investigate is irrational behavior in attackers. In the current MOSG model, full rationality for the defender and all attackers is assumed. However, in practice we know that humans are not fully rational or strictly utility maximizing. Thus, if we wish to build robust model suitable for real world deployment then we must account for this irrationality. Work has been done in this area for single-objective security games [Pita et al., 2009; Yang et al., 2012], which we would seek to extend to the multi-objective case. However, one immediate consequence is that ORIGAMI-M, ORIGAMI-M-BS, and DIRECT-MIN-COV all rely on full rationality and thus would either need to be modified or replaced. These extensions will result in a higher fidelity MOSG model that is applicable to an even larger, more diverse set of domains.

## 4.10 Acknowledgement

This research was supported by the United States Department of Homeland Security through the National Center for Border Security and Immigration (NCBSI).

# Chapter 5: Multiple Defender Objectives (Exploration / Exploitation)

Traffic safety is a significant concern in cities throughout the world. Of the large number of people injured or killed in traffic accidents, a vast majority of these casualties are a direct result of reckless driving. It is for this reason, that the Singapore Police Force and their counterparts in other cities use traffic patrols to persuade drivers to comply with traffic laws through the threat of citations and fines. Such patrols must be randomized to avoid predictability and provide adequate coverage of different areas of a city. Yet, lack of randomization is a well-known problem in human patrol scheduling [Tambe, 2011] and when such randomization must also take into account speed-distance calculations, potential traffic delays, and historical data on traffic violations to ensure appropriate coverage of different areas in a city like Singapore, it presents a very difficult challenge for human schedulers.

Stackelberg security games (SSG) have become an increasingly popular paradigm for modeling security patrolling problems. In SSGs, the defender (i.e., the security agency) commits to a mixed strategy that the adversary (i.e., criminal, terrorist, or in our domain, reckless driver) is able to first *observe* and then best respond [Korzhyk et al., 2010; Basilico et al., 2009]. This mixed strategy represents a probability distribution over the possible patrol schedules. Research

on SSGs has resulted in several real-world systems deployed to protect transportation infrastructure such as airports, ports, and train stations [Tambe, 2011]. These systems have focused predominantly on counter-terrorism domains. Of the few applications that have branched out from counter-terrorism, e.g., TRUSTS [Yin et al., 2012; Jiang et al., 2013b], none have focused on traffic patrolling.

The purpose of this chapter is to introduce a new game-theoretic application, STREETS (**ST**rategic **R**andomization with **E**xploration and **E**xploitation in **T**raffic patrol **S**chedules), which we developed to assist the Singapore Ministry of Home Affairs (MHA) in scheduling randomized traffic patrols on the Singapore road network. We model this problem as a Stackelberg game with one defender (the police) and multiple adversaries (drivers). STREETS represents a novel application of Stackelberg games and required addressing several research challenges. First, road networks are complex and dynamic systems, with unpredictable delays associated with congestion, traffic signals, etc. The presence of this type of uncertainty complicates the process of planning traffic patrols. Second, the game being played at the heart of STREETS is massive in scale in terms of both the number of possible patrol strategies as well as the number of adversaries representing the thousands of drivers who use the Singapore road network. Third, the repeated nature of the traffic patrolling domain results in an abundance of data on traffic, accidents, citations, etc. However, this data is collected when the defender issues citations and thus is inherently available only for patrolled locations. Therefore, it is important to avoid confirmation bias [Nickerson, 1998] from over relying on the data, which can lead to self-reinforcing behavior and undesired consequences.

No previous work on SSGs has addressed these challenges in combination, and in fact none has addressed the challenge of avoiding confirmation bias – leading us to introduce a new concept of exploration versus exploitation in SSGs. Therefore, STREETS required us to develop a new SSG game model and an entirely new algorithm combining three key features. First, to capture the inherent stochasticity of a road network, we use a Markov Decision Process (MDP) to model the defender's patrol scheduling problem. Second, to formulate a game with an exponential number of patrol strategies and a large number of adversaries, we adopt a compact game representation which converts the defender's strategy space to a network flow through a transition graph. Additionally, we use two sampling approaches that improve efficiency by considering only a subset of either adversary types or game states when solving the game. Third, while we exploit all available data to improve patrol effectiveness, to prevent overfitting this data, we introduce an entropy-based approach. The idea being that the defender should patrol all areas of the road network with at least some probability to avoid confirmation bias and to give the perception of omnipresence to drivers. This creates a tradeoff between exploitation (minimizing reckless driving by focusing on high violation areas) and exploration (maximizing omnipresence by dispersing patrols). We explicitly formulate this tradeoff as a bi-objective optimization problem. Rather than having one optimal patrol strategy, the patrolling agency can now choose from the space of optimal tradeoff strategies located on the Pareto frontier.

STREETS was developed in collaboration with the Singapore Ministry of Home Affairs. STREETS is currently being evaluated by Singapore Police Force.

## 5.1 Domain

Traffic safety is a significant concern in cities throughout the world. Of the large number of people injured or killed in traffic accidents, a vast majority of these casualties are a direct result of reckless driving. For example, Singapore experienced 7,188 injury accidents in 2012, resulting in 168 fatalities. Perhaps just as alarming is the 330,909 traffic violations recorded during that same period for a vehicle population of only 965,192 [SPF, 2013]. It is sobering statistics like these that compel the Singapore Traffic Police (TP) and their counterparts in other cities to use traffic patrols to enforce traffic laws through the threat of citations and fines.

Since the number of roads and highways is typically very large, it is not possible to have enough resources to patrol every road and highway at every time. Therefore, a major challenge for TP is to compute patrol strategies on when and where different groups have to patrol so as to reduce the number of violations and accidents.

Due to our collaboration with the Future Urban Mobility (FM) [1] center in Singapore, we are able to obtain both the traffic volumes, violations, and accidents occurring on all the major roads and highways across Singapore. By using this data, we construct models of traffic behavior on various roads and then using the techniques developed in the next section, we generate randomized patrol strategies.

## 5.2 Model

We formally model the interaction between the police and drivers as a defender-attacker Stackelberg game. This game played by the defender and the adversaries takes place on a graph which

[1] FM is part of the Singapore MIT Alliance for Research and Technology (SMART) initiative.

models a road network where vertices represent intersections and edges represent road segments. The graph features a temporal dimension, where traversing a road segment takes some (non-deterministic) amount of time. The defender has a maximum patrol duration of $h$ hours. The defender (the police) commits to a randomized patrol strategy, which is used to generate daily patrol schedules for each of the $r$ resources. A daily patrol schedule consists of a trajectory through the graph, i.e., a sequence of road segments to patrol and the times they are to be patrolled.

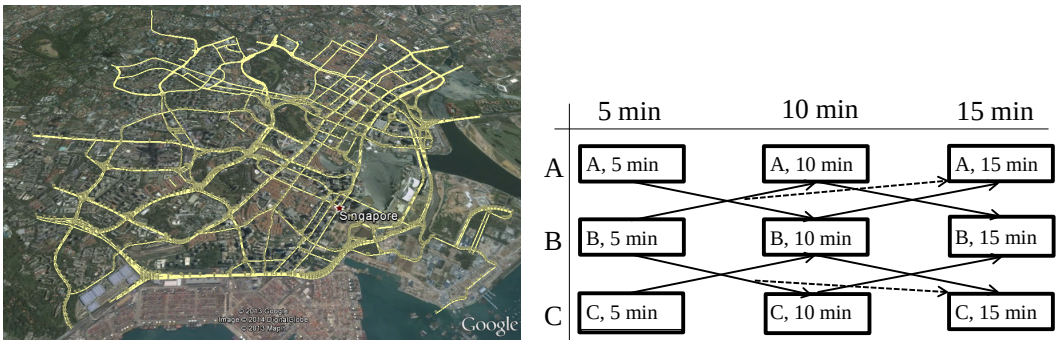
The adversaries (drivers) also follow a schedule but we assume this trajectory through the graph is fixed on a daily basis (travelling to work, school, etc.). Adversaries are able to observe the presence (or lack thereof) of police patrols over a period of time, in the process obtaining an accurate estimation of the probability of encountering the police on any given day. To construct the graph for the road network in the Singapore Central Business District (CBD), shown in Figure 5.1(a), we used data from OpenStreetMap (OSM)[2].

A normal form representation of this game, as used in the original work on Stackelberg security games [Paruchuri et al., 2008], would require us to explicitly enumerate pure strategies for the defender (patrol schedules) as well as for all of the adversaries (obey or violate decisions). This would be an extremely large number of player actions, even for small instances of our traffic patrolling domain. Therefore, we need a technique that allows us to scale up.

### 5.2.1 Achieving Scaleup

We adopt a compact representation in the form of a transition graph, which converts the game, from the defender's perspective, into a spatio-temporal flow problem. Rather than computing a probability distribution over full patrol schedules, the defender now has to compute the optimal

[2]http://www.openstreetmap.org/




(a) Singapore OpenStreetMap Graph (b) Spatio-Temporal MDP Example

Figure 5.1: Converting the Singapore road network into a spatio-temporal Markov Decision Process (MDP).

flow through the transition graph. Such a flow can be interpreted as a marginal coverage vector. These marginals can then be used to reconstruct daily patrol schedules for the defender.

This transition graph formulation is similar to the approach used in TRUSTS which modeled patrolling a train line. However, the traffic patrolling domain features a number of complexities that make our use of a transition graph within a Stackelberg game novel. One of the biggest complexities is the continuous nature of traffic patrolling. Not tied to following predetermined transportation schedules (e.g. train schedules in TRUSTS), a traffic patroller, generally speaking, can be almost anywhere within the road network at any given time. To avoid having to adopt a continuous-time model, and the associated computational overhead, we discretize time to a granularity of $m$ minutes. Therefore, a vertex is added to the transition graph for every intersection in the road network every $m$ minutes until the patrol duration of $h$ hours is reached.

#### 5.2.1.1 Defender Model

In reality, there may be unexpected delays that disrupt the defender's daily patrol schedules. In a road network, a patroller can be delayed from its schedule due to a variety of factors including

congestion or traffic signals. The defender must account for stochasticity in traffic delays when planning patrols. Therefore, we now define an MDP $\langle S, A, T, R \rangle$ to represent the defender's patrol scheduling problem:

- $S$ is a finite set of states. Each state $s \in S$ is a tuple $(l, \tau)$, where $l$ is the current location (i.e., intersection in the road network) of the defender and $\tau$ is the current time.
- $A$ is a finite set of actions. The set of actions available from a given state $s = (l, \tau)$, $A(s)$, is the set of road segments which originate from location $l$.
- $T(s, a, s')$ is the probability of ending up in the state $s'$ after performing action $a$ in state $s$.
- $R(s, a, s')$ is the immediate reward for the defender from ending up in state $s'$ after performing action $a$ in state $s$. However, our main focus is on the game-theoretic reward (i.e., expected number of violations) as a result of the defender patrolling strategy. Thus, for the remainder of this chapter, we assume, without loss of generality, that $R(s, a, s') = 0$, $\forall s, a, s'$.

Figure 5.1(b) shows a toy example of the MDP with three locations (A,B,C) and three time periods (5,10,15). The solid black arrows indicate the transitions available from each vertex. The dashed arrows represent uncertainity in the domain, e.g., anticipating going from $(B, 5) \rightarrow (A, 10)$ but being delayed and ending up in $(A, 15)$. The defender strategy is represented by the probability placed on each edge in the MDP rather than over whole patrols.

#### 5.2.1.2 Adversary Model

The set of adversaries consists of the drivers using the road network, who are assumed to always violate the law in the absence of police presence. A driver type is defined for each state-action

pair $s, a$ in the MDP and we refer to this type as $< s, a >$. This formulation represents the driver entering the transition graph (road network) at a specified vertex (intersection) and time, traversing an edge (road segment), and then exiting at the destination vertex at a later time. Thus, the trajectory of each driver type in the game is modeled as a single road segment. The reasoning being that a driver may change their behavior for different roads, choosing to violate the law on some road segments and comply with the law on others. Thus, if the decision to violate or not is made on a road-by-road basis and the decision for one road segment does not affect the decision at another, then there is no need to model driver types with trajectories with multiple road segments.

Given a fixed trajectory consisting of a single road segment, the only decision made by each individual driver type is the frequency with which they will obey the law as opposed to violate the law. This decision is influenced by the defender's patrol strategy, which we assume to be known to the drivers. If the perceived likelihood of encountering a police officer is high, then the driver will choose to obey the law more frequently [Koper, 1995]. More precisely, we define a coverage threshold $t(s, a)$ for driver type $< s, a >$ that represents the probability of encountering a patroller above which the driver will always obey the law. Starting from always violating in the absence of police patrols, we model that the probability of violating the law decreases linearly as the frequency patrols increases until the threshold $t(s, a)$ is reached and driver type $< s, a >$ no longer violates.

We use $v(s, a)$ to denote the average daily traffic volume along the road segment during the time range $[\tau, \tau + m)$. Similarly, we use $c(s, a)$ to denote the yearly violation / citation count along the road segment during the time range $[\tau, \tau + m)$. We combine the traffic volume and violation count data to define the prior associated with type $< s, a >$ as $p(s, a) = \frac{c(s,a)}{365 \times v(s,a)}$. This provides the defender with a distribution over all the adversary types in the game. Through the

Future Urban Mobility (FM) research centre, we were able to obtain traffic volume and violation count data for the Singapore CBD. We processed this data and utilized it to populate the values of $v(s, a)$, $c(s, a)$, and $p(s, a)$ which serve as input in our game.

## 5.3 Generating Randomized Patrols

Remember that the defender is trying to achieve two objectives simultaneously: (1) minimize violations; and (2) maximize omnipresence. These objectives are conflicting as they drive the defender towards different patterns of behavior. The desire to minimize violations incentivizes the defender to exploit the traffic data and patrol only in areas where violations have occurred before. Meanwhile, the desire to maximize omnipresence incentivizes exploration so that all areas of the road network are patrolled at least occasionally. Given two conflicting objectives, some tradeoff between exploration and exploitation must be made.

We borrow from work on randomized MDPs [Paruchuri et al., 2006] to formalize the tradeoff between exploration and exploitation. For discrete probability distributions, we know that entropy provides a useful measure of randomness. The MDP policy which maximizes entropy is a uniform random policy $\hat{\pi}$. Given this, one way to evaluate the level of exploration achieved is to determine the ratio of randomness compared to $\hat{\pi}$. To do this, we introduce a parameter $\beta = [0, 1]$. An MDP policy $\pi$ is said to be $\beta$-random if the following condition holds $\pi(s, a) \geq \beta\hat{\pi}(s, a)$ $\forall s, a$. Thus, fixing a $\beta$ value can be thought of as placing constraints on $\pi$, forcing it to perform a certain amount of exploration.

However, it is difficult to know *a priori* how to balance the objectives. Therefore, our approach is to generate a set of optimal compromise solutions which form the Pareto frontier using

| Variable | Definition |
|---|---|
| $c(s,a)$ | yearly violation count for type $<s,a>$ |
| $v(s,a)$ | daily traffic volume for type $<s,a>$ |
| $p(s,a)$ | prior for type $<s,a>$ set to $\frac{c(s,a)}{365 \times v(s,a)}$ |
| $t(s,a)$ | coverage threshold for type $<s,a>$ |
| $o(s,a)$ | probability of type $<s,a>$ obeying the law |
| $\hat{\pi}$ | uniform Markov policy (maximizes entropy) |
| $\beta$ | tradeoff parameter between violations / entropy |

Figure 5.2: Linear program formulation definitions for the STREETS game model.

the tradeoff parameter $\beta$, where $\beta = 0$ represents full exploitation and $\beta = 1$ represents full exploration. We present a bi-objective linear program which takes $\beta$ as input and can be solved to generate a point on the Pareto frontier. Different points on the Pareto frontier can be generated by varying the value of $\beta$. The Pareto frontier can then be presented to the end user, who selects their desired solution based any qualitative or quantitative measures they choose.

### 5.3.1 LP Formulation

We can construct a linear program (LP) to solve the MDP formulation of the defender's problem. Let $x(s,a,s')$ denote the marginal probability of the defender reaching state $s$, executing action $a$, and ending up in state $s'$. Similarly, let $w(s,a)$ be the marginal probability of the defender reaching state $s$ and performing action $a$. The probability of adversary type $<s,a>$ obeying the law which is denoted by $o(s,a)$.

We define the bi-objective linear program as follows:

$$\min_{w,\,x} \sum_{s,a} p(s,a)\left[1 - o(s,a)\right] \tag{5.1}$$

$$\text{s.t.}\; x\left(s,\, a,\, s'\right) = w\left(s,\, a\right) T\left(s,\, a,\, s'\right), \forall s,\, a,\, s' \tag{5.2}$$

$$\sum_{s',\,a'} x\left(s',\, a',\, s\right) = \sum_{a} w\left(s,\, a\right), \forall s \tag{5.3}$$

$$\sum_{a} w(s^{+},\, a) = r \tag{5.4}$$

$$\sum_{s,\,a} x(s,\, a,\, s^{-}) = r \tag{5.5}$$

$$w(s,a) \geq 0, \forall s, a \tag{5.6}$$

$$o(s,a) \leq \frac{w(s,a)}{t(s,a)}, \forall s,\, a \tag{5.7}$$

$$0 \leq o(s,a) \leq 1, \forall s,\, a \tag{5.8}$$

$$w(s,a) \geq \beta \hat{\pi}(s,a) \sum_{a'} w(s,a'), \forall s,\, a \tag{5.9}$$

Equation 5.1 is the objective function which minimizes the total expected number of violations in the system. This is a zero-sum game where each violation has the same utility and thus our goal of minimizing the total expected violations means that the minimax defender strategy is also the Strong Stackelberg Equilibrium (SSE) strategy. Constraints 5.2-5.6 are flow constraints, which combine to enforce that $\mathbf{x}$ and $\mathbf{w}$ represent feasible patrolling strategies with respect to the transition function $T$. Constraints 5.2 and 5.3 define the relationship between $\mathbf{x}$ and $\mathbf{w}$, while Constraints 5.4 and 5.5 ensure the flow out of the dummy source state $s^{+}$ as well as into the dummy sink state $s^{-}$ are equal to $r$. Constraint 5.7 computes $o(s,a)$ as the ratio between the coverage placed on the road segment $w(s,a)$ and the coverage threshold of adversary type

$< s, a >$, $t(s, a)$. For $0 \leq w(s, a) \leq t(s, a)$, adversary type $< s, a >$ will obey the law a fraction of the time, specifically $w(s, a)/t(s, a)$. Constraint 5.8 is used to ensure that $o(s, a)$ represents a valid probability, i.e., $o(s, a) \in [0, 1]$, when $w(s, a) > t(s, a)$. (This places no restrictions on $w(s, a)$, as Constraint 5.7 is an inequality constraint.)

Given $\beta$ and $\hat{\pi}$ as input, Constraint 5.9 ensures that the patrolling strategy achieves at least a fraction (i.e., $\beta$) of the randomness of the maximal entropy policy, $\hat{\pi}$, which is a uniform random policy. For example, if two actions $a_1$ and $a_2$ are available from state $s$, then $\hat{\pi}(s, a_1)$ and $\hat{\pi}(s, a_2)$ would both be 0.5. For $\beta = 0.2$, Constraint 5.9 specifies that at least 10% ($0.5 \times 0.2$) of the flow coming out of state $s$, i.e., $\sum_a w(s, a)$, must be directed to each action available from $s$, in this case $a_1$ and $a_2$. This constraint allows for a tradeoff between two objectives: (1) minimizing violations ($\beta = 0$), and (2) maximizing entropy ($\beta = 1$). The Pareto frontier can be generated by solving the LP for different values of $\beta$.

## 5.4 Additional Scaleup

For longer patrol lengths, the resulting linear program can grow quite large. To address this challenge we used constraint and state sampling [De Farias and Van Roy, 2004].

### 5.4.1 Driver Type Sampling

One approach for using constraint sampling in our problem is driver type sampling. Sampling a subset of the driver types reduces the size of the LP, as only the constraints (i.e., Constraints 5.7 and 5.8) and variables (i.e., $o(s, a)$) associated with the sampled driver types are considered. Evaluation becomes more complicated after introducing constraint sampling as we can no longer

just look at the objective value obtained by solving the sampled LP, as it only accounts for violations committed by sampled driver types. However, the defender may still implicitly influence the behavior of unsampled driver type $< s, a >$ by placing coverage on the road segment associated with $< s, a >$ in order to position themselves to interact with the sampled driver types. Thus, we use Monte Carlo simulation to sample patrol schedules from the Markov strategy computed for the sampled LP and evaluate the schedules against all driver types.

### 5.4.2 State Sampling

We can also improve efficiency by only considering a sampled subset of states obtained in a principled manner by using a coarser time granularity. For example, doubling the time granularity $m$ cuts the size of the state space in half. However, some extra steps are required when generating patrol schedules or evaluating the patrol strategy generated from the state-sampled LP on the original MDP using Monte Carlo sampling. In either case, if a state $s = (l, \tau)$ is reached which does not exist in the set of sampled states, then a look up is performed for the policy from state $s' = (l, \tau')$, where $s'$ is the state in the set of sampled states closest in time to $s$ with the same location $l$.

## 5.5 Evaluation

To evaluate STREETS, we conducted a set of simulations using actual traffic volume and violation count data from the Central Business District of Singapore provided to us by the Singapore LTA. For each simulation, we compute the Pareto frontier with an granularity of 0.2 on the $\beta$

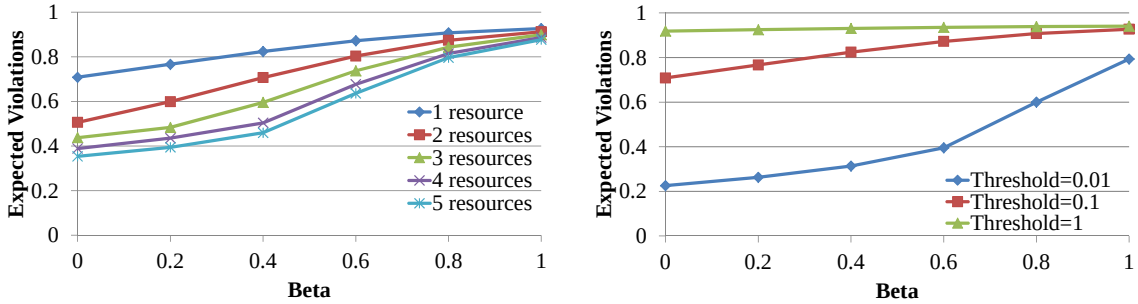
parameter which controls the tradeoff between minimizing violations and maximizing omnipresence. The Pareto frontier allows us to compare a fully game-theoretic approach with $\beta = 0$ (all exploitation) against a uniform random approach with $\beta = 1$ (all exploration), as well as everything in between. Unless otherwise specified, the default experimental setup features a patrol length of 240 minutes, a 5 minute time granularity, 1 defender resource, and a coverage threshold $t(s, a)$ of 0.1 for all drivers. All results are averaged over 30 simulations.

### 5.5.1 Analysis of Tradeoffs

#### 5.5.1.1 Defender Resources

In Figure 5.3(a), we evaluate the effect on the number of expected violations as we vary the number of defender resources $r$. The x-axis is the value of $\beta$ used when solving the LP formulation, while the y-axis is the total expected number of violations in the game, i.e., $\sum_{s,a} p(s, a)\,[1 - o(s, a)]$, achieved by the defender's (Pareto) optimal patrol strategy. As a baseline, we can use these experiments to compare a game-theoretic approach ($\beta = 0$) against a uniform random approach ($\beta = 1$).

From these results, we observe three general trends. First, increasing $r$ leads to a reduction in the expected number of traffic violations in the road network. Second, the benefit of each additional defender resource diminishes as $r$ increases. Third, as $\beta$ increases, so does the number of expected violations. This makes sense, as the defender is moving closer to a uniform random strategy and farther away from optimizing based on the violations data. It is interesting to see that $\beta = 1$ yields almost the same number of expected violations for all values of $r$ because a uniform random strategy does not allow for coordination (even implicitly) between resources.




(a) Defender Resources

(b) Driver Threshold

Figure 5.3: Effect of defender resources and driver threshold on the expected violations of STREETS.

#### 5.5.1.2 Coverage Threshold

In Figure 5.3(b), we evaluate the effect on the number of expected violations as we test three different values for driver coverage threshold, $t(s,a)$. For $t(s,a)=1$, we observe the highest level of violations as well as minimal difference between the performance of the full game-theoretic strategy ($\beta=0$) and the full uniform random strategy ($\beta=1$). This seems reasonable given that for $t(s,a)=1$ it is difficult to dissuade drivers, who are fully deterred from violating only if their road segment is patrolled with probability 1. Decreasing $t(s,a)$ to 0.1, yields a similar level of violations for $\beta=1$, but with $\beta=0$, the game-theoretic approach, which is very deliberate in how it allocates it patrols, results in a reasonable decrease in violations. Finally, at $t(s,a)=0.01$, essentially any amount of patrolling on a road segment will convince the driver types to obey. As a result, the game-theoretic strategy leads to an even greater reduction in the expected number of violations.

#### 5.5.1.3 Patrol Duration

In Figure 5.4, we evaluate the effect on runtime as we vary the patrol duration between 2 and 6 hours. Once again the x-axis is $\beta$, but now the y-axis is the runtime needed to solve the LP

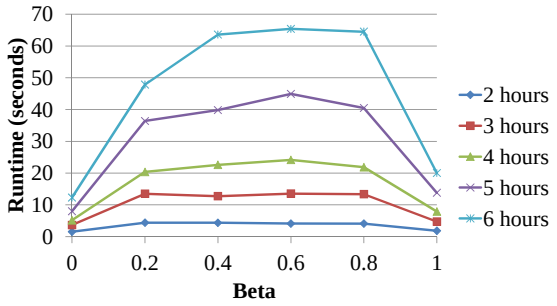



Figure 5.4: Effect of patrol duration on the runtime of STREETS.

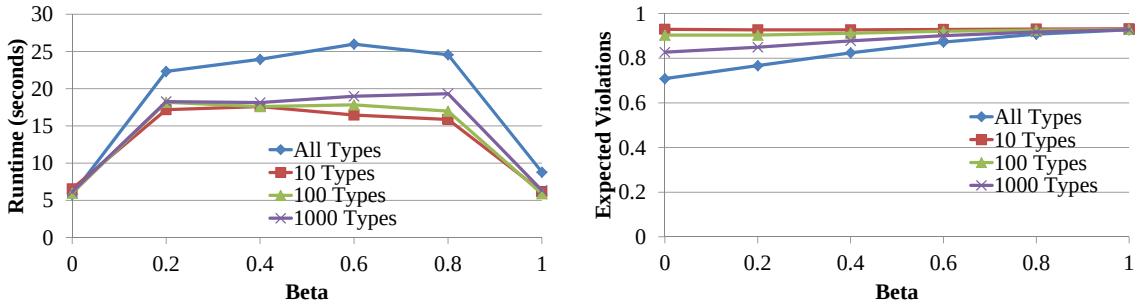
formulation. Intuitively, the results show that the runtime increases as the patrol duration is increased. Additionally, as $\beta$ is varied, we observe significantly reduced runtimes at the two extremes ($\beta = 0$ and $\beta = 1$), as in both cases, the LP is a single objective optimization problem where the other objective is ignored.

### 5.5.2 Scalability

STREETS is currently focused on generating randomized traffic patrols for the Singapore CBD. However, the eventual goal for STREETS is to scale to the entire city. Therefore, we evaluate two scaleup approaches to project how they would perform on larger problem sizes.

#### 5.5.2.1 Driver Type Sampling

In Figure 5.5(a), we evaluate the effect on runtime for different orders of magnitude of sampled driver types. The original game contains 10346 driver types. Reducing the number of driver types to 1000 via uniform random sampling results in a reasonable decrease in runtime. Further decreasing the number of sampled driver types to 100 and 10 only marginally improves the runtime. Meanwhile, in Figure 5.5(b), we evaluate the effect on solution quality as we vary the number of sampled driver types. For the smallest number of sampled types, the game-theoretic strategy




(a) Runtime

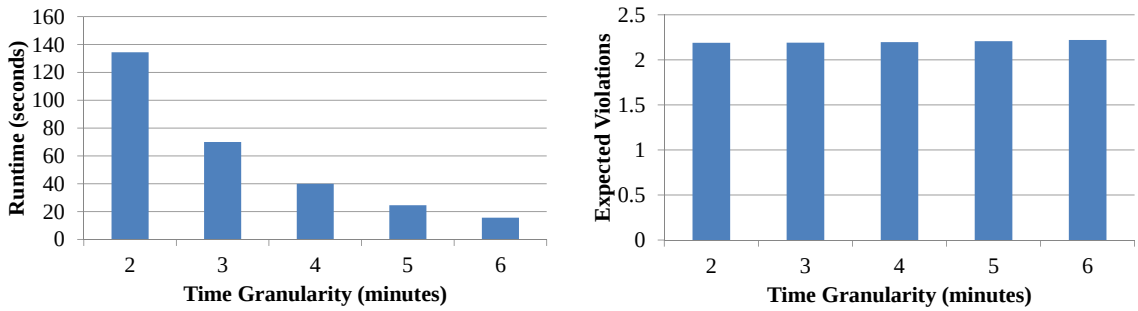
(b) Expected Violations

Figure 5.5: Effect of driver type sampling on the runtime and the expected violations of STREETS.

performs only as well as the uniform random strategy which ignores information about the driver types. Furthermore, the number of violations goes down as the number of sampled types goes up. However, the modest runtime improvements combined with the non-negligible loss in solution quality suggests there are limitations on driver type sampling as a technique for improving scalability.

#### 5.5.2.2 State Sampling

In Figure 5.6(a), we evaluate the effect on runtime as we vary the time granularity $m$ between 2 and 6 minutes. The x-axis is the time granularity and the y-axis is the runtime need to solve the LP formulation for $\beta = 0.5$. We observe an exponential decay in runtime as $m$ is increased. This results in an almost order-of-magnitude runtime decrease by going from $m = 2$ to $m = 6$. Meanwhile, in Figure 5.6(b), we evaluate the effect on solution quality as we vary $m$. The x-axis is still the time granularity $m$, but the y-axis is now the expected violations of the state-sampled strategy when evaluated on the MDP for $m = 2$. We chose to evaluate on $m = 2$ as it was the smallest value of $m$ that we could solve exactly without any sampling. Despite increasing $m$, the number of expected violations is virtually unchanged. The combination of these runtime and




(a) Runtime

(b) Expected Violations

Figure 5.6: Effect of state sampling on the runtime and the expected violations of STREETS.

solution quality results are a clear sign that state sampling via adjusting the time granularity can provide the type of scalability needed to handle patrolling over entire cities.

## 5.6 Chapter Summary

In this chapter we presented STREETS, a application which we developed to assist the Singapore MHA in scheduling randomized traffic patrols in the Singapore CBD. STREETS is currently in the process of being evaluated by the Singapore Police Force. We have already discussed how this work introduces novelties (MDP formulation, compact game representation, exploration / exploitation) over previous game-theoretic approaches for patrolling domains [Tambe, 2011]. There is a body of literature examining how to allocate traffic patrols [Adler et al., 2013; Lee et al., 1979; Koper, 1995] as well as how to influence driver behavior [Ritchey and Nicholson-Crotty, 2011]. That work has established the relation between traffic patrols and their impact on improving traffic safety, which is the basis off which STREETS is built. Much of the related research is prescriptive in nature, offering guidelines and suggestions, but stopping short of providing an implementable approach for patrolling. Our work presents a new perspective on the

problem by modeling the interaction between the police and drivers as a game. Importantly, we provide a principled approach for generating randomized schedules.

## Acknowledgements

This research is supported in part by the National Research Foundation (NRF) Singapore through the Singapore MIT Alliance for Research and Technology (SMART) and its Future Urban Mobility (FM) Interdisciplinary Research Group.

# Chapter 6: Multiple Defender Objectives (Efficacy / Efficency)

Screening people before allowing entry into a secure area is a standard practice throughout the world, e.g., screening countermeasures are used to secure border crossings, sports stadiums, government buildings, etc. Of course, a majority of people will be familiar with airport passenger screening, where each passenger must pass through physical screening consisting of a combination of countermeasures (e.g. x-ray and walk-through metal detector) before boarding their flight. Given the significant projected future growth in aviation, agencies such as the Transportation Security Administration (TSA) in the United States are developing dynamic, risk-based screening approaches which optimize the use of resources so as to maintain a high level of security while handling increased passenger volume [AAAE, 2014].

The screening domains we consider involve a screener inspecting a screenee with the goal of preventing the screenee from passing through with an attack method that could be used to cause harm in a secure area. For example, terrorists with non-metallic explosives may attempt to pass through airport screening undetected in order to attack a flight. The screener utilizes different types of screening countermeasures that have: (i) different levels of effectiveness for detecting different attack methods; and (ii) different capacities in terms of the number of screenees that can be processed within a given time window. Effective screening may require a screenee to go

through multiple screening countermeasures, but the screener may not be able to use the most effective screening countermeasure combination for every screenee. Hence, the screener may exploit available information to categorize screenees to help determine the appropriate scrutiny to apply.

To address the challenge of how to optimally utilize limited screening resources so as to minimize the risk of a successful attack by an adversary, we introduce a formal *threat screening game* (TSG) model. TSGs are played between a screener and an adversary, where the screener commits to a screening strategy, assigning a randomized combination of screening countermeasures to each screenee. The adversary is able to observe the screening strategy and best responds by posing as a screenee and selecting an attack method. The utility of the screener captures the goal of minimizing the risk of an attack across all screenees for all attack methods. The TSG model is inspired by research on security games [Tambe, 2011].

While airport passenger screening is our motivating domain, the purpose of this chapter is to introduce models, algorithms, and insights that are applicable to screening for different kinds of threats (e.g., cargo screening). Our contributions include: (1) the generalized TSG model; (2) an NP-hardness proof for computing the equilibrium of TSGs; (3) a scheme for decomposing TSGs into smaller subgames to improve scalability; (4) a column generation approach to solve TSGs which includes a novel compact multidimensional knapsack slave formulation and heuristics for faster computation; and (5) a minimax regret-based tradeoff analysis for handling uncertainty in the number of screenees and choosing a robust screening strategy. Finally, we empirically evaluate the potential benefit of using a TSG screening approach.

**Related Work** Screening games [Stiglitz and Weiss, 1994] have been used in settings with asymmetric information to model the uninformed leader screening multiple followers according

to their actions, which is different from our use of screening. Closer to our application domain, inspection games [Avenhaus et al., 1996] have looked at inspections for arms control. While the goal is similar to ours, our model has many additional features (e.g., teams) leading to a combinatorial explosion. Our model is inspired by security games [Kiekintveld et al., 2009; Jain et al., 2010b; An et al., 2011b; Korzhyk et al., 2010; Pita et al., 2011; Shieh et al., 2014] and variants such as audit games [Blocki et al., 2013, 2015], adversarial patrolling games [Basilico et al., 2009; Vorobeychik et al., 2014]; however, the properties of threat screening domains are better modeled as a TSG. These properties include (1) a number of non-adversarial screenees that affect the screening of the adversary, (2) multiple resources with varying efficiencies working in teams to screen, and (3) categorization of screenees.

Regarding screening for threats, there have been studies on how to improve screening efficiency [Ormerod and Dando, 2014] and how to screen optimally [McLay et al., 2010; Persico and Todd, 2005]. However, these do not model the game-theoretic aspect of the problem. [Wang et al., 2015] looked at a game-theoretic approach, but with a basic model that did not feature multiple screening countermeasures, screenee categories, and attack methods.

## 6.1 Motivating Domain

While threat screening games are broadly applicable to a variety of domains, in this section we focus on one concrete domain where the TSG model is particularly relevant.

In the United States, the Transportation Security Administration (TSA) is tasked with screening around 800 million air passengers annually. The TSA utilizes a number of screening countermeasures for screening passengers, e.g., X-RAY machines, walk-through metal detectors

(WTMD), advanced imaging technology (AIT) machines, explosive trace detection (ETD) units. Each passenger is required to go through some combination of these screening countermeasures before boarding their flight, with the goal of minimizing the threat of a terrorist passing through screening and attacking a flight (via on-body or carry-on non-metallic explosives, etc.).

The TSA's current DARMS (Dynamic Aviation Risk Management System) initiative aims to enhance aviation security [AAAE, 2014]. In our joint work with the TSA, we focus solely on the passenger screening component of DARMS. Whereas the TSA previously screened all passengers equally, recently they have begun to perform risk-based screening through programs such as TSA Pre✓® in which passengers can choose to submit to background checks in order to receive expedited screening. The idea being that fewer resources should be dedicated to screening lower risk passengers and more resources dedicated to screening higher risk passengers, improving overall screening efficiency and efficacy. In DARMS, the TSA assigns passengers a risk level based on available information such as flight history, frequent flyer membership, TSA Pre✓® status, etc. The TSA also assigns a value to each flight that measures its attractiveness as a target for terrorists based on gathered intelligence.

The innovation in DARMS is that the screening for each passenger is conditioned on both the passenger's *risk level* and *flight*. Our goal is exploit this flexibility by using the TSG model to compute the optimal screening strategy, given the available screening resources.

## 6.2 Game Model

A *threat screening game* (TSG) is a Stackelberg game played between the screener (leader) and an adversary (follower). The adversary attempts to conceal their attack method by posing as one of the other benign screeenees.

**TSG Model** A TSG specification includes a set of screenees $S$ and a set of attack methods (AMs) $M$ indexed by $\{1, \ldots, |M|\}$. Then, all possible AMs for every screenee is represented as $\mathcal{A} = \{\mu_{s_1,1}, \ldots, \mu_{s_1,|M|}, \mu_{s_2,1}, \ldots, \mu_{s_{|S|},|M|}\}$. The screener's action is to allocate screening resources to $\mathcal{A}$. However, the screener can allocate resources only at the level of granularity of each screenee. Thus, any resource may be allocated to $\{\mu_{s_i,1}, \ldots, \mu_{s_i,|M|}\}$ for each $s_i \in S$. The adversary's action is to pose as a screenee $s$ and use an AM indexed by $m$, i.e, choose one of $\mu_{s,m} \in \mathcal{A}$.

The goal of the screener is to detect the adversary. The utility for the screener is given in terms of when the screener successfully detects the adversary $U^d_\sigma(\mu_{s,m})$ or is unable to detect the adversary $U^u_\sigma(\mu_{s,m})$. As our motivating domain is zero-sum, we assume a zero-sum game so the adversary's utility is negation of these, i.e., $U^d_a = -U^d_\sigma$ and $U^u_a = -U^u_\sigma$.

The complete specification of a TSG includes many characteristics, which we list below.

- *Resource types*: The set of types of resources is $\Lambda$, and resource of type $\lambda \in \Lambda$ can be used $C_\lambda$ times.
- *Teams*: A screenee must be screened by a single valid team, where a team is formed by single usage of each resource type in given subset of $\Lambda$. Thus, we can uniquely assign a type $\psi$ to each team, where $\psi \in \Lambda$. The set of all valid team types is given apriori, and denoted by $\Psi$.

Given capacity for each resource type $C_\lambda$, we have the following capacity constraints for the number of usages $T_\psi$ of team type $\psi$:

$$\forall \lambda.\ \sum_{\psi \in \Psi} I_{\lambda \in \psi} T_\psi \leq C_\lambda$$

where $I_{\lambda \in \psi}$ is the indicator for resource type $\lambda$ belonging to the team type $\psi$. Thus, allocations are stated in terms of allocating teams to screenees.

- *Screenee categories*: Screenees in same category are indistinguishable w.r.t. utility for both players. More formally, given the set of categories $\Xi$, if $s, s' \in \xi$ for some $\xi \in \Xi$ then $U^d_\sigma(\mu_{s,m}) = U^d_\sigma(\mu_{s',m})$ for all $m$ (same for $U^u_\sigma$). Thus, we write $U^d_\sigma(\mu_{\xi,m})$ instead of $U^d_\sigma(\mu_{s,m})$ (same for $U^u_\sigma$). The number of screenees in each category $\xi$ is $N_\xi$. Then, all screenees in a category are screened equally in expectation (unequal screening makes the adversary pose as the least screened screenee, thereby wasting resources).

  *Equivalence class of actions*: Due to utility equivalence in any category, the adversary's choice of screenee $s$ reduces to choice of screenee category $\xi$ ($\mu_{\xi,m}$). Similarly, the screener's action specifies the number of team usages (of different types) allocated to each screenee category ($\mathsf{n}_{\psi,\xi}$).

- *Effectiveness*: Teams vary in the level of protection they provide against AMs. Formally, for a team of type $\psi$ screening $s \in \xi$ there is a vector $\vec{E}_{\psi,\xi}$ of size $|M|$ such that the $m^{th}$ component $\vec{E}_{\psi,\xi}(m)$ is the level of protection (probability of perfect detection) against AM indexed by $m$. Observe that given a team for the same AM effectiveness can vary by screenee category.

- *Adversary restrictions*: $\Theta$ is a partition of the screenee categories. An adversary with *restriction* $\theta \in \Theta$ can only pose as a screenee in categories within $\theta$, i.e. his action space is

restricted[1][2]. The adversary knows his own restriction, but the defender does not. The defender knows a prior distribution $W_\theta$ over the adversary restriction.

*Example:* Consider a scaled-down airport screening example, focusing on one hour of screening. The defender has two resource types: Metal detector (D) and Explosive trace detector (E), i.e., $\Lambda = \{\text{D,E}\}$. The capacities $C_D$ is 100 and $C_E$ is 10. In particular, the D machine can screen 100 people in one hour, and the E machine can screen 10 people in one hour. Three team types are possible: $\Psi = \{\{\text{D, E}\}, \{\text{D}\}, \{\text{D}\}\}$. Screenees are partitioned into three categories: $\Xi = \{\langle\text{f1,r1}\rangle, \langle\text{f1,r2}\rangle, \langle\text{f2,r2}\rangle\}$, where f1,f2 are two flights and r1,r2 are two risk levels. Note that no screenee in risk level r1 can buy a ticket for f2. The number of people arriving are given by $N_{\text{f1, r1}} = 20, N_{\text{f1, r2}} = 20, N_{\text{f2, r2}} = 30$. There are two attack methods: $\mu = \{\text{g, e}\}$ that denote guns and explosives. Team D,E has $\vec{E}_{D,E,\xi}(i) = 1$ for any category $\xi$ and AM $i$. Team $E$ has $\vec{E}_{E,\xi}(\text{g}) = 0.1$, $\vec{E}_{E,\xi}(\text{e}) = 1$ for any category $\xi$. Team D is tuned to work more efficiently for risk level $r1$, thus $\vec{E}_{D,\xi}(\text{g}) = 1$, $\vec{E}_{D,\xi}(\text{e}) = 0.4$ for $\xi =$f1,r1, and $\vec{E}_{D,\xi}(\text{g}) = 0.9$, $\vec{E}_{D,\xi}(\text{e}) = 0.1$ otherwise. Adversary has two restrictions $\theta_1, \theta_2$: $\theta_1$ is $\langle$f1,r1$\rangle$ and $\theta_2$ is $\langle$f2,r2$\rangle, \langle$f1,r2$\rangle$, i.e., $\Theta$ partitions $\Xi$ by risk levels or in other words, the adversary is given his risk level and he can only choose flights and AMs. The defender knows the probability $W_{\theta_1} = 0.2$ and $W_{\theta_2} = 0.8$. The utility of screener is stated $U^d_\sigma(\mu_{\xi,i}) = 0$ for any $\xi, i$, and $U^u_\sigma(\mu_{\xi,g}) = -2$, $U^u_\sigma(\mu_{\xi,e}) = -5$ for $\xi =$ f2, r2, and $-1$ otherwise.

We present two additional characteristics that allow for a compact representation of TSGs.

*Default team type:* We call a resource type sufficient if $C_\lambda \geq \sum_\xi N_\xi$, i.e., this resource can be used in every screening. We *do not include* any sufficient resource type in the set of resources

[1]This cannot be modeled using adversary types, as simulating restriction on actions would need to set utility for disallowed actions as $-\infty$. This does not make sense in a zero sum game.

[2]An easy extension is to also allow restriction on AMs. For sake of exposition we focus on restriction in choice of screenee category.

types $\Lambda$. In addition, we also posit a *default team type* indexed by letter $\delta$, where the maximum possible number of default teams is more than the number of screenees. In other words, the default team type is formed from either the sufficient resource types or denotes no screening. In generally, the default team provides basic (light) screening. Again, we *do not include* the default team in the set of team types $\Psi$.

*Sliced Game:* The screening problem also has a temporal dimension. Screenees arrive not all at once, but, over time. We model this by slicing the game into into time slots. The number of time slots in a day is $\eta$. Our description above is just for one time slot. To accommodate multiple time slots, we use superscript $\tau$ for relevant variables to indicate the time slot we are referring to, and we skip $\tau$ when it is clear that we are referring to one time slot only. We abstract away from the continuous nature of arrival of screenees, assuming a steady flow that allows parallel use of resources.

| About | Notations |
|---|---|
| Resource | type: $\lambda \in \Lambda$; capacity: $C_\lambda$ |
| Team | type: $\psi \in \Psi \subseteq 2^\Lambda$; default type $\delta \notin \Psi$ |
| Screenee | $\Xi$ partition of screenees; category: $\xi \in \Xi$ |
| Adv. action | Choose $\xi, m$ represented as $\mu_{\xi,m}$ |
| Adv. restr. | $\Theta$ partition of $\Xi$; $\theta \in \Theta$ |
| Team alloc | $\mathsf{n}_{\psi,\xi}$: allocation of team type $\psi$ to $\xi$ |
| AM | indexed by $m$, also referred to as $m$ |
| Efficiency | $\vec{E}_{\psi,\xi}$: detection prob. of each $m$ by $\psi$ in $\xi$ |

**Pure action space** A pure allocation $A$ can be represented by a non-negative integer valued matrix $M^{\tau,A}$ of size $|\Psi| \times |\Xi|$. The $\psi, \xi$ entry $\mathsf{n}^{\tau,A}_{\psi,\xi}$ is the number of usages of team type $\psi$ allocated

to screenees of category $\xi$ for time slot $\tau$. We have $\sum_\psi \mathsf{n}^{\tau,A}_{\psi,\xi} \leq N^\tau_\xi$ since every screenee is screened at most by one team and any leftover screenees $\mathsf{n}^{\tau,A}_{\delta,\xi} = N^\tau_\xi - \sum_\psi \mathsf{n}^{\tau,A}_{\psi,\xi}$ are screened by the default team. Thus, every screenee is screened by a team. The number of usages of team type $\psi$ for allocation $A$, given by $\vec{T}^{\tau,A}_\psi = \sum_\xi \mathsf{n}^{\tau,A}_{\psi,\xi}$, must satisfy the capacity constraints. Thus, the set of all valid allocations $P^\tau$ ($A \in P^\tau$) for time slot $\tau$ is given by matrices formed from the different *integral* values $\mathsf{n}^\tau_{\psi,\xi}$ that satisfy the inequalities

$$\sum_\psi I_{\lambda\in\psi} \sum_\xi \mathsf{n}^\tau_{\psi,\xi} \leq C^\tau_\lambda \text{ and } \sum_\psi \mathsf{n}^\tau_{\psi,\xi} \leq N^\tau_\xi \tag{6.1}$$

The adversary chooses a screenee category (within restriction $\theta$) and chooses an AM, which we state as $\mu^\tau_{\xi,m}$ and $\xi \notin \theta$ implies the adversary cannot choose $\mu^\tau_{\xi,m}$ for any AM $m$.

*Example Continued:* Continuing the airport example, the default team is D. A allocation allocates the other two team types to three screenee categories. A possible allocation is inspecting 10 screenees in category $\langle$f1,r1$\rangle$ with D,E and the remaining 10, 20, 30 screenees in categories $\langle$f1,r1$\rangle$, $\langle$f1,r2$\rangle$, $\langle$f2,r2$\rangle$ respectively with the default team D. Since all screenees in a category are screened equally, the 10 screenees in category $\langle$f1,r1$\rangle$ to be screened by D,E are chosen at random from the 20 overall screenees in $\langle$f1,r1$\rangle$.

**Mixed Strategy** Given probabilities $p_1, \ldots p_{|P^\tau|}$ ($\sum p_i \leq 1$) over all valid pure allocations $A_1, \ldots, A_{|P^\tau|}$ ($A_i \in P^\tau$), we get the matrix $M^\tau = \sum_i p_i M^\tau_{A_i}$. The elements $n^\tau_{\psi,\xi}$ of $M^\tau$ stand for the expected number of teams of type $\psi$ allocated to screenees in category $\xi$; $n^\tau_{\psi,\xi}$ is a real number. The numbers $n^\tau_{\psi,\xi}$ lie in the convex hull of the integral points given by equation 6.1. We denote that as $n^\tau_{\psi,\xi} \in conv(P^\tau)$.

**Utilities** Given the mixed strategy above, and the fact that all screenees in category $\xi$ are screened equally, we can interpret $n^{\tau}_{\psi,\xi}/N_{\xi}$ (and $n^{\tau}_{\delta,\xi}/N_{\xi}$) as the probability that screenee in category $\xi$ will be screened by team of type $\psi$ (and $\delta$). Then, the level of protection against AMs for adversary in category $\xi$ is given by the vector $\vec{x}^{\tau}_{\xi} = \sum_{\psi} n^{\tau}_{\psi,\xi}\vec{E}_{\psi,\xi}/N^{\tau}_{\xi} + (N^{\tau}_{\xi} - \sum_{\psi} n^{\tau}_{\psi,\xi})\vec{E}_{\delta,\xi}/N_{\xi}$ with each component $\vec{x}^{\tau}_{\xi}(m)$ being the level of protection against AM $m$. Given the adversary's choice $\mu^{\tau}_{\xi,m}$, the defender's utility $U_{\sigma}(\vec{x}^{1}_{\xi}, \ldots, \vec{x}^{\eta}_{\xi}, \mu^{\tau}_{\xi,m})$ is

$$\vec{x}^{\tau}_{\xi}(\mu^{\tau}_{\xi,m})U^{d}_{\sigma}(\mu^{\tau}_{\xi,m}) + (1 - \vec{x}^{\tau}_{\xi}(\mu^{\tau}_{\xi,m}))U^{u}_{\sigma}(\mu^{\tau}_{\xi,m})$$

Analogously, for the adversary $U_{a}(\vec{x}^{1}_{\xi}, \ldots, \vec{x}^{\eta}_{\xi}, \mu^{\tau}_{\xi,m})$ is

$$\vec{x}^{\tau}_{\xi}(\mu^{\tau}_{\xi,m})U^{d}_{a}(\mu^{\tau}_{\xi,m}) + (1 - \vec{x}^{\tau}_{\xi}(\mu^{\tau}_{\xi,m}))U^{u}_{a}(\mu^{\tau}_{\xi,m})$$

**SSE computation** The Strong Stackelberg equilibrium (SSE) computation is given by the following optimization

$$\begin{aligned}
\max_{d_{\theta}, n_{\psi,\xi}} \quad & \textstyle\sum_{\theta} W_{\theta} d_{\theta} \\
\forall \theta, \tau, m, \xi \in \theta. \quad & d_{\theta} \leq U_{\sigma}(\vec{x}^{1}_{\xi}, \ldots, \vec{x}^{\eta}_{\xi}, \mu^{\tau}_{\xi,m}), \\
\forall \tau, \xi. \quad & \vec{x}^{\tau}_{\xi} = \frac{\sum_{\psi} n^{\tau}_{\psi,\xi}\vec{E}_{\psi,\xi} + (N^{\tau}_{\xi} - \sum_{\psi} n^{\tau}_{\psi,\xi})\vec{E}_{\delta,\xi}}{N_{\xi}}, \\
\forall \tau. \quad & n^{\tau}_{\psi,\xi} \in conv(P^{\tau})
\end{aligned}$$

**Algorithm 8:** MODULAR_SOLVER

1 For each $\tau$, form the sub-game $G^\tau$.
2 $\Delta P^\tau \leftarrow ComputeEquilibrium(G^\tau)$ for each $\tau$
3 **return** $\Delta P^1, \ldots, \Delta P^\eta$

## 6.3 Algorithmic Approach

### 6.3.1 Separation

It may seem natural that, since the TSG is sliced into $\eta$ time slots, the equilibrium can be computed by solving the sub-game $G^\tau$ for each time slot (assuming adversary conducts one attack in each time slot) separately and then combining the mixed strategy for each sub-game to get a mixed strategy for the overall game, as shown in Algorithm 8. However, rather surprisingly, this technique does not work in general non zero-sum TSGs.

**Counterexample:** Consider a game with four screenee $s_1, s_2, s_3, s_4$ and one AM 1 that is sliced into two parts $G^1, G^2$, with each sub-game having two screenees and just one resource. The resource can detect the AM perfectly.

| $G^1$ | $\mu_{s_1,1}$ | | $\mu_{s_2,1}$ | |
|---|---|---|---|---|
| | d | u | d | u |
| Def. | 0 | -20 | 0 | -22 |
| Adv. | 3 | 5 | 3 | 5 |

| $G^2$ | $\mu_{s_3,1}$ | | $\mu_{s_4,1}$ | |
|---|---|---|---|---|
| | d | u | d | u |
| Def. | 0 | -2 | 0 | -3 |
| Adv. | 2 | 5 | 2 | 5 |

The optimal strategy for screener in $G^1$ is to allocate the resource to $\mu_{s_1,1}$ with $0.5$ prob. and $\mu_{s_2,1}$ with $0.5$. The adversary chooses $\mu_{s_1,1}$ and get payoff $4$, and screener gets payoff $-10$. Similarly, the optimal strategy for screener in $G^2$ is also $0.5$ and $0.5$. adversary attacker chooses $\mu_{s_3,1}$ and get payoff $3.5$, and screener gets payoff $-1$.

Now, observe that simply combining the equilibrium of the two sub-game makes the adversary choose $\mu_{s_1,1}$ gaining $4$, and screener gets $-10$. However, if in $G^2$ the strategy is changed

to 1 on $\mu_{s_4,1}$, then the adversary chooses $\mu_{s_3,1}$ gaining 5 and screener gets $-2$. Thus, simply combining the equilibrium of the two sub-games is not the most optimal solution.

Next, we present conditions that allow use of Algorithm 8:

**Theorem 1.** *Algorithm 8 produces a Stackelberg equilibrium when all sliced sub-games satisfy the condition: $U_\sigma = -cU_a$, where $c > 0$ is any real number. $U_\sigma$ and $U_a$ are the screener's and adversary's utility (detected or undetected) respectively.*

*Proof Sketch.* The proof works by first proving that the best adversary's best responses from each sub-game from Algorithm 8 is indeed the overall best response. Then, for contradiction, we show that if there is a distribution $\Delta P_0$ that provides higher utility to the screener overall, then there is a sub-game $i$ such that the marginal of $\Delta P_0$ in this sub-game $\Delta P_0^i$ provides higher utility that the SSE for the sub-game. □

Our game is zero-sum, and hence satisfies the condition above. Thus, we drop the time superscript $\tau$ and focus on a sub-game and solve the following optimization $O$

$$\begin{aligned}
\max_{d_\theta, n_{\psi,\xi}} \quad & \textstyle\sum_\theta W_\theta d_\theta \\
\forall \theta, m, \xi \in \theta. \quad & d_\theta \leq U_\sigma(\vec{x}_\xi, \mu_{\xi,m}), \\
\forall \xi. \quad & \vec{x}_\xi = \frac{\sum_\psi n_{\psi,\xi} \vec{E}_{\psi,\xi} + (N_\xi - \sum_\psi n_{\psi,\xi}) \vec{E}_{\delta,\xi}}{N_\xi}, \\
& n_{\psi,\xi} \in conv(P)
\end{aligned}$$

**Simplified notation** To handle the large number of variables, we introduce short hand for them. Let $\frac{(U_\sigma^d(\mu_{\xi,m}) - U_\sigma^u(\mu_{\xi,m}))\vec{E}_{\psi,\xi}(m)}{N_\xi}$ be $I_{\psi,\xi,m}$. Also, let $I_{\psi-\delta,\xi,m} = I_{\psi,\xi,m} - I_{\delta,\xi,m}$ and $N_\xi I_{\delta,\xi,m} + U_\sigma^u(\mu_{\xi,m}) = -J_{m,\xi}$. Then, the inequality with $d_\theta$ can be written as $d_\theta \leq \sum_\psi n_{\psi,\xi} I_{\psi-\delta,\xi,m} - J_{m,\xi}$ . Let $X = [X^1, \ldots, X^{|P|}]$ denote the matrix of size $|\Psi||\Xi| \times |P|$

with each $X^p$ denoting a pure strategy. $X^p$ is formed by arranging the columns of the pure allocation matrix one after another. $X^p_{\psi,\xi}$ denotes the allocation of team type $\psi$ to screenee category $\xi$ in $X^p$. The constraint that $n$ lies in the $conv(P)$ is given by replacing $n$ by $Xq$ where $q$ is a vector of probabilities over $P$.

Thus, the optimization problem $O$ is given by

$$\begin{aligned} \max_{d,q} \quad & \textstyle\sum_\theta W_\theta d_\theta \\ \forall \theta, \xi \in \theta, m. \quad & -d_\theta + \textstyle\sum_\psi I_{\psi-\delta,\xi,m} \sum_{p\in P} X^p_{\psi,\xi} q_p \geq J_{m,\xi}, \\ & \textstyle\sum_{p\in P} q_p = 1, \quad q \geq 0 \end{aligned}$$

### 6.3.2 Relaxation and Projection

**Theorem 2.** *Problem $O$ is NP-Hard to compute.*

*Proof Sketch.* We do a reduction from independent set problem. The core of hardness in $O$ is due to team formation. Thus, we work with the special case with one screenee category $1$ and one AM $1$. In this case we show that solving the resultant LP is equivalent to solving an integer LP with constraints given by equation 6.1. Given a graph, we construct an integer LP instance of our problem by choosing a team type $\psi$ for each vertex, resource types for each vertex and each edge (with all capacities 1). It is shown that the integer LP solution is the size of the max. independent set. □

We use a slightly modified column generation approach with heuristics that provide fast computation for problem $O$, inspired by [Yang et al., 2013]. The outline is provided in Algorithm 9. We start by solving a relaxed version $O_{relax}$ of $O$ obtained by relaxing equation 6.1. The relaxed

**Algorithm 9:** SCREEN

```
P ← φ; z ← 1; X ← φ
do
    n̂, d̂ ← Solve(O_relax); P ← P ∪ X
    z, X, q, cut ← l1PROJECTION(n̂, P)
    O_relax ← Add the constraint cut to O_relax
while z ≠ 0
return X, q
```

problem solution $\hat{n}$ may not be a valid mixed strategy, as shown in a counterexample in Appendix. $O_{relax}$ is shown below

$$\begin{aligned} \max_{d,n} \quad & \textstyle\sum_\theta W_\theta d_\theta \\ \forall \theta, \xi \in \theta, m. \quad & -d_\theta + \textstyle\sum_\psi I_{\psi-\delta,\xi,m} n_{\psi,\xi} \geq J_{m,\xi}, \\ \forall \lambda. \quad & \textstyle\sum_\psi I_{\lambda \in \psi} \sum_\xi n_{\psi,\xi} \leq C_\lambda, \\ \forall \xi. \quad & \textstyle\sum_\psi n_{\psi,\xi} \leq N_\xi, \quad \forall \psi, \xi.\, n_{\psi,\xi} \geq 0 \end{aligned}$$

The l1PROJECTION algorithm finds the l1 distance $z$ of $\hat{n}$ to the mixed strategy space ($conv(P)$) in the original problem $O$. In the process, it also finds the l1 projection of $\hat{n}$ onto $conv(P)$, expressed as a convex combination of pure strategies in set $X$ with the coefficients given by set $q$. In addition, it also finds the deep cut separating $\hat{n}$ and $conv(P)$ via the dual (easily inferable using minimum norm duality theorem [Luenberger, 1997] from functional analysis). Clearly, if $z$ is zero then $\hat{n}, \hat{d}$ is a valid solution, and we obtain our desired result in $X, q$. Otherwise the cut is added to the problem $O_{relax}$, and the loop repeats. Note that we reuse the generated

pure strategies $X$ from one run of l1PROJECTION in the next run. In l1PROJECTION, the optimization is

$$\begin{aligned}
\min_{z,q} \quad & \textstyle\sum_{\psi,\xi} z_{\psi,\xi} \\
\forall \psi, \xi. \quad & z_{\psi,\xi} + \textstyle\sum_{p \in P} X^p_{\psi,\xi} q_p \geq \hat{n}_{\psi,\xi}, \\
\forall \psi, \xi. \quad & z_{\psi,\xi} - \textstyle\sum_{p \in P} X^p_{\psi,\xi} q_p \geq -\hat{n}_{\psi,\xi}, \\
& \textstyle\sum_{p \in P} q_p = 1, \quad q \geq 0, \quad z \geq 0
\end{aligned}$$

The first two set of constraints specify $-z \leq ||\hat{n} - n||_1 \leq z$. We do column generation (master-slave decomposition) to solve the above master problem (called $l1\text{-}primal$). The dual variables for the primal are $v_{\psi,\xi}$, $v'_{\psi,\xi}$ for the two set of inequalities and $o$ for the equality. Let $y = v - v'$. In the column generation iteration, given a $y, o$, we find the next pure strategy to add by solving the following compact integer linear program formulation of the slave (called $Sep\text{-}Oracle$):

$$\begin{aligned}
\max_{\mathsf{n}} \quad & \textstyle\sum_{\psi,\xi} \mathsf{n}_{\psi,\xi} y_{\psi,\xi} + o \\
\forall \lambda. \quad & \textstyle\sum_{\psi} I_{\lambda \in \psi} \sum_{\xi} \mathsf{n}_{\psi,\xi} \leq C_\lambda, \\
\forall \xi. \quad & \textstyle\sum_{\psi} \mathsf{n}_{\psi,\xi} \leq N_\xi, \quad \forall \psi, \xi.\ \mathsf{n}_{\psi,\xi} \in \{0, 1, \ldots\}
\end{aligned}$$

This novel slave formulation and the heuristics used to solve the slave sets us apart from [Yang et al., 2013]. Analogous to security games, we call the above problem "defender oracle" for varying $y$. The defender oracle is an instance of unbounded multidimensional knapsack.

**Lemma 1.** *The defender best response oracle problem is hard to approximate to any constant factor, unless P=NP.*

*Proof Sketch.* The same reduction used in Theorem 2 can be used as a PTAS (approximation preserving) reduction here, with the fact that independent set is hard to approximate. □

**Algorithm 10:** l1PROJECTION($\hat{n}, \hat{d}, P$)

```
do
    z, q ← Solve(l1-primal, P)
    y, o ← ReadDualValues(z, q)
    P' ← Solve-Sep-Oracle(y, o); P ← P ∪ P'
while P' ≠ φ
Get X, q from positive values in q; get cut from the y, o
return z, X, q, cut
```

The pure strategy maximizing the slave objective is added back to the master, if the objective is positive. However, it is sufficient to find a pure strategy that makes the objective positive in each iteration. Thus, we try the following alternatives:

- Best Response: Solve the defender oracle exactly.
- Better Response: Relax the defender oracle to a LP and obtain a solution $\tilde{n}$. If $\tilde{n}$ is integral then it is the same as the best response, so we use $\tilde{n}$ and stop. Otherwise, generate a fixed number of pure strategies by randomly increasing components of $\lfloor \tilde{n} \rfloor$ while still being feasible, checking if any yields a positive objective. Try the better response heuristic first, if it fails solve the defender oracle exactly.
- Slave Iteration Cutoffs: Stop after a given threshold number of iterations (for both better or best reponse).

Also, given dual solution $y^*, o^*$, the hyperplane $\sum_{\psi,\xi} n_{\psi,\xi} y^*_{\psi,\xi} + o^* = 0$ is a deep cut in $O_{relax}$.

### 6.3.3 Addressing Uncertainty

Up to this point, we have assumed that the screenee distribution $N_\xi$ is known exactly. However, there may be uncertainty in real-world screening domains. We consider uncertainty to be limited

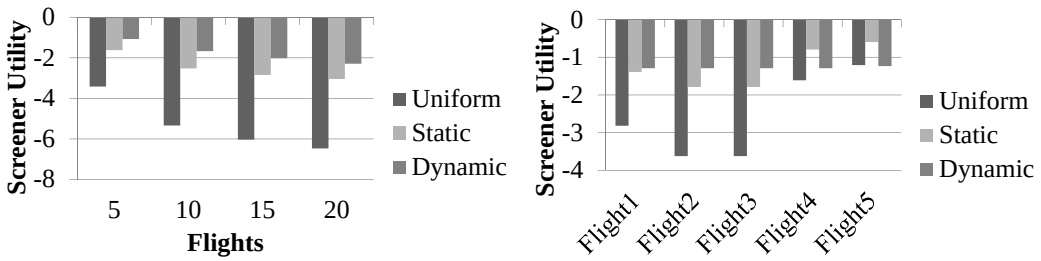
to $K$ distributions $N^k$ for $k \in \{1, \ldots, K\}$, where $N^k_\xi$ specifies the number of screenees in category $\xi$. We assume a probability $p_k$ for each distribution being realized.

Given $p_k$, one approach for handling uncertainty could be to use the *expected* number of screenees in each category when computing the screening strategy. However, this approach is undesirable as underestimating and overestimating $N_\xi$ yield different challenges for the screener and cannot be equivocated. Underestimating $N_\xi$ can lead to an overflow of screenees being assigned to a particular resource team type, causing the workload of screening resources to exceed capacity. Overestimating $N_\xi$ can lead to underflow where screening resource capacity that could have been used elsewhere remains unused leading to regret in screener utility.

Therefore, we propose an alternative approach for handling uncertainty over $N_\xi$. We compute the optimal screening strategy for each $N^k$ and then evaluate that screening strategy on every other distribution $N^{k'}$ to compute the weighted average percentage of overflow screenees and the weighted average screener utility regret. Importantly, the evaluation criteria are kept separate, resulting in a multi-objective space with a compromise solution for each screening strategy that trades off between overflow passengers and screener utility regret. We can then compute the Pareto frontier, enabling the screener to choose their desired Pareto optimal screening strategy.

## 6.4 Evaluation

We evaluate our threat screening game model as well as the associated algorithms and heuristics using experiments inspired by the TSA DARMS passenger screening domain. The game payoffs are zero-sum and randomly generated with $U^u_a$ uniformly distributed in [1,10] and $U^u_\sigma = -U^u_a$. The remaining game payoffs $U^d_\sigma$ and $U^d_a$ are fixed to 0. The default settings for each experiment




(a) Screener Utility (b) Example Game Instance

Figure 6.1: Solution quality comparison of three screening approaches and an example game instance highlighting the benefit of dynamic screening.

are (unless otherwise noted): 4 screenee risk levels, 5 screening resource types, 8 screening team types, and 1 time window. All results are averaged over 30 randomly generated game instances.

### 6.4.1 Screening Approach

TSGs optimize the allocation of screening resources by exploiting screenee categories defined as ⟨flight, risk level⟩ in this domain. To show the benefit of using this screenee categorization, we compare the resulting *dynamic* screening strategy against two baseline approaches: (1) a *uniform* approach which solves the TSG with an additional constraint that all screenees must be screened using the same screening strategy, and (2) a *static* approach which solves the TSG with an additional constraint that all screenees with the same risk level must be screened using the same screening strategy across all flights. Figure 6.1(a) shows the solution quality comparison of the three approaches, where the x-axis is the number of flights and the y-axis is screener utility. As expected, the *dynamic* approach is able to achieve higher screener utility than both the *uniform* and *static* approaches for all numbers of flights. Figure 6.1(b) shows a comparison for a specific game instance with 5 flights and provides intuition as to why the *dynamic* approach performs so well. The inability to strategically adjust screening flight by flight results in both the *uniform*

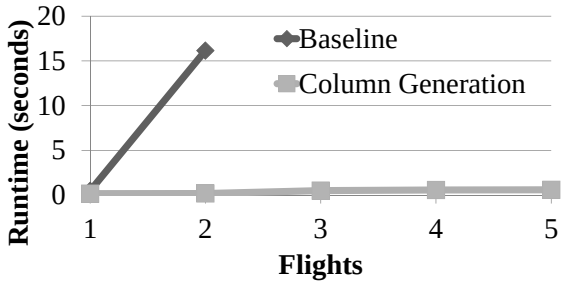



Figure 6.2: Runtime comparison of the baseline approach and column generation approach for solving threat screening games.

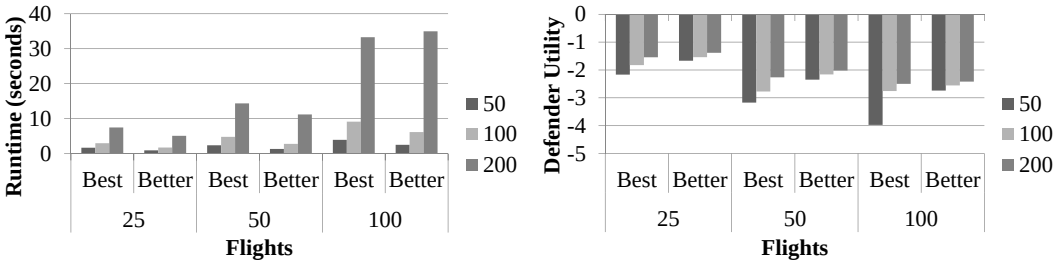
and *static* approaches protecting some flights adequately (Flight 5), while leaving other flights vulnerable (Flight 3), leading to lower screener utility.

### 6.4.2 Algorithmic Approach

The baseline algorithm for solving TSGs involves enumerating every pure strategy of the screener. To illustrate the importance of column generation, we consider a small game with 2 screening resource types, 2 screening team types, and 12 screenees per flight (3 at each risk level). Figure 6.2 shows a runtime comparison of the two algorithms for varying numbers of flights. We observe that the baseline algorithm is unable to scale beyond 2 flights. For 3 flights, the screener has 16,777,216 pure strategies and the baseline algorithm runs out of memory, while column generation easily scales up.

### 6.4.3 Heuristics

Figure 6.3 shows a runtime and solution quality comparison of best response and better response with varying slave iteration cutoffs. Runtime results are presented in Figure 6.3(a), with the x-axis denoting the number of flights as well as the type of slave response (i.e., best or better), and the y-axis indicating the runtime needed to reach the different cutoffs. As expected, the runtimes




(a) Runtime (b) Screener Utility

Figure 6.3: Runtime and solution quality comparison of the best response and better response heuristics with varying slave iteration cutoffs.

for both responses increase as either the number of flights and/or the slave iteration cutoff is increased, with better response requiring less runtime for all but one setting tested. Solution quality results are presented in Figure 6.3(b), where the x-axis again indicates the number of flights and slave response type, but now the y-axis is the screener utility of the solution returned when the slave iteration cutoff is reached. We observe for both response types that the screener utility increases as the slave iteration cutoff is increased, as well as that better response achieves a higher screener utility than best response in all cases. While the first result is intuitive, the second result is perhaps not. While better response may produce suboptimal pure strategies with respect to helping minimize the one-norm distance, the randomness in the better response may provide a more diverse set of pure strategies, resulting in higher screener utility.

### 6.4.4 Uncertainty

To evaluate our approach for handling uncertainty, we consider 50 possible screenee distributions and assume a uniform probability for each of the distributions being realized. Figure 6.4 presents the space of tradeoffs for the resulting screening strategies, with the average percentage of overflow screenees on the x-axis and the average screener utility regret on the y-axis.

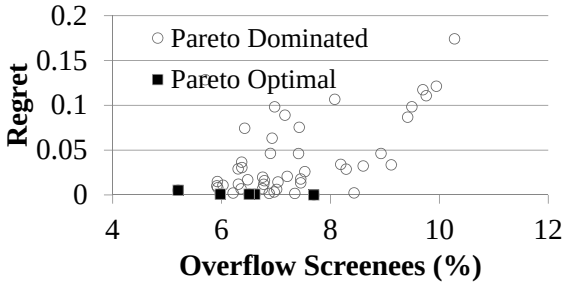



Figure 6.4: Tradeoff between overflow screenees and solution quality loss of different screening strategies when handling passenger distribution uncertainty.

Of the 50 screening strategies, only 4 reside on the Pareto frontier and should be considered by the screener. Within the Pareto optimal screening strategies, we observe that accepting slightly higher average regret can reduce the average percentage of overflow screenees from almost 8% to around 5%. Performing this kind of analysis allows the screener to make a more informed decision and select a screening strategy that is more robust to uncertainty.

## 6.5 Chapter Summary

We have introduced a model for TSGs that effectively utilizes limited screening resources. Additionally, we proved theoretical properties of TSGs and presented algorithms for computing the optimal screening strategy. While we used physical screening to motivate our model, where we are engaged in joint work with the TSA on airport passenger screening, TSGs are applicable to any type of domain where a strategic adversary is trying to pass through a screening process. Beyond the (1) TSG model, our contributions are (2) an NP-hardness proof for computing the equilibrium of TSGs, (3) a decomposing scheme for TSGs; (4) a column generation approach to solve TSGs; and (5) a minimax regret-based tradeoff analysis for handling uncertainty.

# Chapter 7: Multiple Adversary Objectives (Bounded Rationality)

Incorporating human behavioral models [McKelvey and Palfrey, 1995; Camerer, 2003] into security games represents an important progression that has been demonstrated to improve the performance of defender patrol strategies in both simulations and human subject experiments [Pita et al., 2010; Yang et al., 2012, 2013; Nguyen et al., 2013]. Behavioral models allow for the relaxation of the one of the strongest assumptions in classical game theory: namely, that the adversary is a perfectly rational utility maximizer. Instead, behavioral models, such as the quantal response (QR) model [McKelvey and Palfrey, 1995] and the subjective utility quantal response (SUQR) model [Nguyen et al., 2013], feature stochasticity in human decision making. These models are able to better predict the actions of real human adversaries and thus lead the defender to choose strategies that perform better in practice. Boundedly rational human behavioral models raise two fundamental research challenges that previous work has tried to address separately: scalability and robustness.

While perhaps counter-intuitive, modeling adversaries which behave suboptimally actually makes the defender's optimization problem computationally more difficult. Both QR and SUQR are non-linear models and are difficult to use directly in large-scale security domains. This issue of scalability for large-scale security games with boundedly rational adversaries has received

attention in the literature. [Yang et al., 2012] presented a mixed-integer linear programming (MILP) approximation for QR and SUQR models which improves tractability. Additionally, [Yang et al., 2013] introduces a cutting planes approach which can handle general patrol schedules and uses a master-slave formulation to iteratively generate deep cuts. We emphasize that the work [Yang et al., 2012, 2013] only allows for a single boundedly rational adversary.

However, in many domains the defender could encounter multiple different types of boundedly rational human adversaries. Thus, a separate line of security games research has focused on achieving robustness against uncertainty in the true adversary model. [Yang et al., 2014] proposed a Bayesian approach which learns a Gaussian distribution over adversary types. This approach has two potential drawbacks. First, the assumption that the adversary types are normally distributed is difficult to justify in practice. Second, even if the adversaries are normally distributed, a large amount of data is needed to learn the Gaussian distribution. Alternatively, [Haskell et al., 2014] introduced a *maximin* approach which does not use a distribution over the adversary types. Instead, the defender chooses a patrol that maximizes the worst-case expected defender reward over a set of adversary types. In an effort to scale up, [Yang et al., 2014; Haskell et al., 2014] focused on security games with a simplified defender strategy space that do not have complicated patrol schedules.

My thesis merges these two research threads for the first time by addressing scalability and robustness simultaneously. Each thread alone is impractical for important real-world security domains, such as environmental crime. Security games with complicated patrol schedules *and* multiple boundedly rational adversary types present a number of modeling and computational challenges. However, overcoming these challenges is critical as they are precisely the characteristics that define real-world security games. Our main contribution here is MIDAS (MaxImin

Defense Against SUQR) which computes robust defender patrols for large-scale security games with a heterogeneous adversary population. Building off the insights of [Yang et al., 2012, 2013, 2014; Haskell et al., 2014], we offer two key innovations: (i) a *robust* model that generates patrols that hedge against uncertainty about a heterogeneous population of adversaries and (ii) a *tractable* MILP approximation of our robust problem. We develop key theoretical properties of MIDAS and also compare MIDAS against previous approaches in simulation.

In collaboration with the United States Coast Guard (USCG), we have applied MIDAS to protect fisheries in the Gulf of Mexico, where illegal, unreported, and unregulated (IUU) fishing seriously threatens the health of local fish stocks. The USCG has both surface and air assets with which to deter IUU fishing. We frame the interaction between the USCG and illegal fisherman from Mexico (henceforth called Lanchas) as a Stackelberg security game. By using historical data on Lancha sightings, we learn and construct a set of SUQR adversary types. However, there is not sufficient data to accurately construct a probability distribution over Lancha types. Generation of robust defender strategies for this domain has previously been explored in [Haskell et al., 2014]. However, that work was more of a hot spot prediction model and it did not account for actual USCG schedules. In contrast, MIDAS constructs patrol schedules directly, resulting in higher quality patrol schedules for the USCG. The USCG began live testing of patrol schedules generated using MIDAS in July 2014.

## 7.1 Related Work

Game theory has been successfully applied to security problems such as the protection of networks [Manshaei et al., 2013; Nguyen et al., 2009; Píbil et al., 2012] and physical infrastructure

[Tambe, 2011]. In particular, the Stackelberg game model with its leader-follower paradigm has been used extensively in security domains. Stackelberg games capture the fact that, in the real world, the defender (i.e., the security agency) must commit first to a strategy that may be observed and then exploited by adversaries. Given this first mover advantage, it is critical to understand and predict how adversaries will respond to a given strategy in order to find the best strategy. Classical game theory assumes that the adversary is perfectly rational and will always select the best available action in response to the defender's strategy. In some domains, such as network security [Clark et al., 2012; Lu et al., 2013], this assumption is reasonable as the game is played by software agents. For other domains, particularly those with human adversaries, a theoretically optimal defender strategy under standard rationality assumptions can perform poorly in practice. Under the assumption of perfect rationality, the adversary will always select just one action (the utility maximizing action). This assumption can lead to non-robust strategies for the defender.

As such, human behavioral models are becoming an increasingly important aspect of security games research. [Yang et al., 2012] was the first to address human adversaries in security games by incorporating the quantal response (QR) model [McKelvey and Palfrey, 1995] from the social psychology literature. QR predicts a probability distribution over adversary actions where actions with higher utility have a greater chance of being chosen. By anticipating possible adversary deviation from the optimal action, strategies computed with QR are more robust to uncertainty in human decision making. [Jiang et al., 2013a] generalized the QR model to be robust against all adversary models satisfying monotonicity (i.e., higher utility actions are selected more frequently than lower utility actions), but this approach struggles to scale up to larger security games. [Nguyen et al., 2013] extended the QR model by proposing that humans use "subjective utility", a weighted linear combination of factors (such as defender coverage, adversary reward,

and adversary penalty), to make decisions. [Nguyen et al., 2013] proposes the subjective utility quantal response (SUQR) model which was shown to outperform QR in predicting the actions of participants of human subject experiments, thus leading to better defender strategies.

Building off that foundation, [Yang et al., 2013] presented an efficient cutting planes approach for solving security games with a large defender strategy space and a single adversary following a QR model. Meanwhile, two approaches have emerged for handling security games with multiple human adversary types. [Yang et al., 2014] utilized a Bayesian approach which learns a distribution over a set of SUQR types from available data. This distribution was assumed to be normal so as to minimize the number of parameters that need to be learned. Alternatively, [Haskell et al., 2014] developed a robust version of [Yang et al., 2014] and applies it to the fishery protection domain where only limited data about the adversaries is available. Borrowing from the robust optimization literature [Ben-Tal and Nemirovski, 2002; Bertsimas et al., 2011], a *maximin* approach is used to optimize defender expected utility against the worst-case type from the set of possible adversary types. However, [Yang et al., 2013] handles only one adversary type, while [Yang et al., 2014] and [Haskell et al., 2014] both fail to scale up. Neither of these two threads of research is individually able to handle the needs of security game applications in real-world domains such as environmental crime.

Most security problems do not feature static deployments, but rather have dynamic deployments that evolve in time and space. Thus, it is imperative to consider the capabilities of and restrictions on security resources such as personnel, cars, boats, and aircraft. Additionally, the adversaries in most physical security domains are likely to be humans, who have biases and limitations in their decision making process. This bounded rationality makes it difficult to predict the

actions of the adversary and in turn for the defender to optimize their strategy. As a further complication, rather than a single adversary type there is usually a set of potential adversary types that may be encountered and it is critical to be robust against uncertainty in adversary type. Prior work on boundedly rational adversaries in security games has addressed only one of the challenges of scalability and robustness.

My thesis proposes MIDAS which improves upon prior work by providing a holistic model that better captures the practicalities of large-scale, real-world security domains. More specifically, MIDAS enhances the incremental cut generation technique for solving large-scale security games with a single boundedly rational adversary type from [Yang et al., 2013] by using a robust *maximin* formulation for handling the uncertainty posed by multiple potential boundedly rational adversary types. Additionally, the QR model used in [Yang et al., 2013] for modeling boundedly rational adversary types is replaced with the SUQR model. Thus, MIDAS addresses the challenges of both scalability and robustness simultaneously, representing the first and only approach for solving security games with patrols schedules *and* multiple boundedly rational adversary types.

## 7.2 Background

We consider a Stackelberg security game (SSG) where the defender uses $M$ available resources to protect a set of targets $T = \{1, \ldots, |T|\}$ from a set of boundedly rational adversaries $\Omega$. For the remainder of this chapter we will focus on the SUQR behavioral model and treat $\omega \in \Omega$ as an SUQR adversary type. SUQR outperforms QR and other human behavioral models in human

subject experiments. As a result, SUQR is widely considered to be the state of the art for modeling boundedly rational adversaries in security games.

Each target $t \in T$ is assigned a set of payoffs $\{R_t^a, P_t^a, R_t^d, P_t^d\}$: $R_t^a$ is the reward earned by an adversary if they successfully attack target $t$, while $P_t^a$ is the penalty received by an adversary for an unsuccessful attack on target $t$. Conversely, if the defender assigns a resource to protect target $t$ and an adversary attacks target $t$, the defender receives a reward $R_t^d$. If an adversary attacks target $t$ and the defender has not assigned a resource to protect target $t$, the defender receives a penalty $P_t^d$. It should be noted that the payoffs for all adversary types in $\Omega$ are identical, it is the parameters of the SUQR behavioral model that distinguish between types in $\Omega$.

The defender commits to a mixed strategy that the adversaries are able to observe and then respond to by choosing a target to attack (Korzhyk, Conitzer, and Parr 2010; Basilico, Gatti, and Amigoni 2009). We denote the $j^{th}$ defender pure strategy as $A_j$, which is an assignment of all the security resources. $A_j$ is represented as a column vector $A_j = \langle A_{tj} \rangle^T$, where $A_{tj}$ indicates whether target $t$ is covered by $A_j$. For example, in an SSG with 4 targets and 2 resources, $A_j = \langle 1, 1, 0, 0 \rangle$ represents the pure strategy of assigning one resource to target 1 and another to target 2. Let $\mathcal{A} = \{A_j\}$ be the collection of feasible assignments of resources, i.e., the set of defender pure strategies. The defender's mixed strategy can then be represented as a vector $\mathbf{a} = \langle a_j \rangle$, where $a_j \in [0, 1]$ is the probability of choosing $A_j$. For large-scale security games, the number of pure strategies can grow so large that $\mathcal{A}$ cannot be represented explicitly in practice

making it impossible to optimize $\mathbf{a}$ directly. However, there is a more compact "marginal" representation for defender strategies. Let $\boldsymbol{x}$ be the marginal strategy, where $x_t = \sum_{A_j \in \mathcal{A}} a_j A_{tj}$ is the probability that target $t$ is covered. The set of all feasible marginal distributions is

$$\mathcal{X}_f = \left\{ \boldsymbol{x} : x_t = \sum_{A_j \in \mathcal{A}} a_j A_{tj},\, t \in T,\, \sum_{A_j \in \mathcal{A}} a_j = 1,\, \boldsymbol{a} \geq 0 \right\}.$$

We treat $\omega \in \Omega$ as an SUQR adversary type with the weight vector $\omega = \{\omega_1, \omega_2, \omega_3\}$ which encodes the relative importance of $x_t$, $R_t^a$, and $P_t^a$, respectively, in the decision making process of the adversary. Recall that the SUQR model selects a probability distribution over adversary actions rather than deterministically selecting the utility maximizing adversary action. Given defender strategy $\boldsymbol{x}$, the probability that adversary $\omega$ will attack target $t$ is

$$q_t\left(\omega \,|\, \boldsymbol{x}\right) = \frac{e^{\omega_1 x_t + \omega_2 R_t^a + \omega_3 P_t^a}}{\sum_{t'} e^{\omega_1 x_{t'} + \omega_2 R_{t'}^a + \omega_3 P_{t'}^a}}.$$

If an adversary chooses to attack target $t$, then for a given defender strategy $\boldsymbol{x}$, the defender's expected utility is

$$U_t\left(\boldsymbol{x}\right) = x_t R_t^d + \left(1 - x_t\right) P_t^d.$$

Against a known adversary type $\omega \in \Omega$, the defender's optimization problem is then

$$\max_{\boldsymbol{x} \in \mathcal{X}} F\left(\boldsymbol{x} \,|\, \omega\right) \triangleq \sum_t U_t\left(\boldsymbol{x}\right) q_t\left(\omega \,|\, \boldsymbol{x}\right), \tag{7.1}$$

which can be solved for a defender mixed strategy $\boldsymbol{a}$. However, in this chapter we consider an entire population of heterogeneous adversaries in $\Omega$. Thus, the optimization problem above is inadequate.

## 7.3 Adversary Uncertainty

### 7.3.1 Bayesian Estimation

If we have a distribution $\mathbb{P}$ over the set $\Omega$ of all possible types, then the expected utility maximizing problem is

$$\max_{x \in \mathcal{X}_f} \int_{\Omega} F\left(\boldsymbol{x} \mid \omega\right) \mathbb{P}\left(d\omega\right). \tag{7.2}$$

Problem (7.2) maximizes the expected defender utility, where the expectation is over the adversary types. In practice Problem (7.2) requires $\mathbb{P}$ to be estimated from sample data. Estimation of $\mathbb{P}$ presents two potential issues: first, it assumes that the types in $\Omega$ are normally distributed in order to use convenient update rules; second, large amounts of data are required. This method is referred to as Bayesian SUQR [Yang et al., 2014].

### 7.3.2 Maximin

Robust optimization offers up remedies for the shortcomings of Bayesian SUQR. *Maximin* does not require large amounts of data, but it can still utilize data when it is available even if only in small quantities. It is also less sensitive to assumptions about the nature of the underlying data, for instance the assumption that $\mathbb{P}$ is a normal distribution.

We treat $\Omega$ as an uncertainty set in line with robust optimization. For convenience, we assume that $\Omega$ is finite. This assumption is reasonable in practice since we will only ever have finitely many observations of the adversary. Then we solve the robust optimization problem

$$\max_{x \in \mathcal{X}_f} \min_{\omega \in \Omega} F\left(\boldsymbol{x} \,|\, \omega\right) \tag{7.3}$$

to get a patrol for the defender, where again $F\left(\boldsymbol{x} \,|\, \omega\right)$ is the expected utility corresponding to type $\omega$. Problem (7.3) is a nonlinear, nonconvex, nonsmooth optimization problem. For easier implementation, we transform Problem (7.3) into the constrained problem

$$\max_{x \in \mathcal{X}_f,\, s \in \mathbb{R}} \left\{ s : s \leq F\left(\boldsymbol{x} \,|\, \omega\right), \forall \omega \in \Omega \right\}, \tag{7.4}$$

by introducing a dummy variable $s \in \mathbb{R}$ to replace the nonsmooth objective with a collection of smooth constraints.

## 7.4 Mixed-Integer Linear Programming

By considering a human behavior model such as SUQR, Problem (7.4) becomes a nonlinear nonconvex optimization problem. In the general case, this problem class has been shown to be NP-hard to solve to optimality. Our idea in this section is to introduce a tractable MILP approximation scheme.

An approximate approach for solving Problem (7.1) with a single boundedly rational adversary was presented in [Yang et al., 2012, 2013]. This approach is based on a piecewise linear approximation that leads naturally to an MILP. In this section, we generalize this approach to

create MIDAS, an algorithm for solving the robust Problem (7.4) with a set of boundedly rational adversaries.

First notice that, $F(\boldsymbol{x} \mid \omega)$, the defender's payoff against a single adversary type $\omega \in \Omega$ can be written out as

$$F(\boldsymbol{x} \mid \omega) = \sum_t U_t(\boldsymbol{x})\, q_t(\omega \mid \boldsymbol{x}) = \frac{\sum_t \left(\left(R_t^d - P_t^d\right) x_t + P_t^d\right) e^{\omega_1 x_t + \omega_2 R_t^a + \omega_3 P_t^a}}{\sum_t e^{\omega_1 x_t + \omega_2 R_t^a + \omega_3 P_t^a}}$$

which is a fractional function $N(\boldsymbol{x} \mid \omega) / D(\boldsymbol{x} \mid \omega)$ where

$$N(\boldsymbol{x} \mid \omega) = \sum_t \left(\left(R_t^d - P_t^d\right) x_t + P_t^d\right) e^{\omega_1 x_t + \omega_2 R_t^a + \omega_3 P_t^a}$$

and $D(\boldsymbol{x} \mid \omega) = \sum_t e^{\omega_1 x_t + \omega_2 R_t^a + \omega_3 P_t^a}$. The goal in this section is to estimate the optimal value, which we will denote $s^*$, of Problem (7.4), i.e., the defender receives a payoff of at least $s^*$ against every adversary type $\omega \in \Omega$. We use a binary search to compute $s^*$ by updating a parameter $r$. We know that $r \leq s^*$ if there exists some $\boldsymbol{x} \in \mathcal{X}_f$ such that

$$r \leq \frac{N(\boldsymbol{x} \mid \omega)}{D(\boldsymbol{x} \mid \omega)}, \forall \omega \in \Omega.$$

Equivalently, we can rearrange the terms to require

$$r\, D(\boldsymbol{x} \mid \omega) - N(\boldsymbol{x} \mid \omega) \leq 0, \forall \omega \in \Omega.$$

Therefore, to check if $r \leq s^*$, we solve

$$\min_{x \in \mathcal{X}_f,\, \xi \in \mathbb{R}} \left\{ \xi : \xi \geq r\, D\left(\boldsymbol{x} \,|\, \omega\right) - N\left(\boldsymbol{x} \,|\, \omega\right),\, \forall \omega \in \Omega \right\}. \tag{7.5}$$

If the optimal value of the above problem is less than or equal to zero, then $r \leq s^*$; otherwise, $r > s^*$; then $r$ is adjusted appropriately. However, Problem (7.5) is still nonlinear and nonconvex. Thus, we need to find a tractable approximation to implement this scheme.

### 7.4.1 Linear Approximation

The nonlinearity and nonconvexity of Problem (7.5), whose objective function is a summation of nonlinear functions in $\boldsymbol{x}$, can be overcome by approximating each nonlinear function with a piecewise linear function with $K$ pieces. The functions $r\, D\left(\boldsymbol{x} \,|\, \omega\right) - N\left(\boldsymbol{x} \,|\, \omega\right)$ in the constraints of Problem (7.5) can be approximated with piecewise linear functions $L\left(\boldsymbol{x} \,|\, \omega\right)$ of the form:

$$L\left(\boldsymbol{x} \,|\, \omega\right) = \sum_{t \in T} \left(r - P_t^d\right) \left( f_t(0|\omega) + \sum_{k=1}^{K} \gamma_{\omega t k} x_{tk} \right) - \sum_{t \in T} \left(R_t^d - P_i^d\right) \sum_{k=1}^{K} \mu_{\omega t k} x_{tk}$$

where $\gamma_{\omega tk}$ is the slope of the function $f_t(x_t|w)$ in the $k^{th}$ segment while $\mu_{\omega tk}$ is the corresponding slope of $x_t f_t(x_t|\omega)$. With this approximation, we then solve the feasibility check problem

$$\min_{\boldsymbol{x},\xi} \xi \tag{7.6}$$

$$\text{s.t.}\, \xi \geq L\left(\boldsymbol{x} \,|\, \omega\right), \quad \forall \omega \in \Omega, \tag{7.7}$$

$$0 \leq x_{tk} \leq 1/K, \quad \forall t, \quad k = 1 \ldots K, \tag{7.8}$$

$$z_{tk}/K \leq x_{tk}, \quad \forall t, \quad k = 1 \ldots K-1, \tag{7.9}$$

$$x_{t(k+1)} \leq z_{tk}, \quad \forall t, \quad k = 1 \ldots K-1, \tag{7.10}$$

$$z_{tk} \in \{0, 1\}, \quad \forall t, \quad k = 1 \ldots K-1, \tag{7.11}$$

$$x_t = \sum_{A_j \in \mathcal{A}} a_j A_{tj}, \quad \forall t, \tag{7.12}$$

$$\sum_{A_j \in \mathcal{A}} a_j = 1, \tag{7.13}$$

$$\boldsymbol{x},\, \boldsymbol{a} \geq 0. \tag{7.14}$$

### 7.4.2 Column Generation

In this subsection we produce a tractable scheme for solving Problem (7.6) - (7.14). First, we derive a relaxation of Problem (7.6) - (7.14). Second, we show how to iteratively improve this approximation via a network flow problem: to that end Problem (7.6) - (7.14) is used to add new constraints to the relaxed version of the problem, and column generation is used in service of solving Problem (7.6) - (7.14) which then uses the network flow representation. Our network flow problem differs substantially from earlier work, which focused on aviation security and environmental crime, because of the generality of our formulation.

To begin, we approximate the constraint $\boldsymbol{x} \in \mathcal{X}_f$ with a linear relaxation

$$\left\{\boldsymbol{x} : \hat{H}\,\boldsymbol{x} \leq \hat{h}\right\},$$

which represents a subset of linear boundaries of $\mathcal{X}_f$. Then we solve the relaxation

$$\max_{\boldsymbol{x},\, s \in \mathbb{R}} \left\{ s : s \leq F\left(\boldsymbol{x} \,|\, \omega\right),\, \forall \omega \in \Omega,\, \hat{H}\,\boldsymbol{x} \leq \hat{h} \right\} \tag{7.15}$$

using the binary search method, i.e. Problem (7.6) - (7.14).

Given a candidate $\tilde{\boldsymbol{x}}$, we check if $\tilde{\boldsymbol{x}} \in \mathcal{X}_f$ by solving the projection problem

$$\min_{z \in \mathbb{R}^{|T|},\, \boldsymbol{a} \in \mathbb{R}^J} \sum_{t \in T} z_t \tag{7.16}$$

$$\text{s.t. } A\,\boldsymbol{a} - \tilde{\boldsymbol{x}} \leq z, \tag{7.17}$$

$$-\,z \leq A\,\boldsymbol{a} - \tilde{\boldsymbol{x}}, \tag{7.18}$$

$$\sum_{A_j \in \mathcal{A}} a_j = 1, \tag{7.19}$$

$$\boldsymbol{a} \geq 0. \tag{7.20}$$

Problem (7.16) - (7.20) finds the best 1-norm approximation of $\boldsymbol{x}$ in $\mathcal{X}_f$, and returns the optimal value zero if $\boldsymbol{x} \in \mathcal{X}_f$. Otherwise, we find a violated constraint which we add to the approximation $\hat{H}\,\boldsymbol{x} \leq \hat{h}$.

Problem (7.16) - (7.20) has a large number of variables since $\mathcal{A}$ is exponentially large. We solve (7.16) - (7.20) using a column generation method similar to the one introduced in [Jain et al., 2010a]. We solve a restriction of Problem (7.16) - (7.20) with a subset of columns $\hat{A} \subset A$

where $a$ is now understood as a vector in $a \in \mathbb{R}^{|\hat{A}|}$, with $a_j = 0$ for all $j$ with $A_j \notin \hat{A}$. Then we check for columns $A_j$ to add to $\hat{A}$ by computing the reduced costs of variables $a_j$ with $A_j \notin \hat{A}$ via the dual problem.

The dual to Problem (7.16) - (7.20) is

$$\max_{y,\, u} \tilde{x}^T y + u \tag{7.21}$$

$$\text{s.t. } A^T y + u \leq 0, \tag{7.22}$$

$$-1 \leq y \leq 1, \tag{7.23}$$

which has a large number of constraints due to the presence of the matrix $A$. For a subset of columns $\hat{A} \subset A$ (abusing notation since these are matrices), we have the relaxation of the dual

$$\max_{y,\, u} \tilde{x}^T y + u \tag{7.24}$$

$$\text{s.t. } \hat{A}^T y + u \leq 0, \tag{7.25}$$

$$-1 \leq y \leq 1, \tag{7.26}$$

$$g \geq 0. \tag{7.27}$$

We are looking for a column $A_j$ such that

$$A_j^T y + u \leq 0$$

is violated. So, we solve the slave problem

$$\max_{A_j \in \mathcal{A}} \left\{ y^T A_j \right\} + u \tag{7.28}$$

and identify a violated constraint if the optimal value of this problem is positive. Specifically, we solve Problem (7.28) using the technique in [Jain et al., 2010a], i.e. we use a maximum reward network flow problem (since Problem (7.28) is a maximization problem).

To setup this network flow problem, we create a source node with supply 1, and a sink node with demand 1. We have a fixed time horizon, $n = 0, 1, \ldots, N$ stages, so we create a node $(n, t)$ for every target and every time. The variables in this problem are the flow between nodes,

$$\mu_{(t,n),\,(t',n+1)}$$

which indicate a transition in the asset from target $t$ at time $n$ to target $t'$ at time $n+1$ in the next period. Effectively, we are taking a transition graph representation on the state space $T^{N+1}$. This formulation has the advantage of allowing us to express constraints on feasible patrols. The maximum reward network flow problem is then of the form

$$\max_{\mu} \left\{ \sum_{n \in \mathbb{N}} y_t \sum_{n,t} \mu_{(t,n),\,(t',n+1)} : \text{network flow constraints on } \mu \right\}.$$

The preceding network flow problem is a linear programming problem. This problem class is well studied and many efficient solution algorithms (such as the Simplex algorithm) exist that can obtain an exact optimal solution. We also point out that the preceding network flow problem can be solved efficiently for any underlying network topology.

## 7.5 Problem Properties

This section summarizes some key problem properties. The main points are to better understand our approximation scheme, to confirm that our cut generation scheme produces deep cuts, and to see how the standard Bayesian estimation approach relates to our robust approach.

### 7.5.1 MILP Approximation Error

Our underlying approach is a piecewise linear approximation to a nonconvex problem. We want to better understand the error bound for this approximation and the resulting solution quality of the corresponding MILP. We will show that all of the nonconvex functions we are approximating have bounded Lipschitz constants. Thus, since their variability is bounded, we have an upper bound on the piecewise linear approximation error as a function of the fineness of the discretization.

Recall that we are approximating the feasibility check problem, which solves

$$\min_{x \in \mathcal{X}_f} \max_{\omega \in \Omega} \left\{ r\, D\left(\boldsymbol{x} \,|\, \omega\right) - N\left(\boldsymbol{x} \,|\, \omega\right) \right\},$$

by linearly interpolating the functions $r\, D\left(\boldsymbol{x} \,|\, \omega\right) - N\left(\boldsymbol{x} \,|\, \omega\right)$ for all $\omega \in \Omega$. The first step in our approximation analysis is to estimate the Lipschitz constant of $r\, D\left(\boldsymbol{x} \,|\, \omega\right) - N\left(\boldsymbol{x} \,|\, \omega\right)$ for fixed $\omega \in \Omega$.

**Lemma 2.** *The Lipschitz constant of $r\, D\left(\boldsymbol{x} \,|\, \omega\right) - N\left(\boldsymbol{x} \,|\, \omega\right)$ for any $\omega \in \Omega$ is bounded above by*

$$\sum_t e^{1+\max_t R_t^a + \max_t P_t^a} + \sum_t \left(R_t^d - P_t^d\right) e^{1+\max_t R_t^a + \max_t P_t^a}.$$

*Proof.* By direct computation, $r\,D\left(\boldsymbol{x}\,|\,\omega\right) - N\left(\boldsymbol{x}\,|\,\omega\right)$ is equal to

$$r\sum_t e^{\omega_1 x_t + \omega_2 R_t^a + \omega_3 P_t^a} - \sum_t \left(\left(R_t^d - P_t^d\right) x_t + P_t^d\right) e^{\omega_1 x_t + \omega_2 R_t^a + \omega_3 P_t^a}.$$

So

$$\begin{aligned}
&|r\,D\left(\boldsymbol{x}\,|\,\omega\right) - N\left(\boldsymbol{x}\,|\,\omega\right) - r\,D\left(\boldsymbol{x}'\,|\,\omega\right) + N\left(\boldsymbol{x}'\,|\,\omega\right)| \\
\leq &\sum_t |e^{\omega_1 x_t + \omega_2 R_t^a + \omega_3 P_t^a} - \sum_t \left(\left(R_t^d - P_t^d\right) x_t + P_t^d\right) e^{\omega_1 x_t + \omega_2 R_t^a + \omega_3 P_t^a} \\
&- e^{\omega_1 x_t' + \omega_2 R_t^a + \omega_3 P_t^a} - \sum_t \left(\left(R_t^d - P_t^d\right) x_t' + P_t^d\right) e^{\omega_1 x_t' + \omega_2 R_t^a + \omega_3 P_t^a}| \\
\leq &\sum_t |e^{\omega_1 x_t + \omega_2 R_t^a + \omega_3 P_t^a} - e^{\omega_1 x_t' + \omega_2 R_t^a + \omega_3 P_t^a}| \\
&+ \sum_t |\left(\left(R_t^d - P_t^d\right) x_t + P_t^d\right) e^{\omega_1 x_t + \omega_2 R_t^a + \omega_3 P_t^a} - \left(\left(R_t^d - P_t^d\right) x_t' + P_t^d\right) e^{\omega_1 x_t' + \omega_2 R_t^a + \omega_3 P_t^a}|.
\end{aligned}$$

We have

$$|e^{\omega_1 x_t + \omega_2 R_t^a + \omega_3 P_t^a} - e^{\omega_1 x_t' + \omega_2 R_t^a + \omega_3 P_t^a}| \leq e^{\omega_2 R_t^a + \omega_3 P_t^a} e^{\omega_1} |x_t - x_t'|.$$

Additionally,

$$\begin{aligned}
|x_t e^{\omega_1 x_t} - x_t' e^{\omega_1 x_t'}| \leq &|x_t e^{\omega_1 x_t} - x_t e^{\omega_1 x_t'}| + |x_t e^{\omega_1 x_t'} - x_t' e^{\omega_1 x_t'}| \\
\leq &x_t e^{\omega_1} |x_t - x_t'| + e^{\omega_1} |x_t - x_t'| \\
\leq &2 e^{\omega_1} |x_t - x_t'|.
\end{aligned}$$

Now use the fact that $e^{\omega_2 R_t^a + \omega_3 P_t^a} e^{\omega_1}$ is bounded above by

$$e^{1+\max_t P_t^a + \max_t R_t^a},$$

and $2e^{\omega_1}$ is bounded above by $2\,e$. Using Lemma 3 and the triangle inequality, for any $\boldsymbol{x},\ \boldsymbol{x}' \in \mathcal{X}_f$ we compute

$$\begin{aligned} &|\max_{\omega\in\Omega}\{r\,D\left(\boldsymbol{x}\,|\,\omega\right) - N\left(\boldsymbol{x}\,|\,\omega\right)\} - \max_{\omega\in\Omega}\left\{r\,D\left(\boldsymbol{x}'\,|\,\omega\right) - N\left(\boldsymbol{x}'\,|\,\omega\right)\right\}| \\ \leq\, & r\max_{\omega\in\Omega}|D\left(\boldsymbol{x}\,|\,\omega\right) - D\left(\boldsymbol{x}'\,|\,\omega\right)| + \max_{\omega\in\Omega}|N\left(\boldsymbol{x}\,|\,\omega\right) - N\left(\boldsymbol{x}'\,|\,\omega\right)|. \end{aligned}$$

□

We can expand on the previous Lipschitz computation to produce an error estimate for the overall piecewise linear approximation, by using the following fact to bound the Lipschitz constant of

$$\max_{\omega\in\Omega}\{r\,D\left(\boldsymbol{x}\,|\,\omega\right) - N\left(\boldsymbol{x}\,|\,\omega\right)\}.$$

**Lemma 3.** *Let $X$ be a given set, and $f_1 : X \to \mathbb{R}$ and $f_2 : X \to \mathbb{R}$ be two real-valued functions on $X$. Then,*

*(i)* $|\inf_{x\in X} f_1(x) - \inf_{x\in X} f_2(x)| \leq \sup_{x\in X}|f_1(x) - f_2(x)|$, *and*

*(ii)* $|\sup_{x\in X} f_1(x) - \sup_{x\in X} f_2(x)| \leq \sup_{x\in X}|f_1(x) - f_2(x)|$.

*Proof.* To verify part (i), note

$$
\begin{aligned}
\inf_{x\in X} f_1(x) &= \inf_{x\in X} \{f_1(x) + f_2(x) - f_2(x)\} \\
&\leq \inf_{x\in X} \{f_2(x) + |f_1(x) - f_2(x)|\} \\
&\leq \inf_{x\in X} \left\{ f_2(x) + \sup_{y\in Y} |f_1(y) - f_2(y)| \right\} \\
&\leq \inf_{x\in X} f_2(x) + \sup_{y\in Y} |f_1(y) - f_2(y)|,
\end{aligned}
$$

giving

$$
\inf_{x\in X} f_1(x) - \inf_{x\in X} f_2(x) \leq \sup_{x\in X} |f_1(x) - f_2(x)|.
$$

By the same reasoning,

$$
\inf_{x\in X} f_2(x) - \inf_{x\in X} f_1(x) \leq \sup_{x\in X} |f_1(x) - f_2(x)|,
$$

and the preceding two inequalities yield the desired result. Part (ii) follows similarly. □

### 7.5.2 Projection

The feasible region of our problem, $\mathcal{X}_f$, is exactly the same as the one found in [Yang et al., 2013]. Thus, the results of the cut generation algorithm are unchanged and we obtain deep cuts. The results are repeated here for completeness.

**Lemma 4.** *(i) If $\tilde{\boldsymbol{x}} \notin \mathcal{X}_f$, let $(\boldsymbol{y}^*, \boldsymbol{g}^*, u^*)$ be the dual variables at the optimal solution of Problem ((7.16)) - ((7.20)). Then the hyperplane $(\boldsymbol{y}^*)^T \boldsymbol{x} - (\boldsymbol{g}^*)^T \boldsymbol{b} + u^* = 0$ separates $\tilde{\boldsymbol{x}}$ and $\mathcal{X}_f$.*

*(ii) Furthermore, $(\boldsymbol{y}^*)^T \boldsymbol{x} - (\boldsymbol{g}^*)^T \boldsymbol{b} + u^* = 0$ is a deep cut.*

As in [Yang et al., 2013], we now consider a modified norm minimization problem. The idea is that we weight the norm towards an optimal solution using local rate of change information about the objective. In our case, the objective $G(\boldsymbol{x}) = \min_{\omega \in \Omega} F(\boldsymbol{x} \,|\, \omega)$ is a nondifferentiable function, so we use the subgradient instead of the gradient. The subgradient is

$$\partial G(\boldsymbol{x}) = \operatorname{conv}\left\{\nabla_{\boldsymbol{x}} F(\boldsymbol{x} \,|\, \omega) \,:\, F(\boldsymbol{x} \,|\, \omega) = G(\boldsymbol{x})\right\}.$$

For a subgradient $\boldsymbol{s} \in \partial G(\boldsymbol{x})$, we use the objective $\sum_t (\boldsymbol{s}_t + \xi)\, z_t$ where $\xi > 0$ is chosen so that $\boldsymbol{s}_t + \xi > 0$ for all $t$.

### 7.5.3 Duality

Here we comment on the relationship of our approach to Bayesian estimation. Bayesian estimation is a classical and widespread tool for incorporating information under uncertainty. To reveal this relationship, we compute the dual of the constrained variant of Problem (7.3) which we reprint here for convenience:

$$\max_{x \in \mathcal{X}_f,\, s \in \mathbb{R}} \left\{s : s \leq F(\boldsymbol{x} \,|\, \omega),\, \forall \omega \in \Omega\right\}.$$

The constraints above cause Lagrange multipliers to appear; so we can compute the standard Lagrangian dual. To proceed we first introduce the Lagrange multipliers which lie in $\mathbb{R}^{|\Omega|}$ (since there are only finitely many adversary types). We let $\mathbb{R}_+^{|\Omega|}$ denote the set of nonnegative vectors in $\mathbb{R}^{|\Omega|}$.

Let

$$\mathcal{P}(\Omega) \triangleq \left\{ \Lambda \in \mathbb{R}_{+}^{|\Omega|} : \sum_{\omega \in \Omega} \Lambda(\omega) = 1 \right\}$$

be the space of probability measures on $\Omega$, it is a subset of $\mathbb{R}^{|\Omega|}$. We will see shortly that these probability measures are the decision variables in the dual to Problem (7.4).

**Theorem 3.** *The dual to Problem (7.4) is*

$$\min_{\Lambda \in \mathcal{P}(\Omega)} \left\{ d(\Lambda) \triangleq \max_{x \in \mathcal{X}_f} \sum_{\omega \in \Omega} F(\boldsymbol{x} \,|\, \omega) \Lambda(\omega) \right\}. \tag{7.29}$$

*Proof.* Let $\Lambda \in \mathbb{R}_{+}^{|\Omega|}$ be the Lagrange multiplier for the constraint $s \leq F(\boldsymbol{x} \,|\, \omega)$ for all $\omega \in \Omega$. We obtain the Lagrangian

$$L(\boldsymbol{x},\, s,\, \Lambda) = s + \sum_{\omega \in \Omega} \left[ F(\boldsymbol{x} \,|\, \omega) - s \right] \Lambda(\omega).$$

The Lagrangian dual problem is then

$$\min_{\Lambda \in \mathbb{R}_{+}^{|\Omega|}} \max_{\boldsymbol{x} \in \mathcal{X}_f,\, s \in \mathbb{R}} \left\{ L(\boldsymbol{x},\, s,\, \Lambda) \right\}.$$

We see that the inner maximization problem $d(\Lambda)$ yields the implied constraint $\int_{\Omega} \Lambda(d\omega) = 1$ via

$$\max_{s \in \mathbb{R}} s \left( 1 - \sum_{\omega \in \Omega} \Lambda(\omega) \right),$$

which is equal to infinity unless the equality $\sum_{\omega\in\Omega}\Lambda(\omega) = 1$ holds. Thus, we have the dual problem

$$\min_{\Lambda\in\mathbb{R}_+^{|\Omega|}}\left\{\max_{x\in\mathcal{X}_f}\sum_{\omega\in\Omega}F(\boldsymbol{x}\,|\,\omega)\,\Lambda(\omega)\;:\;\sum_{\omega\in\Omega}\Lambda(\omega)=1\right\}.$$

□

We emphasize that the dual decision variables are prior distributions on the set of types. Notice that for any fixed $\Lambda\in\mathcal{P}(\Omega)$, we see that we have a Bayesian problem since we can treat $\Lambda$ as a prior distribution. For $\Lambda$, we can then perform Bayesian estimation as usual. Thus, we see that the dual problem is a search for the "best" prior distribution. As a corollary, we reason that standard Bayesian estimation gives us an upper bound on the optimal value to Problem (7.3).

**Corollary 1.** *(i)* $\max_{x\in\mathcal{X}_f}\min_{\omega\in\Omega}F(\boldsymbol{x}\,|\,\omega)\le\min_{\Lambda\in\mathcal{P}(\Omega)}d(\Lambda)$.

*(ii) Let* $\Lambda\in\mathcal{P}(\Omega)$ *be any prior distribution, then* $\max_{x\in\mathcal{X}_f}\min_{\omega\in\Omega}F(\boldsymbol{x}\,|\,\omega)\le d(\Lambda)$.

*Proof.* Follows from weak duality for Problem (7.4),

$$\max_{x\in\mathcal{X}_f,\,s\in\mathbb{R}}\{s : s\le F(\boldsymbol{x}\,|\,\omega),\,\forall\omega\in\Omega\}\le\min_{\Lambda\in\mathcal{P}(\Omega)}\max_{x\in\mathcal{X}_f}\sum_{\omega\in\Omega}F(\boldsymbol{x}\,|\,\omega)\,\Lambda(\omega)$$

which gives

$$\max_{x\in\mathcal{X}_f}\min_{\omega\in\Omega}F(\boldsymbol{x}\,|\,\omega)\le\min_{\Lambda\in\mathcal{P}(\Omega)}\max_{x\in\mathcal{X}_f}\sum_{\omega\in\Omega}F(\boldsymbol{x}\,|\,\omega)\,\Lambda(\omega)$$

since

$$\max_{x\in\mathcal{X}_f,\,s\in\mathbb{R}}\{s : s\le F(\boldsymbol{x}\,|\,\omega),\,\forall\omega\in\Omega\}=\max_{x\in\mathcal{X}_f}\min_{\omega\in\Omega}F(\boldsymbol{x}\,|\,\omega).$$

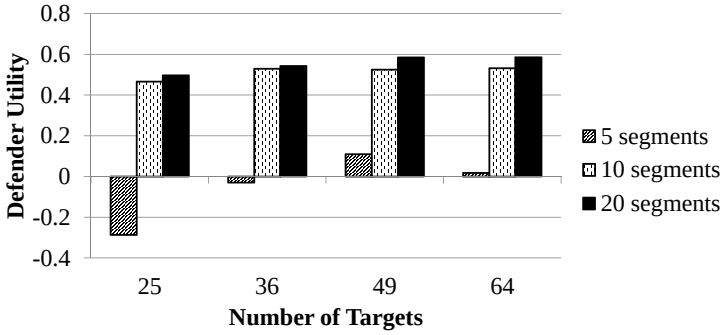
□

## 7.6 Evaluation

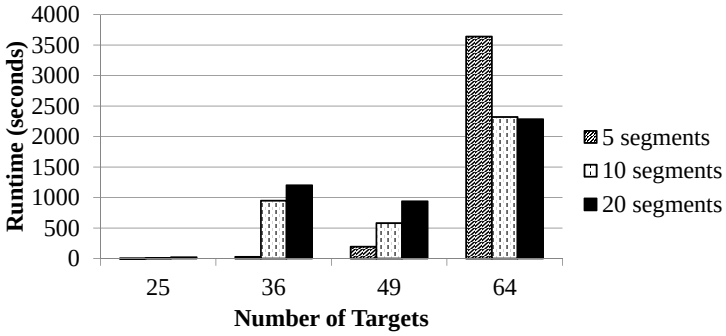
In this section, we evaluate MIDAS in the fishery protection domain, where the USCG must patrol the Gulf of Mexico to prevent Mexican fishermen (Lanchas) from entering the United States Exclusive Economic Zone (EEZ) and fishing illegally. The zero-sum Stackelberg game we consider is played on a square grid, where each grid cell is a potential target. The defender (USCG) commits to a mixed strategy over fixed length patrols, where each target can be visited at most once. Additionally, all patrols must start and end in the first row of the grid. Meanwhile, the Lanchas select their mixed strategies over targets based on the SUQR behavioral model where each adversary has a unique weight vector $\omega$. For our experiments, the game payoffs are randomly generated with $R_t^a$ uniformly distributed in [1,10] and $P_t^d$ uniformly distributed in [-10,-1]. The remaining game payoffs, $R_t^d$ and $P_t^a$, are fixed at 10 and -10, respectively. Note that $R_t^a$ and $P_t^a$ are the same for all adversaries. All the adversary types $\omega \in \Omega$ used in the experiments were learned from USCG data. The default settings for each experiment are: five piecewise linear segments, a set of ten adversary types (i.e., $|\Omega| = 10$), and a patrol length equal to half the number of targets rounded down (i.e. $\lfloor \frac{|T|}{2} \rfloor$). We varied the dimensions of the square grid from $5 \times 5$ to $8 \times 8$ and created thirty randomly generated game instances for each grid size.

### 7.6.1 Linear Approximation

In MIDAS, we use a linear approximation to estimate the nonlinear SUQR behavioral model. The classic tradeoff when using approximation techniques is between solution quality and runtime. Thus, it is important to understand how the granularity of the approximation affects the performance of MIDAS. Figure 7.1(a) shows how varying the number of segments (5, 10, and




(a) Defender Utility




(b) Runtime

Figure 7.1: Effect of the number of piecewise linear segments on the solution quality and the runtime of the MIDAS algorithm.

20) used in the linear approximations impacts the defender's utility. The $x$-axis indicates the size of the grid, while the $y$-axis is the *maximin* utility obtained by the defender mixed strategy computed by MIDAS. For all grid sizes, we observe that increasing the number of segments results in higher utility for the defender as we would expect. In particular, going from 5 to 10 segments has a significant impact on the defender utility, whereas going from 10 to 20 segments produces diminishing returns and a much smaller improvement.

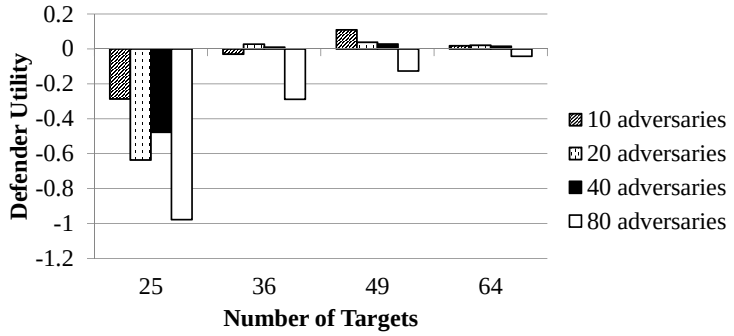
The other half of the tradeoff is how the number of segments impacts the runtime of MIDAS. Increasing the number of segments increases the number of variables and constraints in MIDAS, leading to a larger optimization problem which presumably would take longer to solve. The

results from varying the number of segments used in the linear approximation are shown in Figure 7.1(b). The x-axis again indicates the size of the grid, while the y-axis is now the runtime of MIDAS in seconds. For grid sizes $5 \times 5$ through $7 \times 7$, we see that the runtime increases as the number of segments is increased. However, for the $8 \times 8$ grid, MIDAS actually runs faster for 10 and 20 segments than it does with 5 segments. One possible explanation is that while each iteration of MIDAS algorithm takes longer to compute with more segments, the quality of the cuts generated by the separation oracle improves as the feasible marginal space is represented with higher granularity. Closer examination of the data for the $8 \times 8$ grid suggests that this is indeed the case as MIDAS with 5 segments averages with 125 calls to the separation oracle and patrol generation slave, while 10 and 20 segments average 82 and 70, respectively.

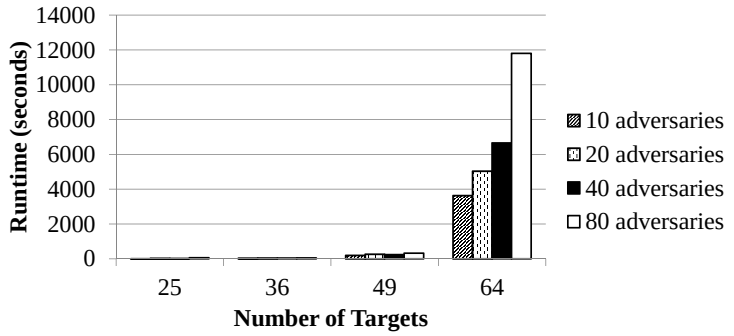
In practice, it is up to the end user to determine the right tradeoff between approximation quality and runtime. Our numerical experiments here offer guidance in this regard.

### 7.6.2 Adversary Types

The primary purpose of MIDAS is to provide a scalable approach for generating game-theoretic patrols protecting against a set of adversaries with complex human behavior models such as SUQR. Therefore, we want to evaluate the effect of the number of adversary types on MIDAS to ensure that it serves its intended function. In Figure 7.2(a), we present the results for the defender *maximin* utility obtained by varying the number of adversary types on different grid sizes. Given that MIDAS computes a robust *maximin* strategy, we would expect that the defender utility monotonically decreases as the set of adversary types expands, as each additional type could present a new possible worst case for the defender. While overall this trend holds, we occasionally observe that the defender utility increases as the size of $\Omega$ is increased. One possible




(a) Defender Utility




(b) Runtime

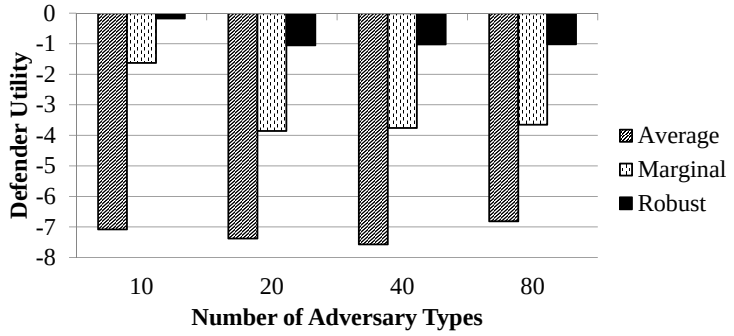
Figure 7.2: Effect of the number of adversary types on the solution quality and the runtime of the MIDAS algorithm.

explanation may be the interaction between the linear approximation and the robust maximin formulation. Using 5 piecewise segments may be leading to a coarse approximation in which the monotincity properties no longer hold. As with the number of piecewise linear segments, we would expect that increasing the number adversary types would also lead to an increase in the runtime. In Figure 7.2(b), we present the runtime results for MIDAS as the size of $\Omega$ is increased, which fall in line with our expectations. In particular, for the $8 \times 8$ grid we see a significant runtime increase as $\Omega$ is expanded. However, we also see that the runtimes are relatively constant for a small number of targets.

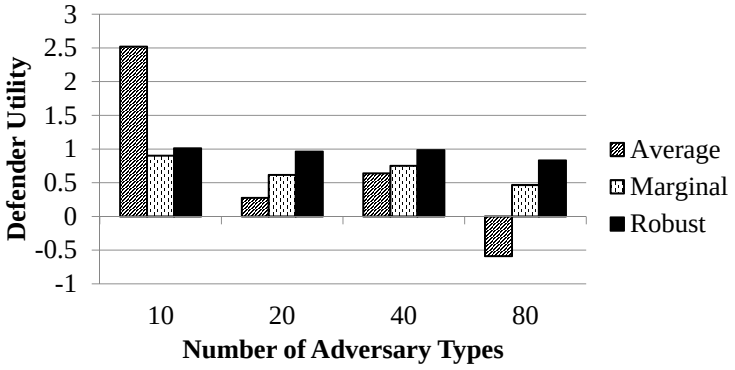
### 7.6.3 Approach Comparison

Thus far, we have evaluated the performance of MIDAS as the scale of security games is increased with respect to size of the grid or the size of $\Omega$. Now we want to compare how well MIDAS performs against other approaches that have introduced for solving security games with multiple boundedly rational adversaries. The first approach we will compare against is *Average*, in which a single adversary type $\omega_{avg}$ is constructed by averaging the weight vectors of the adversary types in $\Omega$. After obtaining $\omega_{avg}$, we can use MIDAS to solve the security game for $\Omega = \{\omega_{avg}\}$. The second approach we will compare against is *Marginal*, which is the robust *maximin* formulation from [Haskell et al., 2014] that ignores resource assignment constraints to produce a marginal coverage distribution over the targets. To compute the *Marginal* strategy, we run MIDAS for a single iteration which produces a marginal defender strategy without considering resource assignment constraints that is then mapped into a probability distribution over patrols using the one-norm projection. The third approach is *Robust* which involves running the MIDAS algorithm to completion.

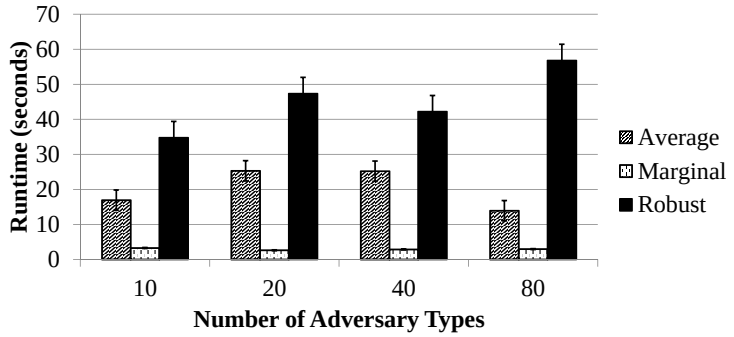
In Figure 7.3(a), we compare the worst case defender utility of the three approaches against sets of varying numbers of boundedly rational adversaries. The x-axis shows the number of adversary types in $\Omega$, while the y-axis indicates the worst case defender utility of the strategies computed by the different approaches against $\Omega$. Perhaps unsurprisingly, the *Average* approach performs the worst out of the three across all sizes of $\Omega$. The defender is optimizing against an artificially constructed adversary type $\omega_{avg}$ that is not in the set $\Omega$. By not considering the extreme points in $\Omega$, the resulting defender's strategy is highly susceptible to being exploited by at least one adversary type which would define the worst case defender utility. The *Marginal* approach




(a) Worst Case Defender Utility




(b) Average Case Defender Utility




(c) Runtime

Figure 7.3: Solution quality and runtime comparison of three approaches for handling heterogeneous populations of adversary types.

shows improvement by being robust against all the types in $\Omega$, even while it initially ignores the resource assignment constraints. Finally, *Robust* uses MIDAS to its full potential and shows additional benefit of considering resource assignment constraints by outperforming *Marginal* for

all sizes of $|\Omega|$. Figure 7.3(b) shows a similar analysis as before but now for average case defender utility. We can see that despite optimizing against the worst case, *Robust* provides consistent performance in the average case, outperforming both *Average* and *Marginal* for $|\Omega| > 10$

In addition to defender utility, runtime can provide another point of comparison between the three approaches, which we analyze in Figure 7.3(c). Here the x-axis again indicates the number of adversary types in $\Omega$, while the y-axis is now the runtime needed to generate the defender's strategy using each approach. One would expect that *Average*, considering one adversary type, would run faster than *Robust*, considering $|\Omega|$ adversary types. By considering more types, the defender's optimization becomes larger with more variables and constraints. Indeed, we observe that *Robust* takes longer than *Average* for all sizes of $\Omega$. The gap between the two approaches seems to grow as the number of adversaries is increased, particularly for $|\Omega| = 80$. However, the runtime improvement of *Average* is likely not enough to make up for the poor solution quality in real-world domains. Meanwhile, *Marginal* produces an essentially fixed runtime by solving only a single iteration of MIDAS and thus requires the least amount of runtime between the three approaches. Given the high stakes of real-world security domains, it is easy to imagine scenarios where security agencies would prefer the improved solution quality of *Robust* over the improved runtime of *Marginal*.

## 7.7 Chapter Summary

The use of bounded rationality models like QR and SUQR in security games is becoming increasingly popular in order to generate strategies that perform better against real human adversaries.

These models raise two main research challenges: (i) scalability when handling resource assignment constraints and (ii) robustness when handling multiple boundedly rational adversaries. Up to this point, previous work has addressed these challenges individually. My thesis addresses both scalability and robustness simultanesouly by introducing a new algorithm, MIDAS. The key feature of MIDAS is the combination of incremental cut generation with a robust *minimax* formulation. Our experiments demonstrate that MIDAS can scale up to security games with complex resource allocation constraints in the form of spatio-temporal patrols. Additionally, MIDAS outperforms previous approaches for protecting against multiple adversaries by providing better solution quality guarantees in terms of worst-case performance. The overall performance of MIDAS suggests that it represents the state of the art for complex security game with boundedly rational adversaries.

**Acknowledgments**: This research was supported by the United States Department of Homeland Security through the National Center for Risk and Economic Analysis of Terrorism Events (CREATE) under award number 2010-ST-061-RE0001 and MURI grant W911NF-11-1-0332.

# Chapter 8: Conclusion and Future Directions

The success of research on Stackelberg security games has produced a number of applications deployed in real world security domains to provide resource allocation decision support. Examples of these applications include ARMOR used at Los Angeles International Airport (LAX) to randomize road checkpoints and canine patrols [Pita et al., 2008], IRIS deployed by the United States Federal Air Marshals Service to assign air marshals to international flights [Tsai et al., 2009], PROTECT utilized by the United States Coast Guard to schedule boat patrols for protecting ports [Shieh et al., 2012], and TRUSTS developed for the Los Angeles Sheriffs Department to generate patrol schedules through the local metro system [Yin et al., 2012]. One commonality between these applications is that both the defender and the adversary are modeled as having a single objective which is to maximize their expected utility. However, the underlying decision making process in many real world security domains is inherently multi-objective and the assumption of the players optimizing a single objective may no longer be adequate in such settings.

My thesis focuses on modeling more of the complexity present in security domains and addresses the research challenges raised by introducing multiple objectives into security games. My research serves to remove the restriction of only modeling players with a single objective and allows for the development of decision aids that construct higher fidelity games models of

the underlying domain and offer finer granularity in the resulting analysis. My thesis is able to achieve this by providing the following contributions:

## 8.1 Contributions

- Multiple Defender Objectives : In order to capture the fact that the defender is explicitly considering multiple objective during the decision making process, I introduced a new model referred to as a multi-objective security game (MOSG). With multiple objectives, MOSGs do not have a single optimal solution but rather a space of compromise solutions known as the Pareto frontier. Thus, I presented the Iterative-$\epsilon$-Constraints algorithm which uses an iterative approach in generating a sequence of subproblems to systematically explore the solution space to find the Pareto frontier. To compute the individual solutions that make up the Pareto frontier, I introduced an exact approach for solving a mixed-integer linear programming formulation of each subproblem. Additional contributions include developing heuristics and approximate approaches that achieve speedup by exploiting the structure of MOSGs, increasing the scalability of Iterative-$\epsilon$-Constraints while providing solution quality guarantees on approximating the Pareto frontier.

- Multiple Adversary Objectives : In addition to considering multiple objectives, human adversaries in many security domains are often boundedly rational in their decision making and thus are not utility maximizing as is assumed by classical game theory. The behavior for such adversaries was captured using the Subjective Utility Quantal Response (SUQR) model and learning the weights associated with the different objectives from data collected

in real-world security domains. To handle such settings, I introduced the MIDAS algorithm which computes robust defender strategies for large-scale SSGs with heterogeneous populations of boundedly rational adversaries with multiple objectives. MIDAS is the first algorithm to address both robustness and scalability simultaneously for such SSGs through a novel combination of a robust maximin formulation and incremental strategy generation. This novel combination of features establishes MIDAS as the current state of art for solving complex security games featuring boundedly rational adversaries with multiple objectives.

## 8.2 Future Directions

### 8.2.1 Multiple Defender and Adversary Objectives

My thesis presents models and techniques for solving SSGs in which *either* the defender or the adversary are considering multiple objectives. Thus, a logical extension would be to address SSGs where *both* the defender and the adversary are trying to optimize multiple objectives. Such SSGs could be modeled and solved using the contributions provided by this thesis. However, scalability is already an challenge that had to be addressed when solving SSGs where only one of the players has multiple objectives. Thus, having both players with multiple objectives further exacerbates the challenge of scalability and would open up numerous research challenges that would require modeling and algorithmic contributions.

Given the imperative to find ways of addressing scalability, one possible direction to explore is parallelized computation. Iterative-$\epsilon$-Constraints produces the Pareto frontier by generating a sequence of constrained single-objective optimization problems (CSOP). Currently, those CSOPs

are solved sequentially using a recursive depth-first tree search. However, there is nothing preventing the CSOPs from being solved in parallel. Thus, as a first step, a queue could be created where the CSOPs are placed as they are generated, with multiple threads servicing that queue. Expanding this idea further could lead to a significant redesign in the way Iterative-$\epsilon$-Constraints explores the solution space, including possibly using multiple starting CSOPs as opposed to just one. Additionally, with each CSOPs being more difficult to solve now with multiple adversary objectives, it would be worthwhile to investigate the potential roles for pruning, heuristics, and coordination between the parallel processes to improve computational efficiency. For example, MIDAS uses incremental strategy generation to improve scalability, and it may be possible to share the strategies generated to solve one CSOP when solving subsequent CSOPs.

The potential impact of modeling a security game with multiple objectives for both the defender and the adversary is significant. At a high level, this new type of security game provides a mathematical framework which better approximates the type of strategic decision making problems found in the real world where it is inherent that multiple competing objectives, factors, constraints, etc. must be balanced. In the process, it opens up of a whole new set of domains which can be modeled as security games as well as the possibility to develop higher fidelity models for domains in which security games have already been applied.

### 8.2.2 Adversary Uncertainty

My thesis also presents models and techniques for addressing adversary uncertainty in the form of facing heterogeneous populations of boundedly rational adversaries with multiple objectives. The particular approach I proposed for handling such uncertainty was a robust maximin formulation which provides solution quality guarantees with respect to worst case performance. However,

the resulting strategies can be conservative with regards to performance in the average case. On the opposite end of the spectrum, a Bayesian approach provides the optimal solution if the exact distribution over the adversaries is known. However, learning such a distribution would require a significant amount of data that is either not available in real world settings or would incur considerable costs in terms of time and resources to acquire.

Thus, another possible direction of future research could be to explore alternative approaches for addressing adversary uncertainty. More specifically, it may be possible to develop an intermediate approach between the robust and Bayesian approaches which can exploit any available data to fullest extent possible while also remaining robust. Such an approach could start out as fully robust and then as more data about the adversaries is collected in the real word the more the adversary models in the game are refined. The idea being that the infusion of additional data serves to mitigate the adversary uncertainty and, as a result, the generated strategies become less and less conservative over time. The end result would be an enhanced version of the MIDAS algorithm which generates better defender strategies the longer it is in use.

A further expansion of this idea could involve incorporating the concepts of exploration and exploitation from the STREETS algorithm. By considering exploration, the defender strategy could be constructed to reduce the uncertainty over the adversaries, i.e., allocating the security resources in such a way so as to learn valuable information about the adversaries. Explicitly considering this type of value of information (VoI) when generating strategies could reduce the cost of acquiring a more accurate adversary model. Additionally, even after the model of the adversaries becomes more refined, exploration would ensure that MIDAS is continuing to learn and adapt to any emerging trends in adversary behavior.

# Bibliography


AAAE. Transportation security policy. Technical report, American Association of Airport Executives, 2014.

MA Abido. Environmental/economic power dispatch using multiobjective evolutionary algorithms. *IEEE Transactions on Power Systems*, 18(4):1529–1537, 2003.

Nicole Adler, Alfred Shalom Hakkert, Jonathan Kornbluth, Tal Raviv, and Mali Sher. Location-allocation models for traffic police patrol vehicles on an interurban network. *Annals of Operations Research*, pages 1–23, 2013.

Maria Joo Alves and Joo Clmaco. A review of interactive methods for multiobjective integer and mixed-integer programming. *European Journal of Operational Research*, 180(1): 99 – 115, 2007. ISSN 0377-2217. doi: 10.1016/j.ejor.2006.02.033. URL `http://www.sciencedirect.com/science/article/pii/S0377221706002384`.

B. An, J. Pita, E. Shieh, M. Tambe, C. Kiekintveld, and J. Marecki. GUARDS and PROTECT: next generation applications of security games. *ACM SIGecom Exchanges*, 10(1): 31–34, 2011a.

B. An, M. Tambe, F. Ordóñez, E.A. Shieh, and C. Kiekintveld. Refinement of strong stackelberg equilibria in security games. In *Proceedings of the 25th AAAI conference on Artificial Intelligence (AAAI'11)*, 2011b.

Rudolf Avenhaus, Morton Canty, D Marc Kilgour, Bernhard Von Stengel, and Shmuel Zamir. Inspection games in arms control. *European Journal of Operational Research*, 90(3):383–394, 1996.

Nicola Basilico, Nicola Gatti, and Francesco Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 57–64. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

Aharon Ben-Tal and Arkadi Nemirovski. Robust optimization–methodology and applications. *Mathematical Programming*, 92(3):453–480, 2002.

Dimitris Bertsimas, David B Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM review*, 53(3):464–501, 2011.

J. Blocki, N. Christin, A. Datta, A.D. Procaccia, and A. Sinha. Audit games. In *Proceedings of the 23rd international joint conference on Artificial Intelligence (IJCAI'13)*, pages 41–47, 2013.

J. Blocki, N. Christin, A. Datta, A.D. Procaccia, and A. Sinha. Audit games with multiple defender resources. In *Proceedings of the 29th AAAI conference on Artificial Intelligence (AAAI'15)*, 2015.

W.K.M. Brauers, E.K. Zavadskas, F. Peldschus, and Z. Turskis. Multi-objective decision-making for road design. *Transport*, 23(3):183–193, 2008.

K. Bringmann, T. Friedrich, F. Neumann, and M. Wagner. Approximation-guided evolutionary multi-objective optimization. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1198–1203, 2011.

Egon Brunswik. *The conceptual framework of psychology*, volume 1. Univ of Chicago Pr, 1952.

Colin Camerer. *Behavioral game theory: Experiments in strategic interaction*. Princeton University Press, 2003.

V. Chankong and Y.Y. Haimes. *Multiobjective decision making: theory and methodology*. North-Holland New York, 1983.

Andrew Clark, Quanyan Zhu, Radha Poovendran, and Tamer Başar. Deceptive routing in relay networks. In *Decision and Game Theory for Security*, pages 171–185. Springer, 2012.

C.A.C. Coello, G.B. Lamont, and D.A. Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Springer-Verlag New York Inc, 2007.

V. Conitzer and D. Korzhyk. Commitment to correlated strategies. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 632–637, 2011.

Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 82–90. ACM, 2006.

Jinshu Cui and Richard S John. Empirical comparisons of descriptive multi-objective adversary models in stackelberg security games. In *Decision and Game Theory for Security*, pages 309–318. Springer, 2014.

Daniela Pucci De Farias and Benjamin Van Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of operations research*, 29(3):462–478, 2004.

K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, 2001.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

J. P. Dickerson, G. I. Simari, V. S. Subrahmanian, and Sarit Kraus. A graph-theoretic approach to protect static and moving targets from adversaries. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, 2010.

M.E. Giuliano and M.D. Johnston. Multi-objective evolutionary algorithms for scheduling the James Webb space telescope. In *International Conference on Automated Planning and Scheduling (ICAPS)*, volume 8, pages 107–115, 2008.

YY Haimes, LS Lasdon, and DA Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 1(3):296–297, 1971.

William Haskell, Debarun Kar, Fei Fang, Sam Cheung, Elizabeth Denicola, and Milind Tambe. Robust protection of fisheries with compass. In *Innovative Application of Artificial Intelligence (IAAI)*, 2014.

C.L. Hwang and A.S.M. Masud. *Multiple objective decision making, methods and applications: a state-of-the-art survey*. Springer-Verlag Berlin, Heidelberg, 1979.

A. Inselberg. Parallel coordinates for visualizing multidimensional geometry. *New Techniques and Technologies for Statistics*, pages 279–288, 1997.

H. Iseki, A. Demisch, B.D. Taylor, and A.C. Yoh. *Evaluating the Costs and Benefits of Transit Smart Cards*. California PATH Research Report, Institute of Transportation Studies, University of California at Berkeley, 2008.

Manish Jain, Erim Kardes, Christopher Kiekintveld, Fernando Ordóñez, and Milind Tambe. Security games with arbitrary schedules: A branch and price approach. In *AAAI*, 2010a.

Manish Jain, Jason Tsai, James Pita, Christopher Kiekintveld, Shyamsunder Rathi, Milind Tambe, and Fernando Ordonez. Software Assistants for Randomized Patrol Planning for the LAX Airport Police and the Federal Air Marshals Service. *Interfaces*, 40:267–290, 2010b.

Manish Jain, Kevin Leyton-Brown, and Milind Tambe. The deployment-to-saturation ratio in security games. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

Albert Xin Jiang, Thanh H. Nguyen, Milind Tambe, and Ariel D. Procaccia. Monotonic maximin: A robust stackelberg solution against boundedly rational followers. In *Conference on Decision and Game Theory for Security (GameSec)*, 2013a.

Albert Xin Jiang, Zhengyu Yin, Chao Zhang, Milind Tambe, and Sarit Kraus. Game-theoretic randomization for security patrolling with dynamic execution uncertainty. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2013b.

IT Jolliffe. *Principal Component Analysis*. Springer New York, 2002.

Debarun Kar, Fei Fang, Francesco Delle Fave, Nicole Sintov, and Milind Tambe. a game of thrones: When human behavior models compete in repeated stackelberg security games. In *14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015)*, 2015.

RL Keeney and H Raiffa. *Decisions with multiple objectives: preferences and value tradeoffs*. New York: Wiley, 1976.

Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordonez, and Milind Tambe. Computing optimal randomized resource allocations for massive security games. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 689–696, 2009.

I.Y. Kim and O.L. de Weck. Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. *Structural and Multidisciplinary Optimization*, 29:149–158, 2005. ISSN 1615-147X. URL `http://dx.doi.org/10.1007/s00158-004-0465-1`.

Christopher S Koper. Just enough police presence: Reducing crime and disorderly behavior by optimizing patrol time in crime hot spots. *Justice Quarterly*, 12(4):649–672, 1995.

D. Korzhyk, V. Conitzer, and R. Parr. Complexity of computing optimal Stackelberg strategies in security resource allocation games. In *Proceedings of the 24th AAAI conference on Artificial Intelligence (AAAI)*, pages 805–810, 2010.

D. Korzhyk, V. Conitzer, and R. Parr. Solving stackelberg games with uncertain observability. In *Proceedings of the Tenth International Conference on Agents and Multi-agent Systems (AAMAS)*, pages 1013–1020, 2011a.

D. Korzhyk, V. Conitzer, and R. Parr. Security games with multiple attacker resources. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 273–279, 2011b.

S. Kukkonen and J. Lampinen. Gde3: The third evolution step of generalized differential evolution. In *IEEE Congress on Evolutionary Computation*, volume 1, pages 443–450, 2005.

M. Laumanns, L. Thiele, and E. Zitzler. An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169(3):932–942, 2006.

Sang M Lee, Lori Sharp Franz, and A James Wynne. Optimizing state patrol manpower allocation. *Journal of the Operational Research Society*, pages 885–896, 1979.

J. Letchford and V. Conitzer. Solving security games on graphs via marginal probabilities. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 591–597, 2013.

J. Letchford and Y. Vorobeychik. Optimal interdiction of attack plans. In *Proceedings of the Twelfth International Conference of Autonomous Agents and Multi-agent Systems (AAMAS).*, pages 199–206, 2013.

J. Letchford, L. MacDermed, V. Conitzer, R. Parr, and C. L. Isbell. Computing optimal strategies to commit to in stochastic games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1380–1386, 2012.

D. Li, J.B. Yang, and MP Biswal. Quantitative parametric connections between methods for generating noninferior solutions in multiobjective optimization. *European journal of operational research*, 117(1):84–99, 1999.

M. Lightner and S. Director. Multiple criterion optimization for the design of electronic circuits. *IEEE Transactions on Circuits and Systems*, 28(3):169 – 179, mar 1981. ISSN 0098-4094. doi: 10.1109/TCS.1981.1084969.

A.V. Lotov, V.A. Bushenkov, and G.K. Kamenev. *Interactive decision maps: Approximation and visualization of Pareto frontier*. Springer Netherlands, 2004.

Wenlian Lu, Shouhuai Xu, and Xinlei Yi. Optimizing active cyber defense. In *Decision and Game Theory for Security*, pages 206–225. Springer, 2013.

Duncan R Luce. *Individual Choice Behavior*. Wiley, 1959.

David G. Luenberger. *Optimization by Vector Space Methods*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1997. ISBN 047118117X.

Mariano Luque, Kaisa Miettinen, Petri Eskelinen, and Francisco Ruiz. Incorporating preference information in interactive reference point methods for multiobjective optimization. *Omega*, 37(2):450 – 462, 2009. ISSN 0305-0483. doi: 10.1016/j.omega.2007.06.001. URL `http://www.sciencedirect.com/science/article/pii/S030504830700093X`.

Mohammad Hossein Manshaei, Quanyan Zhu, Tansu Alpcan, Tamer Bacşar, and Jean-Pierre Hubaux. Game theory meets network security and privacy. *ACM Computing Surveys (CSUR)*, 45(3):25, 2013.

G. Mavrotas. Effective implementation of the $\epsilon$-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, 213(2):455–465, 2009.

Richard D McKelvey and Thomas R Palfrey. Quantal response equilibria for normal form games. *Games and economic behavior*, 10(1):6–38, 1995.

Laura A McLay, Adrian J Lee, and Sheldon H Jacobson. Risk-based policies for airport security checkpoint screening. *Transportation science*, 44(3):333–349, 2010.

Kien C Nguyen, Tansu Alpcan, and Tamer Basar. Stochastic games for security in networks with interdependent nodes. In *Game Theory for Networks, 2009. GameNets' 09. International Conference on*, pages 697–703. IEEE, 2009.

Thanh Hong Nguyen, Rong Yang, Amos Azaria, Sarit Kraus, and Milind Tambe. Analyzing the effectiveness of adversary modeling in security games. *In AAAI*, 2013.

Raymond S Nickerson. Confirmation bias: a ubiquitous phenomenon in many guises. *Review of General Psychology*, 2(2):175, 1998.

Thomas C Ormerod and Coral J Dando. Finding a needle in a haystack: Toward a psychologically informed method for aviation security screening. *Journal of Experimental Psychology*, 2014.

Praveen Paruchuri, Milind Tambe, Fernando Ordonez, and Sarit Kraus. Security in multiagent systems by policy randomization. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2006.

Praveen Paruchuri, Jonathan P. Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 895–902, 2008.

Nicola Persico and Petra E. Todd. Passenger profiling, imperfect screening, and airport security. *The American Economic Review*, 95(2), 2005.

Radek Píbil, Viliam Lisỳ, Christopher Kiekintveld, Branislav Bošanskỳ, and Michal Pěchouček. Game theoretic model of strategic honeypot selection in computer networks. In *Decision and Game Theory for Security*, pages 201–220. Springer, 2012.

J. Pita, M. Jain, J. Marecki, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus. Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 125–132, 2008.

James Pita, Manish Jain, Fernando Ordez, Milind Tambe, Sarit Kraus, and Reuma Magori-Cohen. Effective solutions for real-world stackelberg games: When agents must deal with human uncertainties. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2009.

James Pita, Manish Jain, Fernando Ordonez, Milind Tambe, and Sarit Kraus. Robust solutions to stackelberg games: Addressing bounded rationality and limited observations in human cognition. *Artificial Intelligence Journal, 174(15):1142-1171, 2010*, 2010.

James Pita, Milind Tambe, Chris Kiekintveld, Shane Cullen, and Erin Steigerwald. Guards - game theoretic security allocation on a national scale. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2011.

SD Pohekar and M. Ramachandran. Application of multi-criteria decision making to sustainable energy planninga review. *Renewable and Sustainable Energy Reviews*, 8(4):365–381, 2004.

Mark Ritchey and Sean Nicholson-Crotty. Deterrence theory and the implementation of speed limits in the american states. *Policy Studies Journal*, 39(2):329–346, 2011.

Eric Shieh, Bo An, Rong Yang, Milind Tambe, Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrett Meyer. Protect: A deployed game theoretic system to protect the ports of the united states. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 13–20. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

Eric Anyung Shieh, Albert Xin Jiang, Amulya Yadav, Pradeep Varakantham, and Milind Tambe. Unleashing dec-mdps in security games: Enabling effective defender teamwork. In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pages 819–824, 2014.

Herbert A Simon. A behavioral model of rational choice. *The quarterly journal of economics*, pages 99–118, 1955.

SPF. Annual road traffic situation 2012. Technical report, Singapore Police Force, 2013. URL `http://driving-in-singapore.spf.gov.sg/services/driving_in_singapore/documents/20130507_Annual_Road_Traffic_Situation_Stats.pdf`.

R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation, and Application*. Robert E. Krieger Publishing Company, 1989.

Joseph Stiglitz and Andrew Weiss. Sorting out the differences between screening and signaling models,. *Mathematical Models in Economics. Oxford University Press, Oxford*, 1994.

Milind Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 2011.

R.V. Tappeta and J.E. Renaud. Interactive multiobjective optimization procedure. *AIAA Journal*, 37(7):881–889, 1999.

A. Toffolo and A. Lazzaretto. Evolutionary algorithms for multi-objective energetic and economic optimization in thermal system design. *Energy*, 27(6):549–567, 2002.

Jason Tsai, Shyamsunder Rathi, Christopher Kiekintveld, Fernando Ordez, and Milind Tambe. Iris - a tool for strategic security allocation in transportation networks. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2009.

J.J. van Wijk and R. van Liere. Hyperslice: visualization of scalar functions of many variables. In *Visualization*, pages 119–125. IEEE Computer Society, 1993.

Pradeep Varakantham, Hoong Chuin Lau, and Zhi Yuan. Scalable randomized patrolling for securing rapid transit networks. In *IAAI*, pages 1563–1568, 2013.

Bernhard von Stengel and Shmuel Zamir. Leadership with commitment to mixed strategies. Technical Report LSE-CDAM-2004-01, CDAM Research Report, 2004.

Yevgeniy Vorobeychik, Bo An, Milind Tambe, and Satinder Singh. Computing solutions in infinite-horizon discounted adversarial patrolling games. In *Twenty-Fourth International Conference on Automated Planning and Scheduling*, 2014.

Xiaowen Wang, Cen Song, and Jun Zhuang. Simulating a multi-stage screening network: A queueing theory and game theory application. In *Game Theoretic Analysis of Congestion, Safety and Security*, pages 55–80. Springer, 2015.

Rong Yang, Fernando Ordonez, and Milind Tambe. Computing optimal strategy against quantal response in security games. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 847–854. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

Rong Yang, Albert Xin Jiang, Milind Tambe, and Fernando Ordóñez. Scaling-up security games with boundedly rational adversaries: a cutting-plane approach. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 404–410. AAAI Press, 2013.

Rong Yang, Benjamin Ford, Milind Tambe, and Andrew Lemieux. Adaptive resource allocation for wildlife protection against illegal poachers. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2014.

Zhengyu Yin and Milind Tambe. A unified method for handling discrete and continuous uncertainty in bayesian stackelberg games. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012.

Zhengyu Yin, Manish Jain, Milind Tambe, and Fernando Ordonez. Risk-averse strategies for security games with execution and observational uncertainty. In *AAAI*, 2011.

Zhengyu Yin, Albert Jiang, Matthew Johnson, Milind Tambe, Christopher Kiekintveld, Kevin Leyton-Brown, Tuomas Sandholm, and John Sullivan. Trusts: Scheduling randomized patrols for fare inspection in transit systems. In *Conference on Innovative Applications of Artificial Intelligence (IAAI)*, 2012.

L. Zadeh. Optimality and non-scalar-valued performance criteria. *IEEE Transactions on Automatic Control*, 8(1):59–60, 1963.

E. Zitzler, M. Laumanns, and L. Thiele. *SPEA2: Improving the strength Pareto evolutionary algorithm*. TIK-Report 103. Swiss Federal Institute of Technology (ETH) Zurich, Computer Engineering and Networks Engineering (TIK), 2001.