Addressing Uncertainty in Stackelberg Games for Security: Models and Algorithms

by

Zhengyu Yin

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)

March 2013

Acknowledgments

First and foremost, I would like to thank my advisor, Milind Tambe. When I first came abroad to USC to pursue my doctorate degree, little did I know about doing research and writing papers. It was not easy initially, and my lack of language literacy and common sense of living in the U.S. certainly did not make things any better. Awkward moment as I still remember vividly, my first meeting with Milind had to end in five minutes because I could barely communicate. Over my countless times of feeling disappointed, frustrated, or frightened, Milind has always been supportive and encouraging. He has taught me how to do scientific study and kept reminding me to think of the big picture and always push an idea to its limits. His dedication to students, accessibility, ethical standards, insistence of practicality make him the best academic advisor I could ever ask for. It is never enough for me to say in words how great Milind is as a teacher, a role model, and a sincere friend.

I would also like to thank other members of my dissertation committee for providing valuable feedback to my research and pushing me to think about it in another level: Bhaskar Krishnamachari, Rajiv Maheswaran, Mathew McCubbins, Fernando Ordóñez, and Tuomas Sandholm.

Over the years I have been fortunate to collaborate with many great researchers: Bo An, Matthew Brown, Branislav Bosansky, Emma Bowring, Jesus Cerquides, Vincent Conitzer, Francesco Delle Fave, Manish Jain, Michal Jakob, Albert Jiang, Matthew Johnson, Bostjan

Kaluza, David Kempe, Christopher Kiekintveld, Antonin Komenda, Dmytro Korzhyk, Sarit Kraus, Atul Kumar, Jun-young Kwak, Kevin Leyton-Brown, Samantha Luber, Zbynek Moler, Fernando Ordóñez, Michal Pechoucek, Kanna Rajan, Juan Antonio Rodriguez-Aguilar, Tuomas Sandholm, Eric Shieh, John Sullivan, Milind Tambe, Matthew Taylor, Jason Tsai, Ondrej Vanek, Meritxell Vinyals, Rong Yang, Rob Zinkov.

I also want to thank my friends at USC and entire TEAMCORE community. Special thanks to Manish Jain for helping me through many emergencies, Jason Tsai for correcting my language errors and proofreading my papers for countless times, Matthew Brown for reigniting my excitement on international soccer leagues, Bo Yang and Xin Fang for constantly treating me with delicious food. Thank you all for making the five years of my Ph.D an enjoyable and memorable experience.

Finally, I want to thank my family for their unconditional love and I always feel sad that I have to live far away from my parents. I would like to thank my wife Olivia for always being supportive and taking good care of me when I am busy with work. Olivia, thank you for making our live so cheerful and enjoyable.

Table of Contents

List of Figures

Abstract

Recently, there has been significant research interest in using game-theoretic approaches to allocate limited security resources to protect physical infrastructure including ports, airports, transit systems, and other critical national infrastructure as well as natural resources such as forests, tigers, fish, and so on. Indeed, the leader-follower Stackelberg game model is at the heart of many deployed applications. In these applications, the game model provides a randomized strategy for the leader (security forces), under the assumption that the adversary will conduct surveillance before launching an attack. Inevitably, the security forces are faced with the problem of uncertainty. For example, a security officer may be forced to execute a different patrol strategy from the planned one due to unexpected events. Also, there may be significant uncertainty regarding the amount of surveillance conducted by an adversary. While Bayesian Stackelberg games for modeling discrete uncertainty have been successfully used in deployed applications, they are NP-hard problems and existing methods perform poorly in scaling up the number of types: inadequate for complex real world problems. Furthermore, Bayesian Stackelberg games have not been applied to model execution and observation uncertainty and finally, they require the availability of full distributional information of the uncertainty.

To overcome these difficulties, my thesis presents four major contributions. First, I provide a novel algorithm Hunter for Bayesian Stackelberg games to scale up the number of types. Exploiting the efficiency of Hunter, I show preference, execution and observation uncertainty can be addressed in a unified framework. Second, to address execution and observation uncertainty (where distribution may be difficult to estimate), I provide a robust optimization formulation to compute the optimal risk-averse leader strategy in Stackelberg games. Third, addressing the uncertainty of the adversary's capability of conducting surveillance, I show that for a class of Stackelberg games motivated by real security applications, the leader is always best-responding with a Stackelberg equilibrium strategy regardless of whether the adversary conducts surveillance or not. As the final contribution, I provide TRUSTS, a novel game-theoretic formulation for scheduling randomized patrols in public transit domains where timing is a crucial component. TRUSTS addresses dynamic execution uncertainty in such spatiotemporal domains by integrating Markov Decision Processes into the game-theoretic model. Simulation results as well as real-world trials of TRUSTS in the Los Angeles Metro Rail system provide validations of my approach.

Chapter 1: Introduction

My thesis focuses on game-theoretic approaches to allocate resources to protect critical infrastructure in a variety of security settings. While there is a diverse set of security scenarios, the typical problem among them is that the security agencies have to protect a large set of targets with limited resources, making it impossible to protect all targets at all times. For instance, the security agencies are responsible for protecting large transportation networks such as ports, train stations, and airports, from potential terrorist activities that may cause significant damage. The security agencies are also required to patrol an area or a network to deter crimes or misdemeanors such as illegal extractions of forest resources or fare evasion in public transit systems. Since deterministic allocations of security resources can often be exploited by intelligent adversary who conducts surveillance before an act, it is often more desirable for the security agencies to allocate their resources in a randomized fashion.

1.1 Problem Addressed

Game theory provides a formal mathematical framework for reasoning about the aforementioned resource randomization problems. Indeed, game-theoretic approaches have been used in multiple deployed applications, including ARMOR for randomizing checkpoints and canine units at the Los Angeles International Airport (LAX) Pita et al. [2008], IRIS for randomizing Federal Air Marshals on commercial flights Tsai et al. [2009], PROTECT for randomizing port security patrols at the Boston Coast Guard Shieh et al. [2012], and TRUSTS for randomizing ticket inspections in the Los Angeles Metro Rail System Yin et al. [2012a] (under evaluation as of March 2013). At the backbone of these applications is the leader-follower Stackelberg game model, where the leader (security agency) acts first by committing to a mixed strategy, and the follower (adversary) best-responds after observing the leader's mixed strategy perfectly. Beyond those deployed security applications, the Stackelberg game model has been studied in numerous other security problems ranging from patrolling in adversarial domains Agmon et al. [2008]; Gatti [2008]; Vanek et al. [2011] to natural resource protection Johnson et al. [2012] and computer network security Vanek et al. [2012a].

The Stackelberg game model, despite its recent success in real world deployments, presumes both perfect knowledge about the adversary and perfect execution of the planned security activities. It also assumes the adversary can perfectly observe the mixed strategy of the security agency, i.e., a probability distribution over security activities. Nevertheless, in real world security domains, the security agencies are inevitably faced with various types of uncertainty. Game-theoretic approaches neglecting these types of uncertainty may lead to a significant loss in real world deployments.

My thesis studies three main causes of uncertainty typically found in security applications. First the security agencies may have incomplete information about the adversaries. Adversaries can have distinct objectives and capabilities, leading to varying preferences over different targets. For example, the police at LAX may be facing either a well-funded hard-lined terrorist or criminals from local gangs; and the LA Metro system has tens of thousands of potential fare evaders daily, each of whom may have a distinct intended trip and risk profile. Second, the adversary's observation is most likely imperfect. A deliberate terrorist may get noisy observations when conducting surveillance: he may occasionally not observe an officer patrolling a target, or mistake a passing car as a security patrol. In some situations, the adversary may act without acquiring information about the security strategy, especially when the cost of surveillance (such as monetary expenses and risk of being caught) is prohibitively high or the security measures are difficult to observe (e.g., undercover officers). In these situations, the information that an adversary can acquire through surveillance is either too limited or too noisy to be an important factor in his decision making. Finally, the security agencies may not be able to execute their strategies perfectly. Planned security activities may be interrupted or canceled due to emergency events. For example, a canine unit protecting a terminal at LAX may be urgently called off to another assignment or alternatively a new unit could become available. A patrol schedule of an LA Metro officer may get interrupted due to a variety of reasons such as writing citations, handling emergencies, or felony arrests.

Earlier research on modeling uncertainty in Stackelberg games has primarily focused on the Bayesian extension of Stackelberg games which represents the discrete preference uncertainty using multiple adversary types. Unfortunately, solving Bayesian Stackelberg games is NP-hard Conitzer and Sandholm [2006], with existing methods Conitzer and Sandholm [2006]; Paruchuri et al. [2008]; Jain et al. [2011b] performing poorly in scaling up the number of types: they are inadequate for complex real world problems. The second drawback of Bayesian Stackelberg game model is that it requires full distributional information of the uncertainty which may not always be available. Finally, the Bayesian Stackelberg game model has not been (and in certain situations cannot be) applied to problems where there is uncertainty in the follower's observation and the leader's execution.

Thus, there are four problems to be addressed: The first is to design new efficient and scalable algorithms for Bayesian Stackelberg games. The second is to design models and algorithms to compute robust solutions when the uncertainty distribution is unavailable. The third is to address the follower's observation uncertainty, including the uncertainty in his capability of observing the leader's strategy. The fourth is to address the leader's execution uncertainty, and in particular for time-critical domains where execution errors can affect the leader's ability to carry out the planned schedules in later time steps.

1.2 Contributions

In this context my thesis provides the following four major contributions. The first contribution of my thesis is a new unified method for solving Bayesian Stackelberg games with both discrete and continuous uncertainty Yin and Tambe [2012]. At the core of this unified method is a new algorithm for solving discrete finite Bayesian Stackelberg games, called HUNTER (**H**andling **UN**cerTainty **E**fficiently using **R**elaxation). HUNTER combines the following key ideas:

- efficient pruning via a best-first search in the follower's strategy space;
- a novel linear program for computing tight upper bounds for this search;
- using Bender's decomposition for solving the upper bound linear program efficiently;
- efficient inheritance of Bender's cuts from parent to child;
- an efficient heuristic branching rule.

Then I show that *sample average approximation* approach can be applied together with HUNTER to address preference, execution, and observation uncertainty in both discrete and continuous forms in a unified framework. Furthermore, my experimental results suggest that HUNTER provides orders of magnitude speedups over the best existing methods for Bayesian Stackelberg games Conitzer and Sandholm [2006]; Paruchuri et al. [2008]; Jain et al. [2011b]. The efficiency of HUNTER can be further exploited in the sample average approximation approach to solving problems with both discrete and continuous uncertainty.

My second contribution is a robust optimization framework, called RECON (**R**isk-averse **E**xecution **C**onsidering **O**bservational **N**oise), to address execution and observation uncertainty of unknown distribution, with a focus on security problems motivated by the ARMOR application Yin et al. [2011]. RECON addresses the major drawback of the Bayesian model: the necessity of knowing the precise distribution of the uncertainty, and is particularly useful for security scenarios where no good estimation of such uncertainty distribution is available. For example, the distribution of the follower's observation noise is often difficult to measure statistically due to limited data. RECON models the uncertainty boundary as a hyper-rectangle, and correspondingly computes the optimal risk-averse strategy for the leader. In particular, RECON assumes that nature chooses an uncertainty realization within the given hyper-rectangle to maximally reduce the leader's utility, and maximizes against this worst case. This robust optimization formulation is similar in spirit to Aghassi and Bertsimas [2006a]; the latter, however, is in the context of simultaneous move games. To solve the RECON formulation efficiently, I provide a mixed integer linear program (MILP) and two novel heuristics that speed up the computation of MILP by orders of magnitude. I provide experimental analysis comparing the performance of various security game strategies including those generated by Recon and Hunter in simulated uncertainty settings, showing the value of Recon and Hunter under different assumptions.

The third contribution of my thesis studies security problems where the adversary may or may not conduct surveillance before taking an action Yin et al. [2010]. The assumption that the adversary always observes the leader's strategy (perfectly or imperfectly) is fundamental in both the Stackelberg game model as well as the Recon model. However when the adversary acts without surveillance, a simultaneous-move game model may be a better reflection of the real situation. The leader faces an unclear choice about which strategy to adopt: the recommendation of the Stackelberg model, or of the simultaneous-move model, or something else entirely? My thesis provides theoretical and experimental analysis of the leader's dilemma, focusing on security games, a class of Stackelberg games motivated by the ARMOR and IRIS applications. In particular, I show that in security games that satisfy the SSAS (Subsets of Schedules Are Schedules) property (such as ARMOR games), any Stackelberg game equilibrium strategy for the leader is also a Nash equilibrium strategy. The leader is therefore best-responding with a Stackelberg equilibrium strategy regardless of the follower's ability to observe the leader's strategy, resolving the leader's dilemma. On the other hand, counter-examples to this (partial) equivalence between leader's Stackelberg and Nash equilibrium strategies exist when the *SSAS* property does not hold. However, my experiments show that in this case, the fraction of games where the Stackelberg equilibrium strategy is not in any Nash equilibrium is vanishingly small with increasing problem sizes. In practical terms, my theoretical and experimental contributions imply that security agencies in applications such as ARMOR (where games satisfy the *SSAS* property) and IRIS (where games have small schedule size and a large number of schedules) can simply stick to the Stackelberg game model regardless of the follower's ability to observe the leader's mixed strategy.

The final contribution of this thesis addresses dynamic execution uncertainty in security patrolling for public transit systems. This problem is significantly more complex than earlier problems such as ARMOR and IRIS where security activities are represented as a single action. In transit domains, security activities are patrols within the transit systems, carried out as sequences of actions in different place and time. Execution uncertainty in such spatiotemporal domains has vastly different impact since an execution error can affect the security officers' ability to carry out their planned schedules in later time steps. The result of the investigation is a new game-theoretic model, called TRUSTS (**T**actical **R**andomization for **U**rban **S**ecurity in **T**ransit **S**ystems) Yin et al. [2012a,b]; Jiang et al. [2013]. TRUSTS proposed in my thesis features the following four key ideas:

- I provide a general Bayesian Stackelberg game model for spatiotemporal patrolling with execution uncertainty where the execution uncertainty is represented as Markov Decision Processes.
- I show that when the utility functions have a certain separable structure, the leader's strategy space can be compactly represented. As a result the problem can be reduced to a polynomial-sized optimization problem, solvable by existing approaches for Bayesian Stackelberg games without execution uncertainty.
- TRUSTS employs a novel history duplicate approach to encode constraints on feasible patrols within this compact representation.

- The compactly represented solutions are stochastic patrol policies that can be used to generate randomized patrol schedules with contingency plans. Such contingency plans can be implemented as a smart-phone app carried by patrol units, or as a communication protocol with a central operator.

As an empirical validation of the approach, I apply the game-theoretic model to the problem of fare evasion deterrence in the Los Angeles Metro Rail system, providing details of model creation, simulation results, and smart-phone app design for implementing the patrol policies generated.

1.3 Overview of Thesis

This thesis is organized in the following way. Chapter 2 introduces necessary background for the research presented in this thesis. Chapter 3 presents the algorithm Hunter for Bayesian Stackelberg games, its extension to address preference, execution, and observation uncertainty in a unified framework, and the corresponding experimental results. Chapter 4 presents the robust optimization framework Recon and the corresponding experimental results. Chapter 5 studies the uncertainty of whether the adversary conducts surveillance or not, establishing connection between the Stackelberg equilibrium and the Nash equilibrium in security games. Chapter 6 presents the TRUSTS system, describing the model framework, strategy representation, execution uncertainty model using Markov Decision Processes, and experimental results from computer simulations as well as field trials. Chapter 7 presents related work. And finally, Chapter 8 concludes the thesis and presents issues for future work.

Chapter 2: Background

This chapter begins by introducing motivating examples of real world security applications in Section 2.1. It then provides background on the general Stackelberg game model and its Bayesian extension in Section 2.2 and 2.3. Section 2.4 introduces the standard solution concept known as the *Strong Stackelberg Equilibrium* (SSE) and Section 2.5 describes previous algorithms for finding SSE in general Bayesian Stackelberg games. Finally, in Section 2.6, I introduce a restricted class of Stackelberg games called *security games* motivated by two security applications: ARMOR for the Los Angeles International Airport (LAX) and IRIS for the Federal Air Marshals Services (FAMS).

2.1 Motivating Applications

While there are many potential security applications where game theory is applicable, e.g., protecting ports, road network, forest, fish, etc., in this section, I will emphasize three real world security applications that are closely related to this thesis. The first is the ARMOR security system deployed at the Los Angeles International Airport (LAX) Pita et al. [2008]. In this domain police are able to set up checkpoints on roads leading to particular terminals, and assign canine units (bomb-sniffing dogs) to patrol terminals. Police resources in this domain are homogeneous, and do not have significant scheduling constraints.

IRIS is a similar application deployed by the Federal Air Marshals Service (FAMS) Tsai et al. [2009]. Armed marshals are assigned to commercial flights to deter and defeat terrorist attacks. This domain has more complex constraints. In particular, marshals are assigned to tours of flights that return to the same destination, and the tours on which any given marshal is available to fly are limited by the marshal's current location and timing constraints. The types of scheduling and resource constraints considered in this thesis (in particular Chapter 5) are motivated by those necessary to represent this domain.

The third example is the TRUSTS application for the Los Angeles Metro Rail system focusing on fare evasion deterrence. In the Los Angeles Metro Rail system (and other proof-of-payment transit systems worldwide), passengers are legally required to buy tickets before boarding, but there are no gates or turnstiles. There are, quite literally, no barriers to entry, as illustrated in Figure 2.1. Instead, security personnel are dynamically deployed throughout the transit system, randomly inspecting passenger tickets; fare evaders face significant penalties when caught. This proof-of-payment fare collection method is typically chosen as a more cost-effective alternative to direct fare collection, i.e., when the revenue lost to fare evasion is believed to be less than what it would cost to directly preclude it. (See `http://en.wikipedia.org/wiki/Proof-of-payment` for a list of such systems.)

For the Los Angeles Metro system, with approximately 300,000 riders daily, this revenue loss can be significant; the annual cost has been estimated at $5.6 million Booz Allen Hamilton [2007]. The Los Angeles Sheriffs Department (LASD) deploys uniformed patrols on board trains and at stations for fare-checking (and for other purposes such as crime prevention), in order to

Figure 2.1: Entrance of an LA Metro Rail station.

discourage fare evasion. With limited resources to devote to patrols, it is impossible to cover all locations at all times. The LASD thus requires some mechanism for choosing times and locations for inspections. Any predictable patterns in such a patrol schedule are likely to be observed and exploited by potential fare-evaders. The traditional approach relies on humans for scheduling the patrols. However, human schedulers are poor at generating unpredictable schedules Wagenaar [1972]; Tambe [2011]; furthermore such scheduling for LASD is a tremendous cognitive burden on the human schedulers who must take into account all of the scheduling complexities (e.g., train timings, switching time between trains, and schedule lengths). Indeed, the sheer difficulty of even enumerating the trillions of potential patrols makes any simple automated approach—such as a simple dice roll—inapplicable.

2.2 Stackelberg Games

A Stackelberg game is a two-person game played by a leader and a follower von Stackelberg [1934], where the leader commits to a mixed strategy first, and the follower observes the leader's strategy and responds with a pure strategy, maximizing his utility correspondingly. Since significant portion of this thesis focuses on Stackelberg games for security applications where the leader defends a set of physical assets against potential attacks, the terms "defender" and "leader", and the terms "attacker" and "follower" are used interchangeably respectively. For explanatory purpose, I will also refer to the leader (defender) as "her" and the follower (attacker) as "him".

The leader in Stackelberg games benefits from the power of commitment known as the first mover's advantage in game theory literature. To see the advantage of being a leader, consider a simple game in normal form given below. If the players move simultaneously, the only Nash Equilibrium (NE) of this game is for the row player to play a and the column player c, giving the row player a utility of 2. This can be seen by noticing that b is strictly dominated for the row player. On the other hand, if the row player moves first, she can commit to b. With the column player best responds with d, the row player can receive a utility of 3, better than the simultaneous-move case. In fact, the Stackelberg equilibrium strategy is for the row player to play a with .5 and b with .5, so that the best response for the column player is to play d, which gives the row player an expected utility of 3.5.[2]

	c	d
a	2,1	4,0
b	1,0	3,1

Figure 2.2: Example of a Stackelberg game

In the general form of Stackelberg games, the leader's mixed strategy is an N-dimensional real vector $\mathbf{x} \in \mathbb{R}^N$ subject to a set of linear constraints (e.g., $A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}$). This mixed strategy representation generalizes the traditional mixed strategy concept in game theory where $\sum_{i=1}^{N} x_i = 1$ with x_i representing the probability of playing pure strategy i. The added expressiveness of

[2]In these games it is assumed that if the follower is indifferent, he breaks the tie in the leader's favor (otherwise, the optimal solution is not well defined).

this generalization is useful for compact strategy representation in many security domains, e.g., TRUSTS as we will see in Section 6.1.2.

The leader and follower's expected utilities are both linear combinations of $\mathbf{x}$ with weights dependent on the follower's choice. Given a leader's strategy $\mathbf{x}$, the follower maximizes his expected utility by choosing one of his J pure strategies. For each pure strategy j played by the follower, the leader gets a utility of $\boldsymbol{\mu}_j^{\mathrm{T}}\mathbf{x} + \mu_{j,0}$ and the follower gets a utility of $\boldsymbol{\nu}_j^{\mathrm{T}}\mathbf{x} + \nu_{j,0}$, where $\boldsymbol{\mu}_j, \boldsymbol{\nu}_j$ are real vectors in $\mathbb{R}^N$ and $\mu_{j,0}, \nu_{j,0} \in \mathbb{R}$. It is useful to define the leader's utility matrix U and the follower's utility matrix V as the following,

$$U = \begin{pmatrix} \mu_{1,0} & \dots & \mu_{J,0} \\ \boldsymbol{\mu}_1 & \dots & \boldsymbol{\mu}_J \end{pmatrix}, V = \begin{pmatrix} \nu_{1,0} & \dots & \nu_{J,0} \\ \boldsymbol{\nu}_1 & \dots & \boldsymbol{\nu}_J \end{pmatrix}.$$

Then for a leader's strategy $\mathbf{x}$, the leader and follower's J utilities for the follower's J pure strategies are $U^{\mathrm{T}}\begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}$ and $V^{\mathrm{T}}\begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix}$.

2.3 Bayesian Stackelberg Games

A Bayesian extension to the Stackelberg game allows multiple types of followers, each with its own payoff matrix. This extension is useful in modeling the diversity of potential adversaries in all aspects. For example, the police at LAX may be facing either a well-funded hard-lined terrorist or criminals from local gangs; and the LA Metro system has tens of thousands of potential fare evaders daily, each of whom may have a distinct intended trip and risk profile.

Formally, a Bayesian Stackelberg game is a Stackelberg game between a leader and a follower whose type is drawn randomly from a set of follower types $\{1, 2, \dots, \Lambda\}$. Each type $1 \leq \lambda \leq \Lambda$

is associated with a prior probability p^λ representing the likelihood of its occurrence and a pair of utility matrices (U^λ, V^λ) for the leader and the follower respectively. The leader commits to a mixed strategy knowing the prior distribution of all different follower types but not the type of the follower she faces. The follower, however, knows his own type λ, and plays the best response according to his utility matrix V^λ. For the purpose of equilibrium computation, it is sufficient to consider only pure strategy responses of the follower as shown in Conitzer and Sandholm [2006].

The expected utilities of both players are well-defined for a pair of leader's mixed strategy $\mathbf{x}$ and a vector of the follower's pure responses $\mathbf{j} = (j^1, \ldots, j^\Lambda)$ where j^λ denotes the pure strategy of follower type λ. For the follower of type λ, his expected utility is $v^\lambda(\mathbf{x}, j^\lambda) = (\boldsymbol{\nu}^\lambda_{j^\lambda})^{\mathrm{T}}\mathbf{x} + \nu_{j^\lambda,0}$. For the leader, her expected utility is $u(\mathbf{x}, \mathbf{j}) = \sum_{\lambda=1}^{\Lambda} p^\lambda u^\lambda(\mathbf{x}, j^\lambda)$ where $u^\lambda(\mathbf{x}, j^\lambda) = (\boldsymbol{\mu}^\lambda_{j^\lambda})^{\mathrm{T}}\mathbf{x} + \mu_{j^\lambda,0}$ is the leader's expected utility against follower type λ.

As an example, which we will return to in Chapter 3, consider a Bayesian Stackelberg game with two follower types, where type 1 appears with probability .84 and type 2 appears with probability .16. The leader (defender) chooses a probability distribution of allocating one resource to protect the two targets whereas the follower (attacker) chooses the best target to attack. We show the payoff matrices in Figure 2.3, where the leader is the row player and the follower is the column player. The utilities of the two types are identical except that a follower of type 2 gets a utility of 1 for attacking *Target2* successfully, whereas one of type 1 gets 0. The leader's strategy is a column vector $(x_1, x_2)^{\mathrm{T}}$ representing the probabilities of protecting the two targets. Given

Type 1	*Target1*	*Target2*
Target1	1, -1	-1, 0
Target2	0, 1	1, -1

Type 2	*Target1*	*Target2*
Target1	1, -1	-1, 1
Target2	0, 1	1, -1

Figure 2.3: Payoff matrices of a Bayesian Stackelberg game.

one resource, the strategy space of the leader is $x_1 + x_2 \leq 1, x_1 \geq 0, x_2 \geq 0$, i.e., $A = (1, 1), \mathbf{b} = 1$. The payoffs in Figure 2.3 can be represented by the following utility matrices,

$$U^1 = \begin{pmatrix} 0 & 0 \\ 1 & -1 \\ 0 & 1 \end{pmatrix}, V^1 = \begin{pmatrix} 0 & 0 \\ -1 & 0 \\ 1 & -1 \end{pmatrix}; U^2 = \begin{pmatrix} 0 & 0 \\ 1 & -1 \\ 0 & 1 \end{pmatrix}, V^2 = \begin{pmatrix} 0 & 0 \\ -1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Suppose the leader commits to a mixed strategy $(x_1, x_2)^{\mathrm{T}}$ while follower type 1 attacks *Target1* and type 2 attacks *Target2*. Follower type 1 gets an expected utility of $-x_1 + x_2$ and follower type 2 gets an expected utility of $x_1 - x_2$. On the other hand, the leader's expected utility is $0.84(1 \cdot x_1 + 0 \cdot x_2) + 0.16((-1) \cdot x_1 + 1 \cdot x_2)$.

2.4 Strong Stackelberg Equilibrium

Two types of unique Stackelberg equilibria were proposed in Leitmann [1978], which are typically called "strong" and "weak" after Breton et al. [1988]. The two types both assume the follower best responds to the leader's mixed strategy. But in cases where ties exist, i.e., multiple follower pure strategies yield the same maximum expected utilities for the follower, the strong form assumes that the follower will always choose the optimal strategy for the leader while the weak form assumes that the follower will choose the worst strategy for the leader. A strong Stackelberg equilibrium always exists, but a weak Stackelberg equilibrium may not Basar and Olsder [1995]. In addition, the leader can often induce the favorable strong equilibrium by selecting a strategy arbitrarily close to the equilibrium that causes the follower to strictly prefer the desired strategy von Stengel and Zamir [2004].

The *Strong Stackelberg Equilibrium* (SSE) is adopted in recent works of utilizing the Stackelberg game model for security resource randomization Paruchuri et al. [2008]; Kiekintveld et al. [2009]. In Bayesian Stackelberg games, the follower's strategy specifies the pure strategy of each follower type given the leader's mixed strategy $\mathbf{x}$, i.e., a vector of functions $\mathbf{g} = (g^1, \dots, g^\Lambda)$, where each g^λ maps a leader's mixed strategy to a pure strategy of follower type λ. Let $\mathbf{g}(\mathbf{x})$ be the vector of the follower's responses to $\mathbf{x}$ according to $\mathbf{g}$, i.e., $\mathbf{g}(\mathbf{x}) = (g^1(\mathbf{x}), \dots, g^\Lambda(\mathbf{x}))$. Formally a Strong Stackelberg Equilibrium is defined below:

Definition 1. *For a given Bayesian Stackelberg game with utility matrices* $(U^1, V^1), \dots, (U^\Lambda, V^\Lambda)$ *and type distribution* $\mathbf{p}$*, a pair of strategies* $(\mathbf{x}, \mathbf{g})$ *forms a Strong Stackelberg Equilibrium if and only if:*

1. *The leader plays a best response:*

   $u(\mathbf{x}, \mathbf{g}(\mathbf{x})) \geq u(\mathbf{x}', \mathbf{g}(\mathbf{x}')), \forall \mathbf{x}'.$

2. *The follower plays a best response:*

   $v^\lambda(\mathbf{x}, g^\lambda(\mathbf{x})) \geq v^\lambda(\mathbf{x}, j), \forall 1 \leq \lambda \leq \Lambda, \forall 1 \leq j \leq J.$

3. *The follower breaks ties in favor of the leader:*

   $u^\lambda(\mathbf{x}, g^\lambda(\mathbf{x})) \geq u^\lambda(\mathbf{x}, j), \forall 1 \leq \lambda \leq \Lambda, \forall j$ *that is a best response to* $\mathbf{x}$ *as above.*

2.5 Baseline Solvers

2.5.1 Multiple Linear Programs

The leader's strategy in the SSE is considered the optimal leader's strategy as it maximizes the leader's expected utility assuming the follower best responds. This section explains the baseline algorithms for finding the optimal leader's strategy of a Bayesian Stackelberg game. As shown in Conitzer and Sandholm [2006], the problem of computing the optimal leader's strategy $\mathbf{x}^*$ is equivalent to finding a leader's mixed strategy $\mathbf{x}$ and a follower's pure strategy response $\mathbf{j} = \mathbf{g}(\mathbf{x})$ such that the three SSE conditions are satisfied. Mathematically $\mathbf{x}^*$ can be found by solving the following maximization problem:

$$(\mathbf{x}^*, \mathbf{j}^*) = \arg\max_{\mathbf{x},\mathbf{j}}\{u(\mathbf{x}, \mathbf{j}) | v^\lambda(\mathbf{x}, j^\lambda) \geq v^\lambda(\mathbf{x}, j'), \forall 1 \leq j' \leq J\}. \tag{2.1}$$

Equation (2.1) suggests the multiple linear program (LP) approach for finding $\mathbf{x}^*$ as given in Conitzer and Sandholm [2006]. The idea is to enumerate all possible pure strategy responses of the follower $\mathbf{j} \in \{1, \ldots, J\}^\Lambda$. And for each $\mathbf{j}$, the optimal mixed strategy of the leader $\mathbf{x}^*(\mathbf{j})$ such that $\mathbf{j}$ is a best response of the follower can be found by solving the following LP:[1]

$$\begin{aligned} \max_{\mathbf{x}} \quad & u(\mathbf{x}, \mathbf{j}) \\ s.t. \quad & A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0} \\ & v^\lambda(\mathbf{x}, j^\lambda) \geq v^\lambda(\mathbf{x}, j'), \quad \forall 1 \leq \lambda \leq \Lambda, \forall 1 \leq j' \leq J \end{aligned} \tag{2.2}$$

Some of the LPs may be infeasible but it can be shown that at least one LP will return a feasible solution. The optimal leader's strategy $\mathbf{x}^*$ is then the optimal solution of the LP which has the highest objective value (i.e., the leader's expected utility) among all feasible LPs.

[1]Note the formulation here is slightly different from and has fewer constraints in each LP than the original multiple LPs approach in Conitzer and Sandholm [2006] where a Bayesian game is transformed to a normal-form one using Harsanyi transformation Harsanyi [1967].

2.5.2 Dobss: Mixed-Integer Linear Program

Since the followers of different types are mutually independent of each other, there can be at most J^Λ possible combinations of follower best response actions over the Λ follower types. The multiple LPs approach will then have to solve J^Λ LPs and therefore its runtime complexity grows exponentially in the number of follower types. In fact, the problem of finding the optimal strategy for the leader in a Bayesian Stackelberg game with multiple follower types is NP-hard Conitzer and Sandholm [2006]. Nevertheless, researchers have continued to provide practical improvements. Dobss is an efficient general Stackelberg solver Paruchuri et al. [2008] and is in use for security scheduling at the Los Angeles International Airport Pita et al. [2008]. Dobss obtains a decomposition scheme by exploiting the property that follower types are independent of each other and solves the entire problem as one mixed-integer linear program (MILP):

$$\begin{aligned}
\max_{\mathbf{x},\mathbf{u},\mathbf{v},\mathbf{q}^1,\dots,\mathbf{q}^\Lambda} \quad & \textstyle\sum_{\lambda=1}^{\Lambda} p^\lambda u^\lambda \\
s.t. \quad & A\mathbf{x} \le \mathbf{b}, \mathbf{x} \ge \mathbf{0} \\
& \textstyle\sum_{j=1}^{J} q_j^\lambda = 1, \quad \forall \lambda \\
& q_j^\lambda \in \{0,1\}, \quad \forall \lambda, \forall j \\
& u^\lambda \le u^\lambda(\mathbf{x}, j) + (1 - q_j^\lambda) \cdot M, \quad \forall \lambda, \forall j \\
& 0 \le v^\lambda - v^\lambda(\mathbf{x}, j) \le (1 - q_j^\lambda) \cdot M, \quad \forall \lambda, \forall j
\end{aligned} \tag{2.3}$$

Dobss effectively reduces the problem of solving an exponential number of LPs to a compactly represented MILP which can be solved much more efficiently via modern techniques in operation research. The key idea of the Dobss MILP is to represent the pure strategy of each follower type λ as a binary vector $\mathbf{q}^\lambda = (q_1^\lambda, \dots, q_J^\lambda)$. In particular, the binary variable q_j^λ is 1 if the follower type λ chooses the pure strategy j and 0 otherwise. It is easy to see $\sum_{j=1}^{J} q_j^\lambda = 1$ since only one q_j^λ can be 1. M is (conceptually) an infinitely large constant. Variable u^λ represents the leader's expected utility against type λ, which is equal to $u^\lambda(\mathbf{x}, j)$ when the follower chooses j (i.e., $q_j^\lambda = 1$). Variable v^λ represents the expected utility of follower type λ, which is the maximum of $v^\lambda(\mathbf{x}, j)$ over all possible $1 \leq j \leq J$.

2.5.3 HBGS: Branch-and-Bound Search

In addition to multiple LPs and Dobss, recent work (HBGS) solves the problem via a branch-and-bound tree search Jain et al. [2011b]. In contrast to the branch-and-bound techniques typically used in integer programming where branches are created by considering each side of a separating hyperplane, HBGS uses the knowledge of the problem and creates the search tree by assigning one follower type to one pure strategy at each tree level. For example, Figure 2.4 shows the search tree of the example game in Figure 2.3. Each leaf node corresponds to one LP in the multiple LPs approach. For instance, the corresponding linear program of the leftmost leaf node finds the optimal leader strategy such that both type 1 and type 2 have a best response of attacking *Target1*. The multiple LPs approach will solve and compare across all leaf nodes to find the overall optimal strategy of the leader. In this case, the leaf node where type 1 is assigned to *Target1* and type 2 to *Target2* provides the overall optimal strategy.

Instead of solving an LP for all J^Λ leaf nodes, branch-and-bound techniques can be used to speed up the tree search. The key to efficiency in branch-and-bound is obtaining tight upper and lower bounds for internal nodes, i.e., for nodes shown by circles in Figure 2.4, where subsets of follower types are assigned to particular targets. For example, in Figure 2.4, suppose the left subtree has been explored; now if at the rightmost internal node (where type 1 is assigned to *Target2*) we realize that the upper bound on solution quality is 0.5, we could prune the right subtree without even considering type 2. One possible way of obtaining upper bounds is by relaxing the integrality constraints in Dobss MILP. Unfortunately, when the integer variables in DOBSS are relaxed, the objective can be arbitrarily large, leading to meaningless upper bounds. HBGS Jain et al. [2011b] computes upper bounds by heuristically utilizing the solutions of smaller restricted games. However, the preprocessing involved in solving many small games can be expensive and the bounds computed using heuristics can again be loose. In my thesis, a new framework of computing upper and lower bounds will be presented in Chapter 3 which leads to several orders of magnitudes speedup over both Dobss and HBGS.

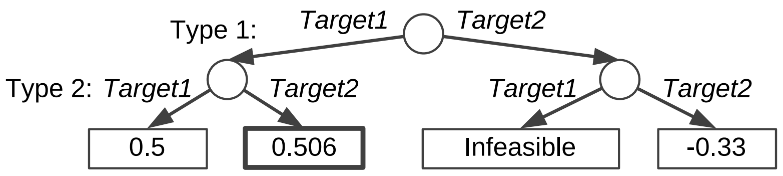


Figure 2.4: Example search tree of solving Bayesian games.

2.6 Security Games

The *security games* definition in this thesis is quite general, but with assumptions motivated by two real-world applications ARMOR and IRIS (see Section 2.1). A *security game* Kiekintveld et al. [2009] is a two-player game between a defender (leader) and an attacker (follower). The attacker may choose to attack any target from a set of N targets: $T = \{t_1, t_2, \ldots, t_N\}$. The defender tries to prevent attacks by covering targets using resources from a set of γ resources.As shown in Figure 2.5, μ_i^c is the defender's utility if t_i is attacked while t_i is covered by some defender resource. If t_i is not covered, the defender gets μ_i^u. The attacker's utility is denoted similarly by ν_i^c and ν_i^u. $\Delta\mu_i = \mu_i^c - \mu_i^u$ denotes the difference between the defender's covered and uncovered utilities. Similarly, $\Delta\nu_i = \nu_i^u - \nu_i^c$. As a key property of security games, we assume $\Delta\mu_i > 0$ and $\Delta\nu_i > 0$. In words, adding resources to cover a target helps the defender and hurts the attacker.

For ease of memorization, the notation here uses μ to denote utility for the leader (defender) and ν to denote the utility for the follower (attacker) similar to that defined for general Stackelberg game introduced in Section 2.2. However, with no explicit definition of utility vectors here, the expected utility calculation in Section 2.2 is not applicable here; instead the expected utilities of the two players for a certain strategy profile are computed in a more compact way as I will explain later (given in (2.4) and (2.5)).

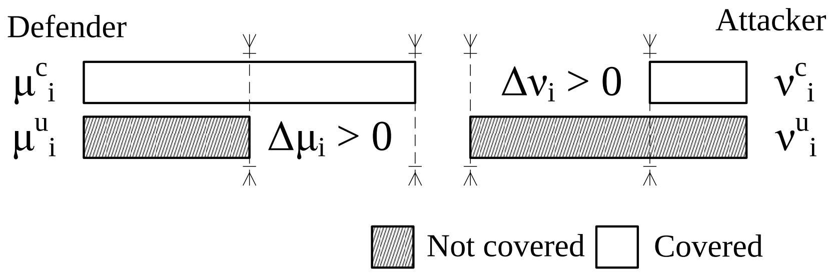


Figure 2.5: Payoff structure of security games.

Motivated by the IRIS application and similar real-world domains, I introduce resource and scheduling constraints for the defender. Resources may be assigned to *schedules* covering multiple targets, $s \subseteq T$. For each resource, there is a subset of the schedules that the resource can potentially cover. In the IRIS application, flights are targets and air marshals are resources. Schedules capture the idea that air marshals fly tours, and must return to a particular starting point. Heterogeneous resources can express additional timing and location constraints that limit the tours on which any particular marshal can be assigned to fly. The IRIS application is an important subset of security games with heterogenous resources where the minimum size of feasible schedules is 2 since an air marshal needs to cover at least a pair of departing and returning flights. The ARMOR application is also an important subclass of security games, with schedules of size 1 and homogeneous resources. In my thesis, the security games for the ARMOR and IRIS application are referred to as ARMOR *games* and IRIS *games* respectively. Figure 2.6 visualizes the relationship among four classes of games defined so far.

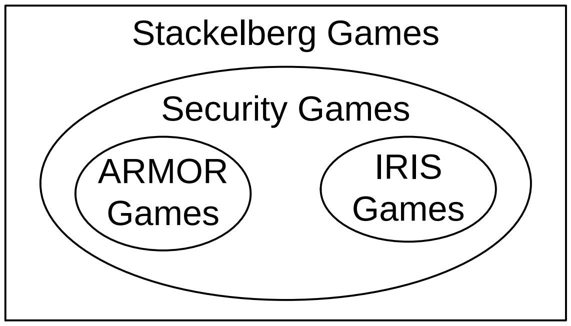


Figure 2.6: Relationship among Stackelberg, security, IRIS, and ARMOR games.

A security game described above can be represented as a strategic form game as follows. The attacker's pure strategy space is the set of targets. The attacker's mixed strategy $\mathbf{a} = (a_1, \ldots, a_N)$ is a vector where a_i represents the probability of attacking t_i. The defender's pure strategy is a feasible assignment of resources to schedules.Since covering a target with one resource is exactly the same as covering it with any positive number of resources, the defender's pure strategy can also be represented by a coverage vector $\mathbf{d} = (d_1, \ldots, d_N) \in \{0, 1\}^N$ where d_i represents whether t_i is covered or not. For example, $(\{t_1, t_4\}, \{t_2\})$ can be a possible assignment, and the corresponding coverage vector is $(1, 1, 0, 1)$. However, not all the coverage vectors are feasible due to resource and schedule constraints. Denote the set of feasible coverage vectors by $\mathcal{D} \subseteq \{0, 1\}^N$.

The defender's mixed strategy $\mathbf{X}$ specifies the probabilities of playing each $\mathbf{d} \in \mathcal{D}$, where each individual probability is denoted by $X_{\mathbf{d}}$. Let $\mathbf{x} = (x_1, \ldots, x_N)$ be the vector of coverage probabilities corresponding to $\mathbf{X}$, where $x_i = \sum_{\mathbf{d} \in \mathcal{D}} d_i X_{\mathbf{d}}$ is the marginal probability of covering t_i. For example, suppose the defender has two coverage vectors: $\mathbf{d}_1 = (1, 1, 0)$ and $\mathbf{d}_2 = (0, 1, 1)$. Then $\mathbf{X} = (.5, .5)$ is one defender's mixed strategy, and the corresponding $\mathbf{x} = (.5, 1, .5)$. Denote the mapping from $\mathbf{X}$ to $\mathbf{x}$ by φ, i.e., $\mathbf{x} = \varphi(\mathbf{X})$. For defender mixed strategy $\mathbf{X}$ and target t_i attacked, denote the defender's and the attacker's expected utility by $u(\mathbf{X}, t_i)$ and $v(\mathbf{X}, t_i)$ respectively. It is easy to see $u(\mathbf{X}, t_i) = x_i \mu_i^c + (1 - x_i)\mu_i^u$ and $v(\mathbf{X}, t_i) = x_i \nu_i^c + (1 - x_i)\nu_i^u$.

If strategy profile $(\mathbf{X}, \mathbf{a})$ is played, the defender's expected utility is

$$u(\mathbf{X}, \mathbf{a}) = \sum_{i=1}^{N} a_i u(\mathbf{X}, t_i) = \sum_{i=1}^{N} a_i \left[x_i \mu_i^c + (1 - x_i)\mu_i^u \right], \tag{2.4}$$

while the attacker's expected utility is

$$v(\mathbf{X}, \mathbf{a}) = \sum_{i=1}^{N} a_i v(\mathbf{X}, t_i) = \sum_{i=1}^{N} a_i \left[x_i \nu_i^c + (1 - x_i)\nu_i^u \right]. \tag{2.5}$$

Given a defender's mixed strategy $\mathbf{X}$, let $g(\mathbf{X}) : \mathbf{X} \to \mathbf{a}$ denotes the attacker's response function. Similar to Definition 1, a Strong Stackelberg Equilibrium in the security game context is defined below.

Definition 2. *A pair of strategies* $\langle \mathbf{X}, g \rangle$ *forms a* Strong Stackelberg Equilibrium *(SSE) of a security game if they satisfy the following:*

1. *The leader (defender) plays a best-response:*

   $u(\mathbf{X}, g(\mathbf{X})) \geq u(\mathbf{X}', g(\mathbf{X}'))$, *for all* $\mathbf{X}'$.

2. *The follower (attacker) plays a best-response:*

   $v(\mathbf{X}, g(\mathbf{X})) \geq v(\mathbf{X}, g'(\mathbf{X}))$*, for all* $\mathbf{X}, g'$.

3. *The follower breaks ties optimally for the leader:*

   $u(\mathbf{X}, g(\mathbf{X})) \geq u(\mathbf{X}, t_i)$*, for all* $\mathbf{X}$ *and for all* t_i *that is a best-response to* $\mathbf{X}$.

As we will see in Chapter 5, the defender in security games may not always have the power of commitment (acting as the leader) in certain situations. If the players move simultaneously, the standard solution concept is Nash equilibrium.

Definition 3. *A pair of strategies* $\langle \mathbf{X}, \mathbf{a} \rangle$ *forms a* Nash Equilibrium *(NE) of a security game if they satisfy the following:*

1. *The defender plays a best-response:*

   $u(\mathbf{X}, \mathbf{a}) \geq u(\mathbf{X}', \mathbf{a}) \ \forall \mathbf{X}'$.

2. *The attacker plays a best-response:*

   $v(\mathbf{X}, \mathbf{a}) \geq v(\mathbf{X}, \mathbf{a}') \ \forall \mathbf{a}'$.

For convenience, I denote the set of mixed strategies for the defender that are played in some Nash Equilibrium by Ω_{NE}, and the corresponding set for Strong Stackelberg Equilibrium by Ω_{SSE}.

Chapter 3: Stackelberg Games with Distributional Uncertainty

As discussed earlier, Bayesian Stackelberg game model is useful in modeling distributional uncertainty in Stackelberg games. A key challenge of applying Bayesian Stackelberg game models to real world problems is to scale up the number of follower types. Scalability of discrete follower types is essential in domains such as road network security Dickerson et al. [2010] and public transit network Yin et al. [2012a], where each follower type could represent an adversary attempting to follow a certain path. Scaling up the number of types is also necessary for the sampling-based algorithms Kiekintveld et al. [2011] to obtain high quality solutions under continuous uncertainty. Unfortunately, such scale-up remains difficult, as finding the equilibrium of a Bayesian Stackelberg game is NP-hard Conitzer and Sandholm [2006]. Indeed, despite the recent algorithmic advancement including Multiple-LPs Conitzer and Sandholm [2006], Dobss Paruchuri et al. [2008], HBGS Jain et al. [2011b], none of these techniques can handle games with more than ≈ 50 types, even when the number of actions per player is as few as 5: inadequate both for scale-up in discrete follower types and for sampling-based approaches.

This chapter presents a novel algorithm for solving Bayesian Stackelberg games called Hunter, combining techniques in artificial intelligence such as best-first search and operation research such as Bender's decomposition. In Section 3.1, I will describe the algorithmic details of Hunter. In Section 3.2, I will show how Hunter can be used, together with *sample average approximation* technique, to solve Stackelberg games with continuous uncertainty such as the defender's execution and the attacker's observation noise. Finally, Section 3.3 contains the experimental results of Hunter, demonstrating its superior scalability compared to existing algorithms.

3.1 Hunter: Discrete Uncertainty

In this section, I will present Hunter (**H**andling **UN**cer**T**ainty **E**fficiently using **R**elaxation) based on the five key ideas: i) best-first search for efficient pruning of the search tree; ii) a novel linear program relaxation for computing upper bounds in that search tree; iii) solving the upper bound LP efficiently using Bender's decomposition; iv) inheritance of Bender's cuts from parent nodes to child nodes for speedup; v) efficient heuristic branching rules utilizing the solution returned by the upper bound LP.

3.1.1 Algorithm Overview

To find the optimal leader's mixed strategy, Hunter would conduct a best-first search in the search tree that results from assigning follower types to pure strategies, such as the search tree in Figure 2.4. Simply stated, Hunter aims to search this space much more efficiently than HBGS Jain et al. [2011b]. As discussed earlier in Section 2.5.3, efficiency gains are sought by obtaining tight upper bounds and lower bounds at internal nodes in the search tree (which corresponds to a partial assignment in which a subset of follower types are fixed). To that end, as illustrated in Figure 3.1, we use an upper bound LP within an internal search node. The LP returns an upper bound UB and a feasible solution $\mathbf{x}^*$, which is then evaluated by computing the follower best response, providing a lower bound LB. The solution returned by the upper bound LP is also utilized in choosing a new type λ^* to create branches. To avoid having this upper bound LP itself become a bottleneck, it is solved efficiently using Bender's decomposition, which will be explained below.

To understand Hunter's behavior on a toy game instance, see Figure 3.2, which illustrates Hunter's search tree in solving the example game in Figure 2.3 (in Section 2.3). To start the

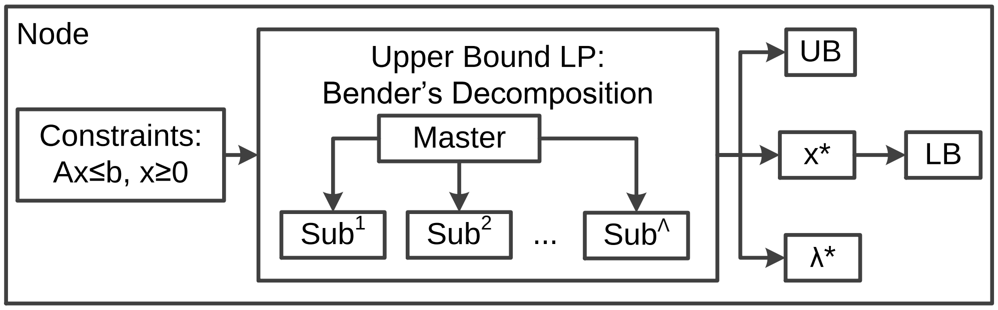


Figure 3.1: Steps of creating internal search nodes in Hunter.

best-first search, at the root node, no type is assigned any targets yet; we solve the upper bound LP with the initial strategy space $x_1 + x_2 \leq 1, x_1, x_2 \geq 0$ (Node 1). As a result, we obtain an upper bound of 0.560 and the optimal solution $x_1^* = 2/3, x_2^* = 1/3$. We evaluate the solution returned and obtain a lower bound of 0.506. Using Hunter's heuristics, type 2 is then chosen to create branches by assigning it to *Target1* and *Target2* respectively. Next, we consider a child node (Node 2) in which type 2 is assigned to *Target1*, i.e., type 2's best response is to attack *Target1*. As a result, the follower's expected utility of choosing *Target1* must be higher than that of choosing *Target2*, i.e., $-x_1 + x_2 \geq x_1 - x_2$, simplified as $x_1 - x_2 \leq 0$. Thus, in Node 2, we impose an additional constraint $x_1 - x_2 \leq 0$ on the strategy space and obtain an upper bound of 0.5. Since its upper bound is lower than the current lower bound 0.506, this branch can be pruned out. Next we consider the other child node (Node 3) in which type 2 is assigned to *Target2*. This time we add constraint $-x_1 + x_2 \leq 0$ instead, and obtain an upper bound of 0.506. Since the upper bound coincides with the lower bound, we do not need to expand the node further. Moreover, since we have considered both *Target1* and *Target2* for type 2, we can terminate the algorithm and return 0.506 as the optimal solution value.

Now let us discuss Hunter's behavior line-by-line (see Algorithm 1). We initialize the best-first search by creating the root node of the search tree with no assignment of types to targets

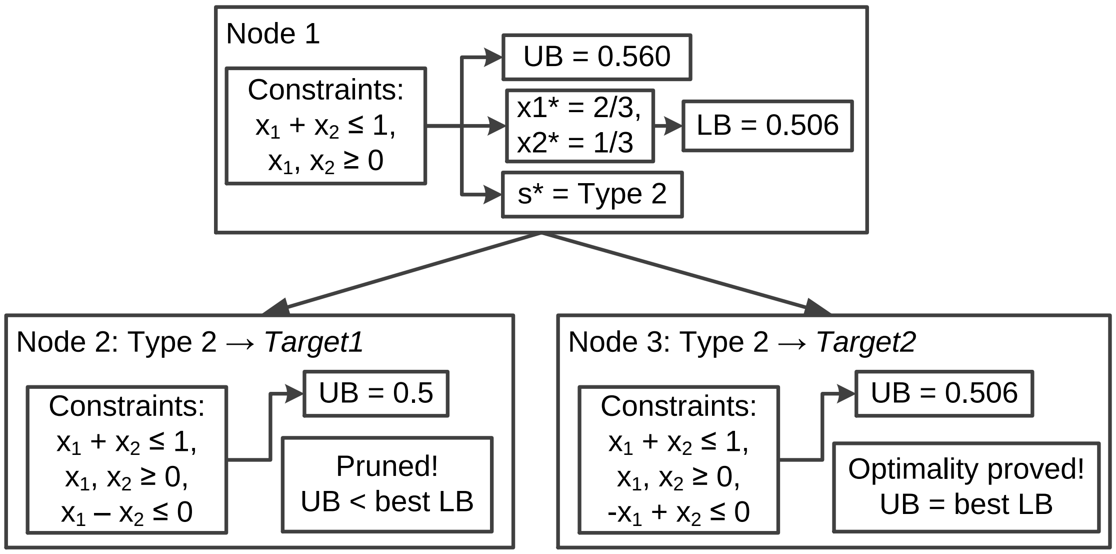

Node 1
Constraints: x1 + x2 ≤ 1, x1, x2 ≥ 0
UB = 0.560
x1* = 2/3, x2* = 1/3
LB = 0.506
s* = Type 2
Node 2: Type 2 → Target1
Constraints: x1 + x2 ≤ 1, x1, x2 ≥ 0, x1 – x2 ≤ 0
UB = 0.5
Pruned! UB < best LB
Node 3: Type 2 → Target2
Constraints: x1 + x2 ≤ 1, x1, x2 ≥ 0, -x1 + x2 ≤ 0
UB = 0.506
Optimality proved! UB = best LB

Figure 3.2: Example of internal nodes in HUNTER's search tree.

and with the computation of the node's upper bound (Line 2 and 3). The initial lower bound is obtained by evaluating the solution returned by the upper bound LP (Line 4). We add the root node to a priority queue of open nodes which is internally sorted in a decreasing order of their upper bounds (Line 5). Each node contains information of the partial assignment, the feasible region of **x**, the upper bound, and the Bender's cuts generated by the upper bound LP. At each iteration, we retrieve the node with the highest upper bound (Line 8), select a type λ^* to assign pure strategies (Line 9), compute the upper bounds of the node's child nodes (Line 12 and 14), update the lower bound using the new solutions (Line 15), and enqueue child nodes with upper bound higher than the current lower bound (Line 16). As shown later, Bender's cuts at a parent node can be inherited by its children, speeding up the computation (Line 12).

In the rest of the section, I will 1) present the upper bound LP, 2) show how to solve it using Bender's decomposition, and 3) verify the correctness of passing down Bender's cuts from parent to child nodes, 4) introduce the heuristic branching rule.

Algorithm 1: Hunter

```
Initialization;
[UB, $\mathbf{x}^*$, BendersCuts] = SolveUBLP($\phi$, $A\mathbf{x} \leq \mathbf{b}$, $-\infty$);
root := $\langle$ UB, $\mathbf{x}^*$, $A\mathbf{x} \leq \mathbf{b}$, $x \geq \mathbf{0}$, BendersCuts $\rangle$ ;
LB := Evaluate($\mathbf{x}^*$);
Enqueue(queue, root);
Best-first Search;
while not Empty(queue) do
    node := pop(queue);
    $\lambda^*$ := PickType(node);
    for $j$ := 1 to $J$ do
        NewConstraints := node.Constraints $\cup\{D_j^{\lambda^*}\mathbf{x} + \mathbf{d}_j^{\lambda^*} \geq \mathbf{0}\}$ ;
        [NewUB, $\mathbf{x}'$, NewBendersCuts] = SolveUBLP(node.BendersCuts,
        NewConstraints, LB) ;
        if $NewUB > LB$ then
            child := $\langle$ NewUB, $\mathbf{x}'$, NewConstraints, NewBendersCuts$\rangle$ ;
            LB := max{Evaluate($\mathbf{x}'$), LB} ;
            Enqueue(queue, child);
        end
    end
end
```

3.1.2 Upper Bound Linear Program

In this section, I will derive a tractable linear relaxation of Bayesian Stackelberg games to provide an upper bound efficiently at each of Hunter's internal nodes. For expository purpose, let us focus on the root node of the search tree. Applying the results in disjunctive program Balas [1998], I will first derive the convex hull for a single type. Then I will show intersecting the convex hulls of all its types provides a tractable, polynomial-size relaxation of the entire Bayesian Stackelberg game.

3.1.2.1 Convex hull of a Single Type

Consider a Stackelberg game with a single follower type (U, V), the leader's optimal strategy $\mathbf{x}^*$ is the best among the optimal solutions of J LPs where each restricts the follower's best response to one pure strategy Conitzer and Sandholm [2006]. Hence we can represent the optimization problem as the following disjunctive program (i.e., a disjunction of ”Multiple LPs” Conitzer and Sandholm [2006]),

$$\begin{aligned} \max_{\mathbf{x},u} \quad & u \\ s.t. \quad & A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0} \\ & \bigvee_{j=1}^{J} \begin{pmatrix} u \leq \boldsymbol{\mu}_j^{\mathrm{T}}\mathbf{x} + \mu_{j,0} \\ D_j\mathbf{x} + \mathbf{d}_j \leq \mathbf{0} \end{pmatrix} \end{aligned} \tag{3.1}$$

where D_j’s and $\mathbf{d}_j$’s are given by,

$$D_j = \begin{pmatrix} \boldsymbol{\nu}_1^{\mathrm{T}} - \boldsymbol{\nu}_j^{\mathrm{T}} \\ \vdots \\ \boldsymbol{\nu}_J^{\mathrm{T}} - \boldsymbol{\nu}_j^{\mathrm{T}} \end{pmatrix}, \mathbf{d}_j = \begin{pmatrix} \nu_{1,0} - \nu_{j,0} \\ \vdots \\ \nu_{J,0} - \nu_{j,0} \end{pmatrix}.$$

The feasible set of (3.1), denoted by H, is a union of J convex sets, each corresponding to a disjunctive term. Applying the results in Balas [1998], the closure of the convex hull of H, clconv$H$, is[1],

$$\mathrm{clconv}H = \left\{ \begin{array}{c|l} & \mathbf{x} = \sum_{j=1}^{J} \chi_j, \chi_j \geq \mathbf{0}, \forall j \\ u \in \mathbb{R} & u = \sum_{j=1}^{J} \psi_j, \psi_j \geq 0, \forall j \\ \mathbf{x} \in \mathbb{R}^n & \sum_{j=1}^{J} \theta_j = 1, \theta_j \geq 0, \forall j \\ & \begin{pmatrix} A & -\mathbf{b} & \mathbf{0} \\ D_j & \mathbf{d}_j & \mathbf{0} \\ -\boldsymbol{\mu}_j^{\mathrm{T}} & -\mu_{j,0} & 1 \end{pmatrix} \begin{pmatrix} \chi_j \\ \theta_j \\ \psi_j \end{pmatrix} \leq \mathbf{0}, \forall j \end{array} \right\}.$$

The intuition here is that the continuous variables $\boldsymbol{\theta}, \sum_{j=1}^{J} \theta_j = 1$ are used to create all possible convex combination of points in H. Furthermore, when $\theta_j \neq 0$, $\langle \frac{\chi_j}{\theta_j}, \frac{\psi_j}{\theta_j} \rangle$ represents a point in the convex set defined by the j-th disjunctive term in the original problem (3.1). Finally, since all the extreme points of clconv$H$ belong to H, the disjunctive program (3.1) is equivalent to the linear program:

$$\max_{\mathbf{x},u} \{u | (\mathbf{x}, u) \in \mathrm{clconv}H\}.$$

This result is important, as it shows that one can use a single linear program (as opposed to multiple LPs Conitzer and Sandholm [2006] or a mixed integer LP Paruchuri et al. [2008]) to solve a Stackelberg game with a single type.

[1]To use the results in Balas [1998], we assume $u \geq 0$ for convenience. In the case where u can be negative, we can replace u by $u^+ - u^-$, with $u^+, u^- \geq 0$.

3.1.2.2 Tractable Relaxation

Building on the convex hulls of individual types, I will now derive the relaxation of a Bayesian Stackelberg game with S types. Let us rewrite this game with Λ types as the following disjunctive program,

$$\begin{aligned}
\max_{\mathbf{x},u^1,\ldots,u^\Lambda} \quad & \sum_{\lambda=1}^{\Lambda} p^\lambda u^\lambda \\
s.t. \quad & A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0} \\
& \bigwedge_{s=1}^{\Lambda} \left[\bigvee_{j=1}^{J} \begin{pmatrix} u^\lambda \leq (\boldsymbol{\mu}_j^\lambda)^{\mathrm{T}}\mathbf{x} + \mu_{j,0}^\lambda \\ D_j^\lambda \mathbf{x} + \mathbf{d}_j^\lambda \leq \mathbf{0} \end{pmatrix} \right]
\end{aligned} \tag{3.2}$$

Returning to the toy example, the corresponding disjunctive program of the game in Figure 2.3 can be written as,

$$\begin{aligned}
\max_{x_1,x_2,u^1,u^2} \quad & 0.84u^1 + 0.16u^2 \\
s.t. \quad & x_1 + x_2 \leq 1, x_1, x_2 \geq 0 \\
& \begin{pmatrix} u^1 \leq x_1 \\ x_1 - 2x_2 \leq 0 \end{pmatrix} \bigvee \begin{pmatrix} u^1 \leq -x_1 + x_2 \\ -x_1 + 2x_2 \leq 0 \end{pmatrix} \\
& \begin{pmatrix} u^2 \leq x_1 \\ x_1 - x_2 \leq 0 \end{pmatrix} \bigvee \begin{pmatrix} u^2 \leq -x_1 + x_2 \\ -x_1 + x_2 \leq 0 \end{pmatrix}
\end{aligned} \tag{3.3}$$

Denote the set of feasible points $(\mathbf{x}, u^1, \ldots, u^\Lambda)$ of (3.2) by H^*. Unfortunately, to use the results of Balas [1998] here and create clconvH^*, we need to expand (3.2) to a disjunctive normal form, resulting in a linear program with an exponential number ($O(NJ^\Lambda)$) of variables. Instead, I now give a much more tractable, polynomial-size relaxation of (3.2). Denote the feasible set of each type λ, $(\mathbf{x}, u^\lambda)$ by H^λ, and define $\widehat{H^*} := \{(\mathbf{x}, u^1, \ldots, u^\Lambda) | (\mathbf{x}, u^\lambda) \in \text{clconv} H^\lambda, \forall 1 \leq \lambda \leq \Lambda\}$. Then the following program is a relaxation of (3.2):

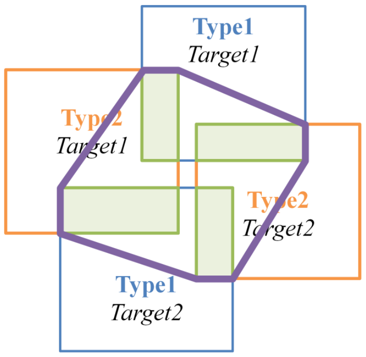
$$\max_{\mathbf{x}, u^1, \ldots, u^\Lambda} \left\{ \sum_{\lambda=1}^{\Lambda} p^\lambda u^\lambda | (\mathbf{x}, u^\lambda) \in \text{clconv} H^\lambda, \forall 1 \leq \lambda \leq \Lambda \right\} \tag{3.4}$$

Indeed, for any feasible point $(\mathbf{x}, u^1, \ldots, u^\Lambda)$ in H^*, $(\mathbf{x}, u^\lambda)$ must belong to H^λ, implying that $(\mathbf{x}, u^\lambda) \in \text{clconv} H^\lambda$. Hence $H^* \subseteq \widehat{H^*}$, implying that optimizing over $\widehat{H^*}$ provides an upper bound on H^*. On the other hand, $\widehat{H^*}$ will in general have points not belonging to H^* and thus the relaxation can lead to an overestimation.

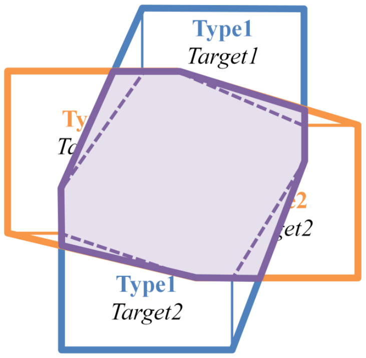
For example, consider the disjunctive program in (3.3). $(x_1 = \frac{2}{3}, x_2 = \frac{1}{3}, u^1 = \frac{2}{3}, u^2 = 0)$ does not belong to H^* since $-x_1 + x_2 \leq 0$ but $u^2 \nleq -x_1 + x_2 = -\frac{1}{3}$. However the point belongs to $\widehat{H^*}$ because: i) $(x_1 = \frac{2}{3}, x_2 = \frac{1}{3}, u^1 = \frac{2}{3})$ belongs to $H^1 \subseteq \text{clconv} H^1$; ii) $(x_1 = \frac{2}{3}, x_2 = \frac{1}{3}, u^2 = 0)$ belongs to $\text{clconv} H^2$, as it is the convex combination of two points in H^2: $(x_1 = \frac{1}{2}, x_2 = \frac{1}{2}, u^2 = \frac{1}{2})$ and $(x_1 = 1, x_2 = 0, u^2 = -1)$,

$$(\frac{2}{3}, \frac{1}{3}, 0) = \frac{2}{3} \times (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}) + \frac{1}{3} \times (1, 0, -1).$$

Perhaps a better way to understand the Hunter relaxation is through the following demonstrative example shown in Figure 3.3. In Figure 3.3(a), the blue and orange rectangles represent the solution spaces of follower type 1 and 2 respectively, i.e., the blue rectangles represent $\{(x_1, x_2, u^1, u^2) \mid (x_1, x_2, u^1) \in H^1, u^2 \in \mathbb{R}\}$ and the red rectangles represent $\{(x_1, x_2, u^1, u^2) \mid (x_1, x_2, u^2) \in H^2, u^1 \in \mathbb{R}\}$. For each type, the two rectangles represent the two disjoint sets corresponding to attacking one of the two targets respectively. Then the intersection



(a) Convex hull clconvH^*.



(b) Relaxation $\widehat{H^*}$.

Figure 3.3: Visualization of the HUNTER relaxation.

of the rectangles of the two types, shown as the green rectangles, represents the feasible solution space H^*. Built upon the four disjoint green rectangles, the convex hull of H^* (clconvH^*) is the area within the purple lines. Shown in Figure 3.3(b), the HUNTER relaxation $\widehat{H^*}$ is the intersection of the convex hulls of the two types, i.e., the purple region within the solid purple lines. As can be easily visualized in Figure 3.3(b), $\widehat{H^*}$ is indeed a relaxation compared to clconvH^*, the area within the dashed purple lines.

The upper bound LP (3.4) has $O(NJ\Lambda)$ number of variables and constraints, and can be written as the following two-stage problem by explicitly representing clconvH^s:

$$\begin{aligned} \max_{\mathbf{x}} \quad & \sum_{\lambda=1}^{\Lambda} p^{\lambda} u^{\lambda}(\mathbf{x}) \\ s.t. \quad & A\mathbf{x} \preceq \mathbf{b}, \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{3.5}$$

where $u^\lambda(\mathbf{x})$ is defined to be the optimal value of,

$$\begin{aligned}
\max_{\chi_j^\lambda,\psi_j^\lambda,\theta_j^\lambda} \quad & \sum_{j=1}^{J} \psi_j^\lambda \\
s.t. \quad & \sum_{j=1}^{J} \chi_j^\lambda = \mathbf{x}, \ \ \forall 1 \le \lambda \le \Lambda \\
& \sum_{j=1}^{J} \theta_j^\lambda = 1, \ \ \forall 1 \le \lambda \le \Lambda \\
& \psi_j^\lambda, \theta_j^\lambda \ge 0, \ \chi_j^\lambda \ge \mathbf{0}, \ \ \forall 1 \le \lambda \le \Lambda, \forall 1 \le j \le J \\
& \begin{pmatrix} A & -\mathbf{b} & \mathbf{0} \\ D_j^\lambda & \mathbf{d}_j^\lambda & \mathbf{0} \\ -(\boldsymbol{\mu}_j^\lambda)^{\mathrm{T}} & -\mu_{j,0}^\lambda & 1 \end{pmatrix} \begin{pmatrix} \chi_j^\lambda \\ \theta_j^\lambda \\ \psi_j^\lambda \end{pmatrix} \le \mathbf{0}, \forall 1 \le \lambda \le \Lambda, \forall 1 \le j \le J
\end{aligned} \tag{3.6}$$

Although written in two stages, the above formulation is in fact a single linear program, as both stages are maximization problems and combining the two stages will not produce any non-linear terms. I display formulations (3.5) and (3.6) in order to reveal the block structure for further speedup as explained below.

Note that so far, we have only derived the relaxation for the root node of Hunter's search tree, without assigning any type to a pure strategy. This relaxation is also applied to other internal nodes in Hunter's search tree. For example, if type λ is assigned to pure strategy j, the leader's strategy space is further restricted by the addition of constraints of $D_j^\lambda \mathbf{x} + \mathbf{d}_j^\lambda \le \mathbf{0}$ to the original constraints $A\mathbf{x} \le \mathbf{b}, \mathbf{x} \ge \mathbf{0}$. That is, we now have obtained the same form of constraints as in the root node: $A'\mathbf{x} \le \mathbf{b}', \mathbf{x} \ge \mathbf{0}$ where $A' = \begin{pmatrix} D_j^\lambda \\ A \end{pmatrix}$ and $\mathbf{b}' = \begin{pmatrix} -\mathbf{d}_j^\lambda \\ \mathbf{b} \end{pmatrix}$.

3.1.3 Bender's Decomposition

Although much easier than solving a full Bayesian Stackelberg game, solving the upper bound LP can still be computationally challenging. Here we invoke the block structure of (3.4) observed above, which partitioned it into (3.5) and (3.6), where, (3.5) is a master problem and (3.6) for $\lambda = 1, \ldots, \Lambda$ are Λ subproblems. This block structure allows us to solve the upper bound LP efficiently using multi-cut Bender's Decomposition Birge and Louveaux [1988]. Generally speaking, the computational difficulty of optimization problems increases significantly with the number of variables and constraints. Instead of considering all variables and constraints of a large problem simultaneously, Bender's decomposition partitions the problem into multiple smaller problems, which can then be solved in sequence. For completeness, I will briefly describe the technique here in the context of solving LP formulation (3.5) - (3.6). General detailed explanation of Bender's decomposition can be found in Appendix A.

In Bender's decomposition, the second-stage maximization problem (3.6) is replaced by its dual minimization counterpart, with dual variables $\boldsymbol{\omega}_j^\lambda, \boldsymbol{\pi}^\lambda, \eta^\lambda$ for $\lambda = 1, \ldots, \Lambda$:

$$\begin{aligned} u^\lambda(\mathbf{x}) = & \min_{\boldsymbol{\omega}_j^\lambda \geq \mathbf{0}, \boldsymbol{\pi}^\lambda, \eta^\lambda} (\boldsymbol{\pi}^\lambda)^{\mathrm{T}}\mathbf{x} + \eta^\lambda \\ s.t. & \begin{pmatrix} A^{\mathrm{T}} & (D_j^\lambda)^{\mathrm{T}} & -\boldsymbol{\mu}_j^\lambda \\ -\mathbf{b}^{\mathrm{T}} & (\mathbf{d}_j^\lambda)^{\mathrm{T}} & -\mu_{j,0}^\lambda \\ \mathbf{0}^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix} \boldsymbol{\omega}_j^\lambda + \begin{pmatrix} \boldsymbol{\pi}^\lambda \\ \eta^\lambda \\ -1 \end{pmatrix} \geq \mathbf{0}, \forall 1 \leq j \leq J \end{aligned} \tag{3.7}$$

Since the feasible region of (3.7) is independent of $\mathbf{x}$, its optimal solution is reached at one of a finite number of extreme points (of the dual variables). Since $u^\lambda(\mathbf{x})$ is the minimum of

$(\boldsymbol{\pi}^\lambda)^\mathrm{T}\mathbf{x} + \eta^\lambda$ over all possible dual points, we know the following inequality must be true in the master problem,

$$u^\lambda \leq (\boldsymbol{\pi}_k^\lambda)^\mathrm{T}\mathbf{x} + \eta_k^\lambda, \quad k = 1, \ldots, K \tag{3.8}$$

where $(\boldsymbol{\pi}_k^\lambda, \eta_k^\lambda), k = 1, \ldots, K$ are all the dual extreme points. Constraints of type (3.8) for the master problem are called optimality cuts (infeasibility cuts, another type of constraint, turn out not to be relevant in this context).

Since there are typically exponentially many extreme points for the dual formulation (3.7), generating all constraints of type (3.8) is not practical. Instead, Bender's decomposition starts by solving the master problem (3.5) with a subset of these constraints to find a candidate optimal solution $(\mathbf{x}^*, u^{1,*}, \ldots, u^{\Lambda,*})$. It then solves Λ dual subproblems (3.7) to calculate $u^\lambda(\mathbf{x}^*)$. If all the subproblems have $u^\lambda(\mathbf{x}^*) = u^{\lambda,*}$, the algorithm stops. Otherwise for those $u^\lambda(\mathbf{x}^*) < u^{\lambda,*}$, the corresponding constraints of type (3.8) are added to the master program for the next iteration.

As a numerical example, let us consider the example given in Figure 3.2 again. As mentioned earlier the example problem can be written as disjunctive program (3.3). To illustrate the process of applying Bender's decomposition to solve an upper bound program, let us focus on the root search node where no type has been assigned to any target yet. At the beginning, the master program (3.5) has no Bender's cuts:

$$\begin{aligned} \max_{x_1, x_2} \quad & 0.84u^1 + 0.16u^2 \\ s.t. \quad & x_1 + x_2 = 1, x_1, x_2 \geq 0. \end{aligned}$$

Although the above master program is unbounded, an arbitrary feasible strategy $\mathbf{x}$ can be returned as the optimal solution. Without loss of generality, let $x_1^{(1)} = 1$ and $x_2^{(1)} = 0$ be the optimal $\mathbf{x}$

returned from the master program in the first iteration. Similarly, I will denote by $x_1^{(k)}$, $x_2^{(k)}$, $u^{1,(k)}$, and $u^{2,(k)}$ the optimal solution obtained from the master program in the k^{th} iteration. Since the first master program is unbounded, $u^{1,(1)}$ and $u^{2,(1)}$ can be considered as $+\infty$.

Given a solution $x_1^{(k)}$ and $x_2^{(k)}$, two subproblems corresponding to the two follower types need to be solved. For better readability, I will give the subproblems in their primal form only, although the dual solution is used to construct the Bender's cuts. Note when solving the primal problem using primal-dual methods, the values of dual variables can be obtained as well. The subproblem for follower type 1 is:

$$\begin{aligned}
\max_{\chi_1^1,\chi_2^1,\psi_1^1,\psi_2^1,\theta_1^1,\theta_2^1} \quad & \psi_1^1+\psi_2^1, \psi_1^1 \leq \theta_1^1, \psi_2^1 \leq \theta_2^1 \\
s.t. \quad & \chi_{1,1}^1+\chi_{1,2}^1 = x_1^{(k)}, 0 \leq \chi_{1,1}^1 \leq \theta_1^1, 0 \leq \chi_{1,2}^1 \leq \theta_2^1 \\
& \chi_{2,1}^1+\chi_{2,2}^1 = x_2^{(k)}, 0 \leq \chi_{2,1}^1 \leq \theta_1^1, 0 \leq \chi_{2,2}^1 \leq \theta_2^1 \\
& \theta_1^1+\theta_2^1 = 1, \theta_1^1, \theta_2^1 \geq 0 \\
& -\chi_{1,1}^1+\chi_{2,1}^1 \geq -\chi_{2,1}^1 \\
& \psi_1^1 \leq \chi_{1,1}^1 \\
& -\chi_{1,2}^1+\chi_{2,2}^1 \leq \chi_{2,2}^1 \\
& \psi_2^1 \leq -\chi_{1,2}^1+\chi_{2,2}^1
\end{aligned}$$

Similarly, the subproblem for follower type 2 is:

$$\begin{aligned}
\max_{\chi_1^2,\chi_2^2,\psi_1^2,\psi_2^2,\theta_1^2,\theta_2^2} \quad & \psi_1^2 + \psi_2^2, \psi_1^2 \leq \theta_1^2, \psi_2^2 \leq \theta_2^2 \\
s.t. \quad & \chi_{1,1}^2 + \chi_{1,2}^2 = x_1^{(k)}, 0 \leq \chi_{1,1}^2 \leq \theta_1^2, 0 \leq \chi_{1,2}^2 \leq \theta_2^2 \\
& \chi_{2,1}^2 + \chi_{2,2}^2 = x_2^{(k)}, 0 \leq \chi_{2,1}^2 \leq \theta_1^2, 0 \leq \chi_{2,2}^2 \leq \theta_2^2 \\
& \theta_1^2 + \theta_2^2 = 1, \theta_1^2, \theta_2^2 \geq 0 \\
& -\chi_{1,1}^2 + \chi_{2,1}^2 \geq \chi_{1,1}^2 - \chi_{2,1}^2 \\
& \psi_1^2 \leq \chi_{1,1}^2 \\
& -\chi_{1,2}^2 + \chi_{2,2}^2 \leq \chi_{1,2}^2 - \chi_{2,2}^2 \\
& \psi_2^2 \leq -\chi_{1,2}^2 + \chi_{2,2}^2
\end{aligned}$$

Recall in the first iteration, $x_1^{(1)} = 1$ and $x_2^{(1)} = 0$. The two subproblems are both feasible and bounded. Solving the two subproblems gives the dual solutions $\pi_1^1 = -1$, $\pi_2^1 = 4$, $\eta^1 = 0$, and $\pi_1^2 = -1$, $\pi_2^2 = 2$, $\eta^2 = 0$. Here recall that π_j^λ is the dual variable corresponding to the constraint $\sum_{j'}^J \chi_{j,j'}^\lambda = x_j$ and η^λ is the dual variable corresponding to the constraint $\sum_j^J \theta_j^\lambda = 1$. The optimal solutions of both subproblems are -1.0, lower than $u^{1,(1)}$ and $u^{2,(1)}$. Therefore each subproblem can generate one Bender's cut to be added to the master problem. The two cuts are $u^1 \leq -x_1 + 4x_2$ and $u^2 \leq -x_1 + 2x_2$.

After adding the cuts, the master program becomes the following in the second iteration:

$$\begin{aligned}
\max_{x_1,x_2} \quad & 0.84u^1 + 0.16u^2 \\
s.t. \quad & x_1 + x_2 = 1, x_1, x_2 \geq 0 \\
& u^1 \leq -x_1 + 4x_2 \\
& u^2 \leq -x_1 + 2x_2.
\end{aligned}$$

The new optimal solution of the master program is $x_1^{(2)} = 0$, $x_2^{(2)} = 1$, $u^{1,(2)} = 4$, and $u^{2,(2)} = 2$. Solving the subproblems again generates two Bender's cuts: $u^1 \leq x_1$ and $u^2 \leq x_1$. Hence the master program becomes:

$$\begin{aligned}
\max_{x_1,x_2} \quad & 0.84u^1 + 0.16u^2 \\
s.t. \quad & x_1 + x_2 = 1, x_1, x_2 \geq 0 \\
& u^1 \leq -x_1 + 4x_2 \\
& u^2 \leq -x_1 + 2x_2 \\
& u^1 \leq x_1 \\
& u^2 \leq x_1.
\end{aligned}$$

The optimal solution of the third iteration is $x_1^{(3)} = 2/3$, $x_2^{(3)} = 1/3$, $u^{1,(3)} = 2/3$, and $u^{2,(3)} = 0$. This time, the optimal values of the two subproblems are 2/3 and 0 respectively. Since these optimal values are the same as $u^{1,(3)}$ and $u^{2,(3)}$ respectively, no further Bender's cut needs to be added. Therefore the process of Bender's decomposition terminates with an upper bound value of $0.84 \times 2/3 + 0.16 \times 0 = 0.56$ at the root node of the search tree.

3.1.4 Reusing Bender's Cuts

It is possible to further speed up the upper bound LP computation at internal nodes of Hunter's search tree by not creating all of the Bender's cuts from scratch; instead, the Bender's cuts from the parent node can be reused in its children. Suppose $u^\lambda \leq (\boldsymbol{\pi}^\lambda)^\mathrm{T}\mathbf{x} + \eta^\lambda$ is a Bender's cut in the parent node. This means u^λ cannot be greater than $(\boldsymbol{\pi}^\lambda)^\mathrm{T}\mathbf{x} + \eta^\lambda$ for any $\mathbf{x}$ in the feasible region of the parent node. Intuitively because a child node's feasible region is always more restricted than its parent's, it can be concluded that u^λ cannot be greater than $(\boldsymbol{\pi}^\lambda)^\mathrm{T}\mathbf{x} + \eta^\lambda$ for any $\mathbf{x}$ in the child node's feasible region. Hence, $u^\lambda \leq (\boldsymbol{\pi}^\lambda)^{\mathrm{T}}\mathbf{x} + \eta^\lambda$ must also be a valid cut for the child node. The following Proposition provides a formal proof.

Proposition 1. *The Bender's cuts generated for a parent node are valid cuts for its child nodes.*

Proof. Let the feasible region of a parent node be $A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}$, and the feasible region of a child node be $A'\mathbf{x} \leq \mathbf{b}', \mathbf{x} \geq \mathbf{0}$, where $A' = \begin{pmatrix} \tilde{A} \\ A \end{pmatrix}$ and $\mathbf{b}' = \begin{pmatrix} \tilde{\mathbf{b}} \\ \mathbf{b} \end{pmatrix}$. Assume $u^\lambda \leq (\boldsymbol{\pi}^\lambda)^{\mathrm{T}}\mathbf{x} + \eta^\lambda$ is a cut of the parent node, implying there exists $\boldsymbol{\omega}_j^\lambda \geq \mathbf{0}$, for all $j = 1, \ldots, J$, such that,

$$\begin{pmatrix} A^{\mathrm{T}} & (D_j^\lambda)^{\mathrm{T}} & -\boldsymbol{\mu}_j^\lambda \\ -\mathbf{b}^{\mathrm{T}} & (\mathbf{d}_j^\lambda)^{\mathrm{T}} & -\mu_{j,0}^\lambda \\ \mathbf{0}^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix} \boldsymbol{\omega}_j^\lambda + \begin{pmatrix} \boldsymbol{\pi}^\lambda \\ \eta^\lambda \\ -1 \end{pmatrix} \geq \mathbf{0}, \forall 1 \leq j \leq J$$

Then $\boldsymbol{\pi}^\lambda$, η^λ for all $\lambda = 1, \ldots, \Lambda$ is a feasible point of the dual problem (3.7) for the child node because,

$$\begin{pmatrix} \tilde{A}^{\mathrm{T}} & A^{\mathrm{T}} & (D_j^\lambda)^{\mathrm{T}} & -\boldsymbol{\mu}_j^\lambda \\ -\tilde{\mathbf{b}}^{\mathrm{T}} & -\mathbf{b}^{\mathrm{T}} & (\mathbf{d}_j^\lambda)^{\mathrm{T}} & -\mu_{j,0}^\lambda \\ \mathbf{0}^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ \boldsymbol{\lambda}_j^\lambda \end{pmatrix} + \begin{pmatrix} \boldsymbol{\pi}^\lambda \\ \eta^\lambda \\ -1 \end{pmatrix} \geq \mathbf{0}, \forall 1 \leq j \leq J$$

The above result implies $u^\lambda \leq (\boldsymbol{\pi}^\lambda)^{\mathrm{T}}\mathbf{x} + \eta^\lambda$ is a valid cut for the child node. □

3.1.5 Heuristic Branching Rules

Given an internal node in the search tree of Hunter, one must decide on the type to branch on next, i.e., the type for which J child nodes will be created at the next lower level of the tree. The simplest way of selecting such type is to randomly choose one type that has been selected before. However, as I will show in Section 3.3 later, this branching type has a significant effect on efficiency and therefore it is important to choose such type intelligently. While multiple heuristics can be developed, I will limit the focus to the following one within the scope of this thesis.

Throughout the branch-and-bound search process, after a new search node is evaluated, the global lower bound increases and the maximum upper bound decreases. The algorithm terminates with the optimal solution when the lower bound meets the upper bound. Hence intuitively, one should select a type whereby the upper bound at these children nodes will decrease the most significantly. While the best type can be found by a one-step lookahead, such lookahead requires solving many upper bound linear programs and incurs significant extra runtime. It is therefore desirable to choose one type heuristically without further lookahead.

To this end, Hunter chooses the type whose $\boldsymbol{\theta}^{\lambda}$ returned by (3.6) which violates the integrality constraint the most. By branching on this type, the integrality constraint of its $\boldsymbol{\theta}^{\lambda}$ must be satisfied. This in turn will reduce the upper bound as the problem becomes more constrained. Recall that $\boldsymbol{\theta}^{\lambda}$ is used to generate the convex combinations. If all $\boldsymbol{\theta}^{\lambda}$ returned by (3.6) are integer vectors, the solution of the upper bound LP (3.5) and (3.6) is a feasible point of the original problem (3.2), implying the relaxed LP already returns the optimal solution. More specifically, as inspired by Gilpin and Sandholm [2011], Hunter chooses type λ^* whose corresponding $\boldsymbol{\theta}^{\lambda^*}$ has the maximum entropy, i.e., $\lambda^* = \arg\max_{\lambda} - \sum_{j=1}^{J} \theta_j^{\lambda} \log \theta_j^{\lambda}$.

3.2 Extension to Continuous Uncertainty

This section extends Hunter to handle continuous uncertainty via the *sample average approximation* technique Ahmed et al. [2002]. I first introduce the uncertain Stackelberg game model with continuously distributed uncertainty in leader's execution, follower's observation, and both players' utilities. Then I show the uncertain Stackelberg game model can be written as a two-stage mixed-integer stochastic program, to which existing convergence results of the sample average approximation technique apply. Finally, I show the sampled problems are equivalent to Bayesian Stackelberg games, and consequently could also be solved by Hunter.

3.2.1 Uncertain Stackelberg Game Model

Let us consider the following types of uncertainty in Stackelberg games with known distributions. First, similar to Kiekintveld et al. [2011], I assume there is uncertainty in both the leader and the follower's utilities U and V. Second, the leader's execution and the follower's observation can also be noisy. More specifically, I assume the executed strategy and observed strategy are linear perturbations of the intended strategy, i.e., when the leader commits to $\mathbf{x}$, the actual executed strategy is $\mathbf{y} = F^{\mathrm{T}}\mathbf{x} + \mathbf{f}$ and the observed strategy by the follower is $\mathbf{z} = G^{\mathrm{T}}\mathbf{x} + \mathbf{g}$, where $(F, \mathbf{f})$ and $(G, \mathbf{g})$ are uncertain. Intuitively $\mathbf{f}$ and $\mathbf{g}$ are used to represent the execution and observation noise that is independent on $\mathbf{x}$, while F and G are $N \times N$ matrices representing execution and observation noise that is linearly dependent on $\mathbf{x}$. For example, we can represent an execution noise that is independent of $\mathbf{x}$ and follows a Gaussian distribution with $\mathbf{0}$ mean using $F = I_N$ and $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \Sigma)$, where I_N is the $N \times N$ identity matrix. U, V, F, $\mathbf{f}$, G, and $\mathbf{g}$ are random variables that follow some known continuous (joint) distributions. Note that G and $\mathbf{g}$ can be dependent on F and $\mathbf{f}$ to capture the correlation between the defender's executed strategy and the attacker's observed strategy. We use a vector $\xi = (U, V, F, \mathbf{f}, G, \mathbf{g})$ to represent a realization of the above inputs, and use the notation $\xi(\varpi)$ to represent the corresponding random variable.

I now show the uncertain Stackelberg game can be written as a two-stage mixed-integer stochastic program. Let $Q(\mathbf{x}, \xi)$ be the leader's utility for a strategy $\mathbf{x}$ and a realization ξ, assuming the follower chooses the best response. The first stage maximizes the expectation of leader's utility with respect to the joint probability distribution of $\xi(\omega)$, i.e., $\max_{\mathbf{x}} \{\mathbb{E}[Q(\mathbf{x}, \xi(\varpi))] | A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$. The second stage computes $Q(\mathbf{x}, \xi)$:[2]

$$\begin{aligned} Q(\mathbf{x}, \xi) = &\quad \boldsymbol{\mu}_{j^*}^{\mathrm{T}}(F^{\mathrm{T}}\mathbf{x} + \mathbf{f}) + \mu_{j^*,0} \\ \text{where} \qquad j^* = &\quad \arg\max_{j=1}^{J} \boldsymbol{\nu}_j^{\mathrm{T}}(G^{\mathrm{T}}\mathbf{x} + \mathbf{g}) + \nu_{j,0}. \end{aligned} \tag{3.9}$$

3.2.2 Sample Average Approximation

Sample average approximation is a popular solution technique for stochastic programs with continuously distributed uncertainty Ahmed et al. [2002]. It can be applied to solving uncertain Stackelberg games as follows. First, a sample $\xi^1, \ldots, \xi^\Lambda$ of Λ realizations of the random vector $\xi(\varpi)$ is generated. The expected value function $\mathbb{E}[Q(\mathbf{x}, \xi(\varpi))]$ can then be approximated by the sample average function $\frac{1}{\Lambda}\sum_{\lambda=1}^{\Lambda} Q(\mathbf{x}, \xi^\lambda)$. The sampled problem is therefore given by,

$$\max_{\mathbf{x}} \left\{ \sum_{\lambda=1}^{\Lambda} \frac{1}{\Lambda} Q(\mathbf{x}, \xi^\lambda) | A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0} \right\}. \tag{3.10}$$

[2]Problem (3.9) can be formulated as a mixed-integer linear program similar to the DOBSS Paruchuri et al. [2008] formulation shown in Section 2.5.2.

The sampled problem provides tighter and tighter statistical upper bound of the true problem with increasing number of samples Mak et al. [1999]; the number of samples required to solve the true problem to a certain accuracy grows linearly in the dimension of **x** Ahmed et al. [2002]. More specifically, Ahmed and Shapiro 2002 showed that if the objective function in terms of **x** is Lipschitz continuous, the sample size K which is required to solve the true problem with probability $1-\alpha$ and accuracy $\epsilon > 0$ by solving the sample average approximation problem (3.10) with accuracy $\delta < \epsilon$, grows linearly in dimension of the first stage problem (C is a constant dependent on the feasible space of **x** and the objective function):

$$K \geq \frac{12\sigma^2}{(\epsilon-\delta)^2}\left(|\mathbf{x}|\log\frac{2C}{\epsilon-\delta} - \log\alpha\right).$$

In the sampled problem, each sample ξ corresponds to a tuple $(U, V, F, \mathbf{f}, G, \mathbf{g})$. The following proposition shows that the sampled execution and observation noise can be handled by simply perturbing the utility matrices, i.e., ξ is equivalent to some $\hat{\xi}$ where $\hat{F} = \hat{G} = I_N$ and $\hat{\mathbf{f}} = \hat{\mathbf{g}} = \mathbf{0}$.

Proposition 2. *For any leader's strategy* **x** *and follower's strategy* j*, both players get the same expected utilities in two noise realizations* $(U, V, F, \mathbf{f}, G, \mathbf{g})$ *and* $(\hat{U}, \hat{V}, I_N, \mathbf{0}, I_N, \mathbf{0})$*, where,*

$$\hat{U} = \begin{pmatrix} 1 & \mathbf{f}^{\mathrm{T}} \\ \mathbf{0} & F \end{pmatrix} U, \hat{V} = \begin{pmatrix} 1 & \mathbf{g}^{\mathrm{T}} \\ \mathbf{0} & G \end{pmatrix} V.$$

Proof. We can calculate both players' expected utility vectors for both noise realizations to establish the equivalence:

$$\hat{U}^{\mathrm{T}} \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} = U^{\mathrm{T}} \begin{pmatrix} 1 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{f} & F^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} = U^{\mathrm{T}} \begin{pmatrix} 1 \\ F^{\mathrm{T}}\mathbf{x} + \mathbf{f} \end{pmatrix}.$$

$$\hat{V}^{\mathrm{T}} \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} = V^{\mathrm{T}} \begin{pmatrix} 1 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{g} & G^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} = V^{\mathrm{T}} \begin{pmatrix} 1 \\ G^{\mathrm{T}}\mathbf{x} + \mathbf{g} \end{pmatrix}. \quad \square$$

□

A direct implication of Proposition 2 is that the sampled problem (3.10) and (3.9) is equivalent to a Bayesian Stackelberg game of Λ equally weighted types, with utility matrices $(\widehat{U^{\lambda}}, \widehat{V^{\lambda}}), \lambda = 1, \ldots, \Lambda$. Hence, via sample average approximation, Hunter could be used to solve Stackelberg games with continuous payoff, execution, and observation uncertainty.

3.2.3 A Unified Approach

Both discrete and continuous uncertainty can be handled simultaneously using Hunter by applying sample average approximation in Bayesian Stackelberg games with discrete follower types. The idea is to replace each discrete follower type by a set of samples of the continuous distribution, converting the original Bayesian Stackelberg game to a larger one that can be solved by Hunter.

3.3 Experimental Results

Since none of the existing algorithm can handle both discrete and continuous uncertainty in Stackelberg games, I provide two sets of experiments in this section considering (i) only discrete uncertainty and (ii) both types of uncertainty. The utility matrices were randomly generated from a uniform distribution between -10 and 10. All experimental results were obtained on a standard 2.8GHz machine with 2GB main memory, and were averaged over 30 trials.

The main focus of the experiments in this section is to show the scalability of Hunter in comparison with existing algorithms. As described earlier, an important motivation of scaling up the number of types is in applying sample average approximation to handling continuous uncertainty. Therefore it is also interesting to see how good the solutions returned by the Hunter-based sample average approximation approach are in the presence of continuous uncertainty. I will indeed provide such experimental results in the next chapter after I introduced Recon, a robust optimization alternative aiming at providing risk-averse strategies for the defender.

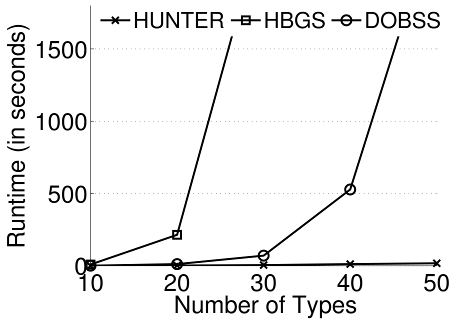
3.3.1 Handling Discrete Follower Types

For discrete uncertainty, I compared the runtime of Hunter with Dobss Paruchuri et al. [2008] and HBGS Jain et al. [2011b] (specifically, HBGS-F, the most efficient variant), the two fastest known algorithms for general Bayesian Stackelberg games. I compared the performance of these algorithms with varying number of types and varying number of pure strategies per player. The tests used a cutoff time of one hour for all three algorithms.

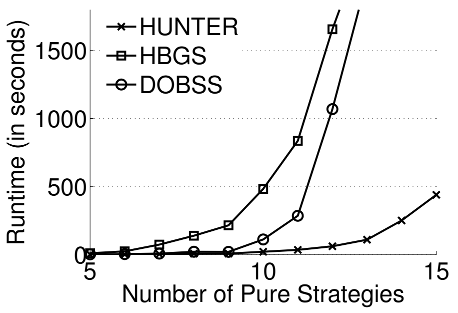
Figure 3.4(a) shows the performance of the three algorithms when the number of types increases. The games tested in this set have 5 pure strategies for each player. The x-axis shows the number of types, while the y-axis shows the runtime in seconds. As can be seen in Figure 3.4(a), HUNTER provides significant speed-up, of orders of magnitude over both HBGS and DOBSS[3](the line depicting HUNTER is almost touching the x-axis in Figure 3.4(a)). For example, HUNTER can solve a Bayesian Stackelberg game with 50 types in 17.7 seconds on average, whereas neither HBGS nor DOBSS can solve an instance in an hour. Figure 3.4(b) shows the performance of the three algorithms when the number of pure strategies for each player increases. The games tested in this set have 10 types. The x-axis shows the number of pure strategies for each player, while the y-axis shows the runtime in seconds. HUNTER again provides significant speed-up over both HBGS and DOBSS. For example, HUNTER on average can solve a game with 13 pure strategies in 108.3 seconds, but HBGS and DOBSS take more than 30 minutes.

Let us now turn to analyzing the contributions of HUNTER's key components to its performance. First, we consider the runtime of HUNTER with two search heuristics, best-first (BFS) and depth-first (DFS), when the number of types is further increased. I set the pure strategies for each player to 5, and increased the number of types from 10 to 200. In Table 3.1, I summarize the average runtime and average number of nodes explored in the search process. As we can see, DFS is faster than BFS when the number of types is small, e.g., 10 types. However, BFS always explores significantly fewer number of nodes than DFS and is more efficient when the number types is large. For games with 200 types, the average runtime of BFS based HUNTER is 20 minutes, highlighting its scalability to a large number of types. Such scalability is achieved by efficient pruning—for a game with 200 types, HUNTER explores on average 5.3×10^3 nodes with BFS and 1.1×10^4 nodes with DFS, compared to a total of $5^{200} = 6.2 \times 10^{139}$ possible leaf nodes.

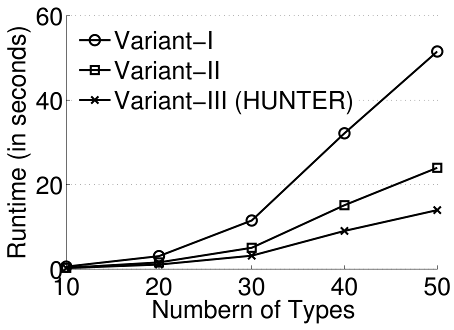
[3]The runtime results of HBGS and DOBSS are inconsistent with the results in Jain et al. [2011b] because I used CPLEX 12 for solving mixed integer linear program instead of GLPK which was used in Jain et al. [2011b].



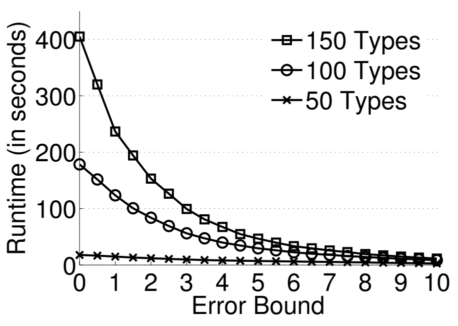
(a) Scaling up types.



(b) Scaling up pure strategies.



(c) Effectiveness of heuristics.



(d) Finding approximate solutions.

Figure 3.4: Experimental analysis of Hunter and runtime comparison against HBGS, and Dobss.

#Types	10	50	100	150	200
BFS Runtime (s)	5.7	17.7	178.4	405.1	1143.5
BFS #Nodes Explored	21	316	1596	2628	5328
DFS Runtime (s)	4.5	29.7	32.1	766.0	2323.5
DFS #Nodes Explored	33	617	3094	5468	11049

Table 3.1: Scalability of Hunter to a large number of types

Second, I tested the effectiveness of the two heuristics: inheritance of Bender's cuts from parent node to child nodes and the branching rule utilizing the solution returned by the upper bound LP. I fixed the number of pure strategies for each agent to 5 and increased the number of types from 10 to 50. In Figure 3.4(c), I show the runtime results of three variants of Hunter: i) Variant-I does not inherit Bender's cuts and chooses a random type to create branches; ii) Variant-II does not inherit Bender's cuts and uses the heuristic branching rule; iii) Variant-III (Hunter) inherits Bender's cuts and uses the heuristic branching rule. The x-axis represents the number of types while the y-axis represents the runtime in seconds. As we can see, each individual heuristic helps speed up the algorithm significantly, showing their usefulness. For example, it took 14.0 seconds to solve an instance of 50 types when both heuristics were enabled (Variant-III) compared to 51.5 seconds when neither of them was enabled (Variant-I).

Finally, let us consider the performance of Hunter in finding quality bounded approximate solutions. To this end, Hunter is allowed to terminate once the difference between the upper bound and the lower bound decreases to ϵ, a given error bound. The solution returned is therefore an approximate solution provably within η of the optimal solution. In this set of experiment, we test 30 games with 5 pure strategies for each player and 50, 100, and 150 types with varying error bound ϵ from 0 to 10. As shown in Figure 3.4(d), Hunter can effectively trade off solution quality for further speedup, indicating the effectiveness of its upper bound and lower bound heuristics. For example, for games with 100 types, Hunter returns within 30 seconds a suboptimal solution at most 5 away from the optimal solution (the average optimal solution quality is 60.2). Compared to finding the global optimal solution in 178 seconds, Hunter is able to achieve six-fold speedup by allowing at most 5 quality loss.

3.3.2 Handling Both Types of Uncertainty

In the other set of experiments, I consider Stackelberg games with both discrete and continuous uncertainty. Since no previous algorithm can handle both, I will only show the runtime results of Hunter. I tested on security games with five targets and one resource, and with multiple discrete follower types whose utilities are randomly generated. For each type, a certain number of samples from a continuous uniform distribution was drawn. Table 3.2 summarizes the runtime results of Hunter for 3, 4, 5, 6 follower types, and 10, 20 samples per type. As we can see, Hunter can efficiently handle both uncertainty simultaneously. For example, HUNTER spends less than 4 minutes on average to solve a problem with 5 follower types and 20 samples per type.

#Discrete Types	3	4	5	6
10 Samples	4.9	12.8	29.3	54.8
20 Samples	32.4	74.6	232.8	556.5

Table 3.2: Runtime results (in seconds) of HUNTER for handling both discrete and continuous uncertainty.

Chapter 4: Robust Solutions for Security Games

While Bayesian Stackelberg game model is a useful tool for modeling various types of uncertainty in security domains, the requirement of full distributional information of the uncertainty limits its applicability. The lack of precise uncertainty distribution is a particularly important challenge in security domains where historical data is scarce. This chapter focuses on security systems like ARMOR Pita et al. [2008] and considers two types of uncertainty: The defender's execution error and the attacker's observation noise. Instead of modeling the execution and observation uncertainty probabilistically like Section 3.2, I provide a robust optimization framework, called Recon (**R**isk-averse **E**xecution **C**onsidering **O**bservational **N**oise), to find risk-averse strategies for the defender. Recon assumes that nature chooses noise (of a known boundary) to maximally reduce defenders utility, and Recon maximizes against this worst case.

Section 4.1 describes the formal Recon model and the notation specific to this chapter in addition to the standard notation of security games introduced in Section 2.6. Section 4.2 provides a mixed-integer linear program (MILP) for Recon that computes the optimal risk-averse strategy and two novel heuristics that speed up the computation of Recon MILP by orders of magnitude. Finally, Section 4.3 contains the experimental results that demonstrate the superiority of Recon in uncertain domains where existing algorithms perform poorly.

4.1 Formal Model

This chapter restricts its investigation to ARMOR games which are security games with schedules of size 1 and homogeneous resources as defined earlier in Section 2.6. For ARMOR games, a strategy profile can be restricted to the form of $\langle \mathbf{x}, t_i \rangle$ where $\mathbf{x} = (x_1, \ldots, x_N)$ is a vector of probabilities of defender *coverage* over all targets and t_i is the attacker's choice of which target to attack. The sum of all coverage probabilities is not more than the number of available resources γ, i.e., $\sum_{i=1}^{N} x_i \leq \gamma$. For example, a mixed strategy for the defender can be .25 coverage on t_1 and .75 coverage on t_2. I assume y_i, the defender's actual coverage on t_i, can vary from the intended coverage x_i by the amount α_i, that is, $|y_i - x_i| \leq \alpha_i$. Thus, if $\alpha_1 = 0.1$, it would mean that $0.15 \leq y_1 \leq 0.35$. Additionally, I assume that the attacker wouldn't necessarily observe the actual implemented mixed strategy of the defender; instead the attacker's perceived coverage for t_i, denoted by z_i, can vary by β_i from the implemented coverage y_i. Therefore, $|z_i - y_i| \leq \beta_i$. Thus, in the earlier example, if y_1 was 0.3 and β_1 was set to 0.05, then $0.25 \leq z_1 \leq 0.35$. Table 4.1 summarizes notation used in this chapter.

To provide the rationale behind the uncertainty model in the context of a real world scenario, let us consider the ARMOR application at the LAX. ARMOR might generate a schedule for two canines to patrol Terminals 1, 2, 3, 4 with probabilities of 0.2, 0.8, 0.5, 0.5 respectively. However, a last-minute cargo inspection may require a canine unit to be called away from, say, Terminal 2 in its particular patrol, or an extra canine unit may become available by chance and get sent to Terminal 3. Additionally, an attacker may fail to observe a canine patrol on a terminal, or he may mistake an officer walking across as engaged in a patrol. Since each target is patrolled and observed independently, we can assume that both execution and observation noise are independent per target.

Variable	Definition
T	$T = \{t_1, \ldots, t_N\}$ is a set of N targets
μ_i^u	Defender's payoff if target t_i is uncovered
μ_i^c	Defender's payoff if target t_i is covered
ν_i^u	Attacker's payoff if target t_i is uncovered
ν_i^c	Attacker's payoff if target t_i is covered
γ	Number of defender resources
x_i	Defender's intended coverage of target t_i
y_i	Defender's actual coverage of target t_i
z_i	Attacker's observed coverage of target t_i
$\Delta\mu_i$	$\Delta\mu_i = \mu_i^c - \mu_i^u$
$\Delta\nu_i$	$\Delta\nu_i = \nu_i^u - \nu_i^c$
$D_i(x_i)$	Defender's expected utility for target t_i
	$D_i(x_i) = \mu_i^u + \Delta\mu_i x_i$
$A_i(x_i)$	Attacker's expected utility for target t_i
	$A_i(x_i) = \nu_i^u - \Delta\nu_i x_i$
α_i	Maximum execution error for target t_i
β_i	Maximum observation error for target t_i

Table 4.1: Notation for Recon

Target	μ_i^c	μ_i^u	ν_i^c	ν_i^u
t_1	10	0	-1	1
t_2	0	-10	-1	1

Figure 4.1: Example ARMOR game with two targets.

To see why SSE can be vulnerable to execution and observation noise, consider the example in Figure 4.1 with two targets, t_1 and t_2 and one defender resource. The SSE strategy for the defender would be protecting t_1 and t_2 with 0.5 probability each, making them indifferent for the attacker. The attacker breaks ties in defender's favor and chooses t_1 to attack, giving the defender an expected utility of 5. This SSE strategy is not robust to any noise – by deducting an infinitesimal amount of coverage probability from t_2, the attacker's best response changes to t_2, reducing the defender's expected utility to -5. In this case, it is better for the security agency to use a risk-averse strategy, which provides the defender the maximum worst-case expected utility. For example, assuming no execution error and 0.1 observational uncertainty ($\alpha = 0$ and $\beta = 0.1$), the optimal risk-averse defender strategy is to protect t_1 with $0.4 - \epsilon$ probability and t_2 with $0.6 + \epsilon$ probability so that even in the worst-case, the attacker would choose t_1, giving the defender an expected utility of 4. Finding the optimal risk-averse strategy for general games remains difficult, as it is essentially a *bi-level* programming problem Bard [2006].

The objective is to find the optimal risk-averse strategy $\mathbf{x}$, maximizing the worst-case defender utility, $u^*(\mathbf{x})$ (Constraint (4.1) and (4.2)). Given a fixed maximum execution and observation noise, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ respectively, $u^*(\mathbf{x})$ is computed by the minimization problem from Constraint (4.3) to (4.6).

$$\max_{\mathbf{x}} \quad u^*(\mathbf{x}) \tag{4.1}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} x_i \leq \gamma, \; 0 \leq x_i \leq 1 \tag{4.2}$$

$$u^*(\mathbf{x}) = \min_{\mathbf{y},\mathbf{z},t_j} \quad D_j(y_j) \tag{4.3}$$

$$\text{s.t.} \quad t_j \in \arg\max_{t_i \in T} A_i(z_i) \tag{4.4}$$

$$-\alpha_i \leq y_i - x_i \leq \alpha_i, \; 0 \leq y_i \leq 1 \tag{4.5}$$

$$-\beta_i \leq z_i - y_i \leq \beta_i, \;\; 0 \leq z_i \leq 1 \tag{4.6}$$

The overall problem is a bi-level programming problem. For a fixed defender strategy $\mathbf{x}$, the *second-level* problem from Constraint (4.3) to (4.6) computes the worst-case defender's executed coverage $\mathbf{y}$, the attacker's observed coverage $\mathbf{z}$, and the target attacked t_j. $(\mathbf{y}, \mathbf{z}, t_j)$ is chosen such that the defender's expected utility $D_j(y_j)$ (see Table 4.1) is minimized, given that the attacker maximizes his believed utility[1] $A_j(z_j)$ (Constraint (4.4)). This robust optimization is similar in spirit to Aghassi and Bertsimas 2006b, although that is in the context of simultaneous move games.

This also highlights the need to separately model both execution and observation noise. Indeed a problem with uncertainty defined as $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is different from a problem with $(\boldsymbol{\alpha}' = 0, \boldsymbol{\beta}' = \boldsymbol{\alpha} + \boldsymbol{\beta})$ (or vice-versa), since the defender utility is different in the two problems. Other key properties of our approach include the solution of the above problem is an SSE if $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0}$. Furthermore, a Maximin strategy is obtained when $\boldsymbol{\beta} = \mathbf{1}$ with $\boldsymbol{\alpha} = \mathbf{0}$, since $\mathbf{z}$ can be arbitrary. Finally, $\boldsymbol{\alpha} = \mathbf{1}$ implies that the execution of the defender is independent of $\mathbf{x}$ and thus, any feasible $\mathbf{x}$ is optimal.

[1]The attacker's believed utility is computed using the strategy observed by the attacker, and it may not be achieved, since $\mathbf{z}$ can be different from $\mathbf{y}$, which can be different from $\mathbf{x}$.

4.2 Approach

I will present the a mixed-integer linear programming (MILP) formulation for Recon to compute the risk-averse defender strategy in the presence of execution and observation noise. It encodes the necessary and sufficient conditions of the second-level problem (Constraint (4.4)) as linear constraints. The intuition behind these constraints is to identify $\mathcal{S}(\mathbf{x})$, the *best-response* action set for the attacker given a strategy $\mathbf{x}$, and then break ties against the defender. Additionally, Recon represents the variables $\mathbf{y}$ and $\mathbf{z}$ in terms of the variable $\mathbf{x}$ – it reduces the bi-level optimization problem to a single-level optimization problem. I will first define the term *inducible target* and then the associated necessary/sufficient conditions of the second level problem.

Definition 4. *A target t_j is said to be **weakly inducible** by a mixed strategy $\mathbf{x}$ if there exists a strategy $\mathbf{z}$ with $0 \leq z_i \leq 1$ and $|z_i - x_i| \leq \alpha_i + \beta_i$ for all $t_i \in T$, such that t_j is the best response to $\mathbf{z}$ for the attacker, i.e., $t_j = \arg\max_{t_i \in T} A_i(z_i)$.*

Additionally, I define the upper and lower bounds on the utility the attacker may believe to obtain for the strategy profile $\langle \mathbf{x}, t_i \rangle$. These bounds will then be used to determine the *best response* set $\mathcal{S}(\mathbf{x})$ of the attacker.

Definition 5. *For the strategy profile $\langle \mathbf{x}, t_i \rangle$, the upper bound of attacker's believed utility is given by $A_i^+(x_i)$, which would be reached when the attacker's observed coverage of t_i reaches the lower bound $\max\{0, x_i - \alpha_i - \beta_i\}$.*

$$A_i^+(x_i) = \min\{v_i^u, A_i(x_i - \alpha_i - \beta_i)\} \tag{4.7}$$

Similarly, denote the lower bound of attacker's believed utility of attacking target t_i by $A_i^-(x_i)$, which is reached when the attacker's observed coverage probability on t_i reaches the upper bound $\min\{1, x_i + \alpha_i + \beta_i\}$.

$$A_i^-(x_i) = \max\{\nu_i^c, A_i(x_i + \alpha_i + \beta_i)\} \tag{4.8}$$

Lemma 1. *A target t_j is weakly inducible by $\mathbf{x}$ if and only if $A_j^+(x_j) \geq \max_{t_i \in T} A_i^-(x_i)$.*

Proof. If t_j is weakly inducible, consider $\mathbf{z}$ such that $t_j = \arg\max_{t_i \in T} A_i(z_i)$. Since $z_j \geq \max\{0, x_j - \alpha_j - \beta_j\}$ and for all $t_i \neq t_j$, $z_i \leq \min\{1, x_i + \alpha_i + \beta_i\}$, we have:

$$A_j^+(x_j) = \min\{\nu_j^u, A_j(x_j - \alpha_j - \beta_j)\} \geq A_j(z_j)$$
$$\geq A_i(z_i) \geq \max\{\nu_i^c, A_i(x_i + \alpha_i + \beta_i)\} = A_i^-(x_i).$$

On the other hand, if $A_j^+(x_j) \geq A_i^-(x_i)$ for all $t_i \in T$, we can let $z_j = \max\{0, x_i - \alpha_j - \beta_j\}$ and $z_i = \min\{1, x_i + \alpha_i + \beta_i\}$ for all $t_i \neq t_j$, which satisfies $t_j = \arg\max_{t_i \in T} A_i(z_i)$. This implies t_j is weakly inducible. □

Let us also define $D_i^-(x_i)$, the lower bound on the defender's expected utility for the strategy profile $\langle \mathbf{x}, t_i \rangle$. This lower bound is used to determine the defender's worst-case expected utility.

Definition 6. *For the strategy profile $\langle \mathbf{x}, t_i \rangle$, $D_i^-(x_i)$ is achieved when the defender's implemented coverage on t_i reaches the lower bound* $\max\{0, x_i - \alpha_i\}$, *and is given by:*

$$D_i^-(x_i) = \max\{\mu_i^u, D_i(x_i - \alpha_i)\} \tag{4.9}$$

Lemma 2. *Let $\mathcal{S}(\mathbf{x})$ be the set of all targets that are weakly inducible by* $\mathbf{x}$*, then* $u^*(\mathbf{x}) = \min_{t_i \in \mathcal{S}(\mathbf{x})} D_i^-(x_i)$.

Proof. A target not in $\mathcal{S}(\mathbf{x})$ cannot be attacked, since it is not the best response of the attacker for any feasible $\mathbf{z}$. Additionally, for any target t_i in $\mathcal{S}(\mathbf{x})$, the minimum utility of the defender is $D_i^-(x_i)$. Therefore, $u^*(\mathbf{x}) \geq \min_{t_i \in \mathcal{S}(\mathbf{x})} D_i^-(x_i)$.

Additionally, we prove $u^*(\mathbf{x}) \leq \min_{t_i \in \mathcal{S}(\mathbf{x})} D_i^-(x_i)$ by showing there exist $(\mathbf{y}, \mathbf{z}, t_j)$ satisfying Constraint (4.4) to (4.6) with $D_j(y_j) = \min_{t_i \in \mathcal{S}(\mathbf{x})} D_i^-(x_i)$. To this end, we choose $t_j = \arg\min_{t_i \in \mathcal{S}(\mathbf{x})} D_i^-(x_i)$, $y_j = \max\{0, x_j - \alpha_j\}$, $z_j = \max\{0, x_j - \alpha_j - \beta_j\}$, and $y_i = \min\{1, x_i + \alpha_i\}$, $z_i = \min\{1, x_i + \alpha_i + \beta_i\}$ for all $t_i \neq t_j$. The choice of $\mathbf{y}$ and $\mathbf{z}$ here is to maximally reduce the actual and perceived coverage on t_j and maximally increase the actual and perceived coverage on all other targets $t_i \neq t_j$. By construction, $\mathbf{y}$ and $\mathbf{z}$ satisfy Constraint (4.5) and (4.6). And since t_j is weakly inducible, we have for all $t_i \neq t_j$, $A_j(z_j) = A_j^+(x_j) \geq A_i^-(x_i) = A_i(z_i)$, implying $t_j = \arg\max_{t_i \in T} A_i(z_i)$. □

Lemma (1) and (2) are the necessary and sufficient conditions for the second level optimization problem, reducing the bi-level optimization problem into a single level MILP.

4.2.1 Recon MILP

Now we present the MILP formulation for Recon. It maximizes the defender utility, denoted as u. v represents the *highest lower-bound* on the believed utility of the attacker, given in Constraint (4.11). The binary variable q_i is 1 if the target t_i is weakly inducible; it is 0 otherwise. Constraint (4.12) says that $q_i = 1$ if $A_i^+(x_i) \geq v$ (M is a large constant and ϵ is a small positive constant which together ensure that $q_i = 1$ when $A_i^+(x_i) = v$) and together with Constraint (4.11), encodes Lemma 1. The constraint that $q_i = 0$ if $A_i^+(x_i) < v$ could be added to RECON, however, it is redundant since the defender wants to set $q_i = 0$ in order to maximize u. Constraint (4.13) says that the defender utility u is less than $D_i^-(x_i)$ for all inducible targets, thereby implementing Lemma 2. Constraint (4.14) ensures that the allocated resources are no more than the number of available resources γ, maintaining feasibility.

$$\max_{\mathbf{x},\mathbf{q},u,v} \quad u \tag{4.10}$$

$$\text{s.t.} \quad v = \max_{t_i \in T} A_i^-(x_i) \tag{4.11}$$

$$A_i^+(x_i) \leq v + q_i M - \epsilon \tag{4.12}$$

$$u \leq D_i^-(x_i) + (1 - q_i)M \tag{4.13}$$

$$\sum_i x_i \leq \gamma \tag{4.14}$$

$$x_i \in [0, 1] \tag{4.15}$$

$$q_i \in \{0, 1\} \tag{4.16}$$

The max function in Constraint (4.11) can be formulated using N binary variables, $(h_1, \ldots, h_N)$, in the following manner:

$$A_i^-(x_i) \leq v \leq A_i^-(x_i) + (1 - h_i)M \tag{4.17}$$

$$\sum_{i=1}^{N} h_i = 1, \quad h_i \in \{0, 1\} \tag{4.18}$$

Constraint (4.17) ensures that $v \geq A_i^-(x_i)$ for all $1 \leq i \leq N$ and $v = A_j^-(x_j)$ when $h_j = 1$ and Constraint (4.18) ensures that only one h_j is set to 1.

The min operation in $A_i^+(x_i)$ is also implemented similarly. For example, Equation (4.7) can be encoded as:

$$v_i^u - (1 - l_i)M \leq A_i^+(x_i) \leq v_i^u$$

$$A_i(x_i - \alpha_i - \beta_i) - l_i M \leq A_i^+(x_i) \leq A_i(x_i - \alpha_i - \beta_i)$$

$$l_i \in \{0, 1\}$$

It is easy to see that $A_i^+(x_i) \leq \min\{v_i^u, A_i(x_i - \alpha_i - \beta_i)\}$. Furthermore, when $l_i = 1$, the first constraint enforces $A_i^+(x_i) = v_i^u$ and when $l_i = 1$, the second constraint enforces $A_i^+(x_i) = A_i(x_i - \alpha_i - \beta_i)$.

I will omit the details for expanding $A_i^-(x_i)$ and $D_i^-(x_i)$—they can be encoded in exactly the same fashion.

4.2.2 Speeding up

I described a MILP formulation of Recon to compute the risk-averse strategy for the defender. Solving this MILP is however computationally challenging as it involves a large number of integer variables. Using integer variables increases the complexity of the linear programming problem; indeed solving integer programs is NP-hard. MILP solvers internally use branch-and-bound to evaluate integer assignments. Availability of good lower bounds implies that less combinations of integer assignments (branch-and-bound nodes) need to be evaluated. Such lower bounds can be supplied to Recon MILP by simply adding a constraint, e.g., $u \geq u_b$ where u_b is a lower bound.

This is indeed the intuition behind speeding up the execution of RECON MILP. I will provide two methods, a-RECON and i-RECON, to generate lower bounds.

4.2.2.1 a-RECON:

a-RECON solves a *restricted* version of RECON. This restricted version has lower number of integer variables, and thus generates solutions faster. It replaces $A_i^+(x_i)$ by $A_i(x_i - \alpha_i - \beta_i)$ and $D_i^-(x_i)$ by $D_i(x_i - \alpha_i)$, thereby rewriting Constraints (4.12) and (4.13) as follows:

$$A_i(x_i - \alpha_i - \beta_i) \leq v + q_i M - \epsilon \tag{4.19}$$

$$u \leq D_i(x_i - \alpha_i) + (1 - q_i)M \tag{4.20}$$

a-RECON is indeed more *restricted* — the LHS of Constraint (4.19) in a-RECON is *no less than* the LHS of Constraint (4.12); and the RHS of Constraint (4.20) is *no greater than* the RHS of Constraint (4.13). Therefore, any solution generated by a-RECON is feasible in RECON, and acts as a lower bound.

4.2.2.2 i-RECON:

i-RECON uses an iterative method to obtain monotonically increasing lower bounds $u^{(k)}$ of RECON. Using the insight that Constraint (4.19) is *binding* only when $q_i = 0$, and (4.20) when $q_i = 1$, i-RECON rewrites Constraints (4.19) and (4.20) as follows:

$$x_i \geq \begin{cases} \rho_{a,i}(v) = \frac{v_i^u - v + \epsilon}{\Delta v_i} + \alpha_i + \beta_i & \text{if } q_i = 0 \\ \rho_{d,i}(u) = \frac{u - \mu_i^u}{\Delta \mu_i)} + \alpha_i & \text{if } q_i = 1 \end{cases} \tag{4.21}$$

Constraint (4.21) says that $q_i = 0$ implies $x_i \geq \rho_{a,i}(v)$ and $q_i = 1$ implies $x_i \geq \rho_{d,i}(u)$.[2] Constraint (4.21) is equivalent to:

$$\begin{aligned} x_i \geq \ & \min\{\rho_{d,i}(u), \rho_{a,i}(v)\} \\ = \ & \rho_{d,i}(u) + \min\{0, \rho_{a,i}(v) - \rho_{d,i}(u)\} \end{aligned} \tag{4.22}$$

The equivalence between Constraint (4.21) and (4.22) can be verified as follows: $(\mathbf{x}, u, v)$ from any feasible solution $(\mathbf{x}, \mathbf{q}, u, v)$ of (4.21) is trivially feasible in (4.22). On the other hand, given a feasible solution $(\mathbf{x}, u, v)$ to Constraint (4.22), we choose $q_i = 1$ if $x_i \geq \rho_{d,i}(u)$ and 0 otherwise, and thus obtain a feasible solution to Constraint (4.21). Hence, an equivalent problem of a-Recon can be obtained by replacing Constraints (4.12) and (4.13) by Constraint (4.22). In the kth iteration, i-Recon substitutes $\rho_{d,i}(u) - \rho_{a,i}(v)$ by a constant, $\Delta\rho_i^{(k)}$, *restricting* Constraint (4.22). This value is updated in every iteration while maintaining a restriction of Constraint (4.22). Such a substitution reduces Constraint (4.22) to a linear constraint, implying that i-Recon performs a polynomial-time computation in every iteration.[3]

Observe that $\rho_{d,i}(u)$ is increasing in u where as $\rho_{a,i}(v)$ is decreasing in v (refer Constraint (4.21)), and hence $\rho_{d,i}(u) - \rho_{a,i}(v)$ is *increasing* in both u and v. i-Recon generates an increasing sequence of $\{\Delta\rho_i^{(k)} = \rho_{d,i}(u^{(k)}) - \rho_{a,i}(v^{(k)})\}$ by finding increasing sequences of $u^{(k)}$ and $v^{(k)}$. As I will show later, substituting $\rho_{d,i}(u) - \rho_{a,i}(v)$ with $\{\Delta\rho_i^{(k)}\}$ in Constraint (4.22) guarantees the correctness. Since a higher value of $\Delta\rho_i^{(k)}$ implies a lower value of $\min\{0, -\Delta\rho_i^{(k)}\}$, a weaker restriction is imposed by Constraint (4.22), leading to a better lower bound $u^{(k+1)}$.

[2]This is *not* equivalent to the unconditional equation $x_i \geq \max\{\rho_{a,i}(v), \rho_{d,i}(u)\}$.

[3]While the formulation has integer variables from Constraint (4.11), it can be considered as $2N$ LPs since there are only $2N$ distinct combinations of integer assignments.

Algorithm 2: Pseudo code of i-Recon

```
1 $k = 0, u^{(0)} = v^{(0)} = -\infty$;
2 while $|v^{(k+1)} - v^{(k)}| \leq \eta$ and $|u^{(k+1)} - u^{(k)}| \leq \eta$ do
3     $v^{(k+1)}$ = Solve(A-LP $(u^{(k)}, v^{(k)})$);
4     $u^{(k+1)}$ = Solve(D-LP $(u^{(k)}, v^{(k)})$);
5     $k = k + 1$;
6 end
```

Given $u^{(k)}$ and $v^{(k)}$, i-Recon uses `D-LP` to compute the $u^{(k+1)}$, and `A-LP` to compute $v^{(k+1)}$. The pseudo-code for i-Recon is given in Algorithm 2. `D-LP` is the following maximization linear program, which returns the solution vector $(\mathbf{x}, u, \hat{v})$, such that u is the desired lower bound.

$$\max_{\mathbf{x},u,\hat{v}} \ u$$

$$\text{s.t. Constraint(4.11), (4.14) and (4.15)}$$

$$x_i \geq \rho_{d,i}(u) + \min\{0, -\Delta\rho_i^{(k)}\} \tag{4.23}$$

$$u \geq u^{(k)}; \quad \hat{v} \geq v^{(k)} \tag{4.24}$$

Constraint (4.24) is added to `D-LP` to ensure that we get a monotonically increasing solution in every iteration. Similarly, given $u^{(k)}$ and $v^{(k)}$, `A-LP` is the following minimization problem. It minimizes v to guarantee that Constraint (4.23) in `D-LP` remains a restriction to Constraint (4.22) for the next iteration, ensuring D-LP always provides a lower bound of Recon. More detail is given in Proposition 3 which proves the correctness of i-Recon.

$$\min_{\mathbf{x},u,v} \ v$$

$$\text{s.t. Constraint (4.11), (4.14) and (4.15)}$$

$$x_i \geq \rho_{a,i}(v) + \min\{\Delta\rho_i^{(k)}, 0\} \tag{4.25}$$

$$v \geq v^{(k)} \tag{4.26}$$

Proposition 3. *Both* D-LP *and* A-LP *are feasible and bounded for every iteration k until i-*Recon *converges.*

Proof. A-LP is bounded for every iteration because $v \geq \max_{t_i \in T} v_i^c$ by Constraint (4.11). I will prove the rest of the proposition using induction. First I establish that both D-LP and A-LP are feasible and bounded in the first iteration. In the first iteration, D-LP is feasible for any value of $x_i \geq 0$ when $u = \min_{t_i \in T}\{\mu_i^u - \alpha_i \Delta\mu_i\}$ (from Constraint (4.21)), and it is bounded since $\rho_{d,i}(u) \leq x_i \leq 1$ for all $t_i \in T$. In the same way, for A-LP, Constraint (4.25) becomes $x_i \geq -\infty$ in the first iteration. Thus, $v = \max_{t_i \in T} A_i^-(x_i) > -\infty$ is a feasible solution.

Assuming that D-LP and A-LP are feasible and bounded for iterations $1, 2, \ldots, k$, I now show that they remain bounded and feasible in iteration $k + 1$. Firstly, D-LP is bounded in the $k + 1^{\text{th}}$ iteration since $\rho_{d,i}(u) \leq 1 - \min\{0, -\Delta\rho_i^{(k)}\}$ for all $t_i \in T$. D-LP is feasible because the solution from the k^{th} iteration, $(\mathbf{x}^{(k)}, u^{(k)}, \hat{v}^{(k)})$, remains feasible. To see this, observe that since $\rho_{d,i}^{(k)}$ is increasing and $\rho_{a,i}^{(k)}$ is decreasing with k, thus we have $\Delta\rho_i^{(k)} \geq \Delta\rho_i^{(k-1)}$. Hence $\min\{0, -\Delta\rho_i^{(k-1)}\} \geq \min\{0, -\Delta\rho_i^{(k)}\}$, implying that $(\mathbf{x}^{(k)}, u^{(k)})$ satisfies Constraint (4.23). Moreover, Constraints (4.11), (4.14), (4.15) and (4.24) are trivially satisfied.

Similarly, A-LP is also feasible in the $k+1^{th}$ iteration since $\langle \mathbf{x}^{(k+1)}, u^{(k+1)}, \hat{v}^{(k+1)} \rangle$, the solution returned by D-LP in the $k+1^{th}$ iteration, satisfies all the constraints of A-LP. Firstly, Constraints (4.11), (4.14), (4.15) and (4.26) are trivially satisfied. Secondly, Constraint (4.25) is also satisfied since:

$$\rho_{d,i}(u^{(k+1)}) - \rho_{a,i}(\hat{v}^{(k+1)}) \geq \Delta\rho_i^{(k)}. \tag{4.27}$$

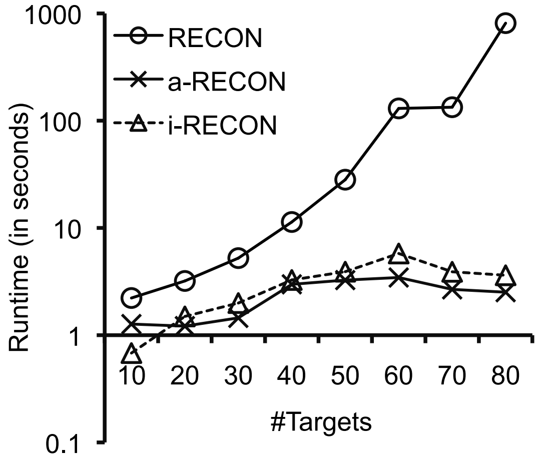
$$\begin{aligned}
x_i^{(k+1)} &\geq \rho_{d,i}(u^{(k+1)}) + \min\{0, -\Delta\rho_i^{(k)}\} && \text{from (4.23)} \\
&= \min\{\rho_{d,i}(u^{(k+1)}), \rho_{d,i}(u^{(k+1)}) - \Delta\rho_i^{(k)}\} \\
&\geq \min\{\rho_{d,i}(u^{(k+1)}), \rho_{a,i}(\hat{v}_a^{(k+1)})\} && \text{from (4.27)} \\
&= \rho_{a,i}(\hat{v}_a^{(k+1)}) + \min\{\rho_{d,i}(u^{(k+1)}) - \rho_{a,i}(\hat{v}_a^{(k+1)}), 0\} \\
&\geq \rho_{a,i}(\hat{v}_a^{(k+1)}) + \min\{\Delta\rho_i^{(k)}, 0\} && \text{from (4.27)}
\end{aligned}$$

Similarly, $(\mathbf{x}^{(k+1)}, u^{(k+1)}, \hat{v}^{(k+1)})$ is a feasible solution of a-Recon for any k using inequality (4.27), and hence, $u^{(k+1)}$ is a lower bound of Recon. Additionally, since the sequence $\{u^{(k)}\}$ is bounded and monotonically increasing, it must converge. □

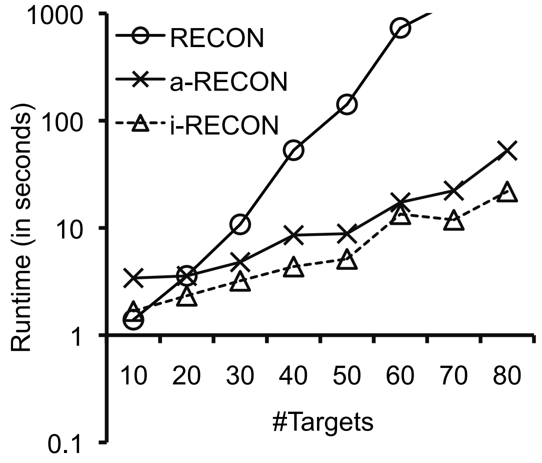
4.3 Experimental Results

I provide two sets of experimental results: (i) I provide the runtime results of Recon, showing the effectiveness of the two heuristics a-Recon and i-Recon. (ii) I compare the solution quality of strategies generated by Eraser, Cobra, Hunter, and Recon, under execution and observation uncertainty.

4.3.1 Runtime of Recon



(a) Runtime of Recon with $\alpha = \beta = 0.01$.



(b) Runtime of Recon with $\alpha = \beta = 0.1$.

Figure 4.2: Runtime of Recon MILP and the speedup of lower bound heuristics a-Recon and i-Recon.

In this set of experiments, I show the runtime of the three variants of Recon with increasing number of targets. In all test instances, I set the number of defender resources to 20% of the number of targets. The results were obtained using CPLEX on a standard 2.8GHz machine with 2GB main memory, and averaged over 30 trials. Figures 4.2(a) and 4.2(b) show the runtime results of Recon without any lower bounds, and with lower bounds provided by a-Recon and i-Recon respectively. The x-axis shows the number of targets and the y-axis (in logarithmic scale) shows the total runtime in seconds. Both a-Recon and i-Recon heuristics help reduce the total runtime significantly in both uncertainty settings—the speedup is of orders of magnitude in games with large number of targets. For instance, for cases with 80 targets and high uncertainty, Recon without heuristic lower bounds takes $3,948$ seconds, whereas Recon with a-Recon lower bound takes a total runtime of 52 seconds and Recon with i-Recon lower bound takes a total runtime of 22 seconds.

4.3.2 Performance under uncertainty

In this set of experiments, I compared the performance of various candidate strategies under continuous execution and observation uncertainty. Two scenarios of the ARMOR security games were considered: (i) games with 5 targets and 2 defender resources and (ii) games with 8 targets and 3 defender resources. Payoffs μ_i^c and ν_i^u are integers chosen uniformly randomly from 1 to 10 while μ_i^u and ν_i^c are integers chosen uniformly randomly from -10 to -1.

Let us define parameterized uncertainty distributions Φ_ρ as simplified examples of continuous execution and observation uncertainty. In ARMOR security games with uncertainty distribution Φ_ρ, the defender's execution and the attacker's observation at every target follows independent uniform distributions determined by ρ. More specifically, given an intended defender strategy $\mathbf{x} = (x_1, \ldots, x_N)$, where x_i represents the probability of protecting target i, the actual executed strategy $\mathbf{y} = (y_1, \ldots, y_N)$ has every y_i following a uniform distribution between $x_i - \rho$ and $x_i + \rho$ and the actual observed strategy $\mathbf{z} = (z_1, \ldots, z_N)$ has every z_i following a uniform distribution between $y_i - \rho$ and $y_i + \rho$. Here ρ is referred to as the uncertainty parameter and a higher ρ implies a higher amount of uncertainty.

The following candidate strategies provided were compared:

- Eraser: defender's SSE strategy computed by the Eraser algorithm Kiekintveld et al. [2009].

- Cobra-1: defender strategy generated by Cobra Pita et al. [2010], the latest algorithms that addresses attacker's observational error, with bounded rationality parameter ϵ set to 1.0.[4]

- Cobra-2: defender strategy generated by Cobra with $\epsilon = 2.0$ (as suggested in Pita et al. [2010]).

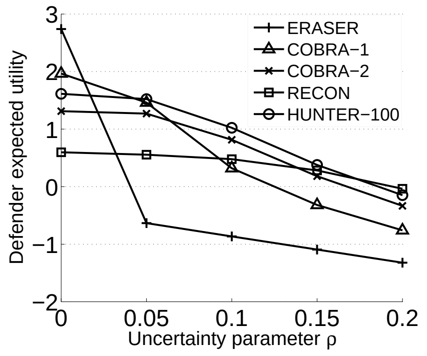
- Hunter-100: defender strategy generated by Hunter-based sample average approximation with 100 samples (see Section 3.2). The samples of uncertainty realization were drawn randomly from distribution $\Phi_{0.1}$.

- Recon: defender strategy generated by the Recon MILP with $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0.1}$, i.e., the maximum execution and observation noise for every target was set to 0.1.

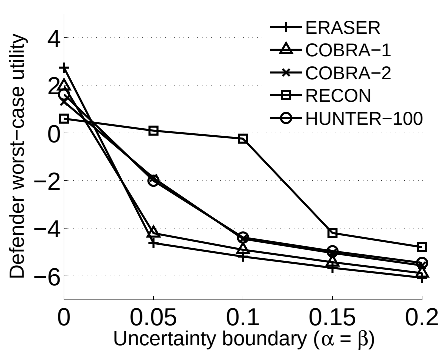
To understand the performance of the candidates strategies above, two major metrics were employed for a given uncertainty parameter ρ: (i) expected defender utility under uncertainty distribution Φ_ρ and (ii) worst-case defender utility given a maximum execution error $\boldsymbol{\alpha}$ and a maximum observation error $\boldsymbol{\beta}$ where $\alpha_i = \beta_i = \rho$. The expected defender utility was computed by evaluating the strategy for $10,000$ sample uncertainty realizations and taking the average. The expected defender utility is a valuable metric to evaluate a strategy's performance when the uncertainty distribution follows some smooth and continuous distribution (here independent uniform distributions Φ_ρ). The worst-case defender utility was computed using the second-level optimization problem given in Constraints (4.3) to (4.6). The worst-case defender utility is also an important metric which determines how robust a strategy is given some uncertainty boundary

[4]The human bias parameter in Cobra is set to 1 since the experiments here are not tested against human subjects.

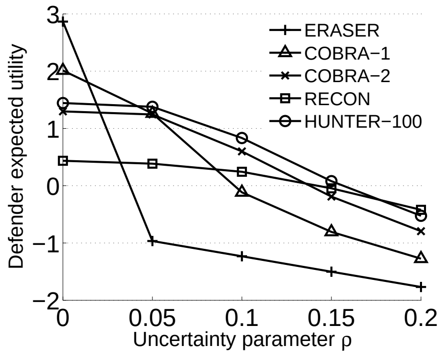
(here a hyper-rectangle boundary defined by ρ). In the experimental results reported in Figure 4.3, for each strategy and each metric, 5 uncertainty settings were evaluated: $\rho = 0, 0.05, 0.1, 0.15, 0.2$.



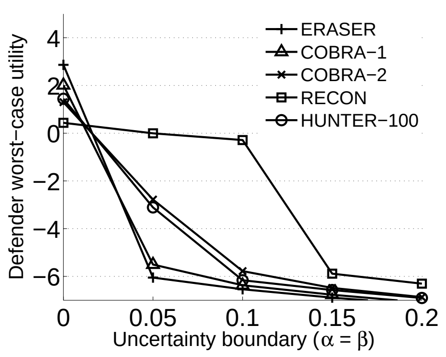
(a) Expected defender utility with increasing uncertainty (games with 5 targets and 2 defender resources).



(b) Worst defender utility with increasing uncertainty (games with 5 targets and 2 defender resources).



(c) Expected defender utility with increasing uncertainty (games with 8 targets and 3 defender resources).



(d) Worst defender utility with increasing uncertainty (games with 8 targets and 3 defender resources).

Figure 4.3: Performance of strategies generated by RECON, HUNTER, ERASER, and COBRA.

Figure 4.3(a) and Figure 4.3(b) show the expected defender utility and the worst-case defender utility with increasing uncertainty for ARMOR games with 5 targets and 2 defender resources. Here the x-axis represents the value of ρ and the y-axis represent the defender's utility (either expected or worst-case). Figure 4.3(c) and Figure 4.3(d) show exactly the same comparison but for ARMOR games with 8 targets and 3 defender resources. As we can see, the trends observed in the two scenarios were consistent. All the comparisons claimed below between two candidate strategies were statistically significant with p-value under 0.05. The take away messages from Figure 4.3 are:

- The SSE strategy computed by Eraser performs poorly in the presence of execution and observation uncertainty in terms of both expected and worst-case utility metrics. A small amount of noise $\rho = 0.05$ was sufficient to lower the expected utility of Eraser from 2.73 to -0.63 in the 5-target scenario and from 2.86 to -0.96 in the 8-target scenario. Indeed, when there was non-zero uncertainty, Eraser was consistently outperformed by other candidate strategies in both expected and worst-case utility metrics.

- Comparing Cobra-1 and Cobra-2, we can see the ϵ parameter in Cobra offers tradeoff between expected utility and robustness to noise. Cobra-1 had higher expected utility than Cobra-2 when there was low uncertainty but degraded faster than Cobra-2 when the amount of uncertainty increased.

- When the true uncertainty distribution was close to the distribution used in generating the strategy ($\Phi_{0.05}, \Phi_{0.1}, \Phi_{0.15}$), Hunter-based sample average approximation provided the best expected utility consistently in both the 5-target and the 8-target scenarios. This suggests that Hunter performs well even when the modeled uncertainty distribution ($\Phi_{0.1}$) is different from the actual uncertainty distribution.

- When the true uncertainty distribution was drastically different from the modeled uncertainty distribution, e.g., when the true uncertainty distribution was Φ_0 and $\Phi_{0.2}$, Hunter-100 was outperformed by other candidate strategies. It therefore is valuable to obtain good estimate of the true uncertainty distribution in order for Hunter-based sample average approximation approach to work well in practice.

- In the presence of uncertainty, Recon was consistently the best performer in terms of worst-case utility. This is indeed the motivation of Recon—being able to provide guarantees on the defender's utility is extremely valuable in situations where precise uncertainty distribution is unavailable. It worths noting that the worst-case utility of Recon can still be very bad when the actual uncertainty realization can exceed the estimated boundary. For example, when uncertainty boundary increased from $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0.1}$ to $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0.15}$, the worst-case utility of Recon dropped from -0.24 to -4.20 and from -0.29 to -5.89 for 5-target and 8-target scenarios respectively (although Recon was still better than other candidate strategies when the uncertainty boundary was $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0.15}$).

- Recon was outperformed by Hunter-100 and the variants of Cobra in terms of expected utility when the uncertainty was low (Φ_ρ when $\rho \leq 0.1$). This implies that the Recon-generated strategies can be overly conservative when the uncertainty boundary used was too loose. For example, when the true uncertainty distribution is $\Phi_{0.1}$, the Recon strategy assuming an uncertainty boundary of $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0.1}$ is too conservative to generate good expected utility.

Chapter 5: Stackelberg vs. Nash in Security Games

A key element of the Stackelberg paradigm is the concept of leadership, which naturally defines a party of the game as the leader who commits to a possibly randomized strategy whereas the other party acts as the follower who attempts to observe the leader's strategy. In previous chapters, despite the fact that the follower's observation is possibly noisy, this leadership paradigm is always taken for granted. However, there are legitimate concerns about whether the Stackelberg model is appropriate in all cases. In some situations attackers may choose to act without acquiring costly information about the security strategy, especially if security measures are difficult to observe (e.g., undercover officers) and insiders are unavailable. In such cases, a simultaneous-move game model may be a better reflection of the real situation. The defender faces an unclear choice about which strategy to adopt: the recommendation of the Stackelberg model (SSE strategy), or of the simultaneous-move model (NE strategy), or something else entirely? Recall the example given in Figure 2.2, the equilibrium strategy can in fact differ between these models.

In this chapter I will provide theoretical and experimental analysis of the leader's dilemma, focusing on *security games* defined in Chapter 2. Section 5.1 characterizes a set of key properties of security games. In particular, I show that when the security games satisfy the *SSAS* (**S**ubsets of **S**chedules **A**re **S**chedules) property, the defender's SSE strategy is also an NE strategy. In this case, the defender is always playing a best response by using an SSE regardless of whether the attacker can observe or not. Section 5.2 shows that this property no longer holds when the attacker can attack multiple targets. Section 5.3 contains experimental results.

5.1 Properties of Security Games

The challenge faced here is to understand the fundamental relationships between the SSE and NE strategies in security games. A special case is zero-sum security games, where the defender's utility is the exact opposite of the attacker's utility. For finite two-person zero-sum games, it is known that the different game theoretic solution concepts of NE, minimax, maximin and SSE all give the same answer. In addition, Nash equilibrium strategies of zero-sum games have a very useful property in that they are *interchangeable*: an equilibrium strategy for one player can be paired with the other player's strategy from *any* equilibrium profile, and the result is an equilibrium, and the payoffs for both players remain the same.

Unfortunately, security games are not necessarily zero-sum (and are not zero-sum in deployed applications). Many properties of zero-sum games do not hold in security games. For instance, a minimax strategy in a security game may not be a maximin strategy. Consider the example in Table 5.1, in which there are 3 targets and one defender resource. The defender has three actions; each of defender's actions can only cover one target at a time, leaving the other targets uncovered. While all three targets are equally appealing to the attacker, the defender has varying utilities of capturing the attacker at different targets. For the defender, the unique minimax strategy, $(1/3, 1/3, 1/3)$, is different from the unique maximin strategy, $(6/11, 3/11, 2/11)$.

	t$_1$		**t$_2$**		**t$_3$**	
	Cov.	Unc.	Cov.	Unc.	Cov.	Unc.
Defender	1	0	2	0	3	0
Attacker	0	1	0	1	0	1

Table 5.1: Security game which is not strategically zero-sum

Strategically zero-sum games Moulin and Vial [1978] are a natural and strict superset of zero-sum games for which most of the desirable properties of zero-sum games still hold. This is exactly the class of games for which no completely mixed Nash equilibrium can be improved upon. Moulin and Vial proved a game (A, B) is strategically zero-sum if and only if there exist $\alpha > 0$ and $\beta > 0$ such that $\alpha A + \beta B = I_c + I_r$, where I_c is a matrix with identical columns and I_r is a matrix with identical rows Moulin and Vial [1978]. Unfortunately, security games are not even strategically zero-sum. The game in Table 5.1 is a counterexample, because otherwise there must exist $\alpha, \beta > 0$ such that,

$$\alpha \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix} + \beta \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} c_1 & c_1 & c_1 \\ c_2 & c_2 & c_2 \\ c_3 & c_3 & c_3 \end{pmatrix} + \begin{pmatrix} r_1 & r_2 & r_3 \\ r_1 & r_2 & r_3 \\ r_1 & r_2 & r_3 \end{pmatrix}$$

From these equations, $c_1 + r_2 = c_1 + r_3 = c_2 + r_1 = c_2 + r_3 = c_3 + r_1 = c_3 + r_2 = \beta$, which implies $r_1 = r_2 = r_3$ and $c_1 = c_2 = c_3$. We also know $c_1 + r_1 = \alpha$, $c_2 + r_2 = 2\alpha$, $c_3 + r_3 = 3\alpha$. However since $c_1 + r_1 = c_2 + r_2 = c_3 + r_3$, α must be 0, which contradicts the assumption $\alpha > 0$.

Nevertheless, I will show in the rest of this section that security games still have some important properties. I will start by establishing equivalence between the set of defender's minimax strategies and the set of defender's NE strategies. Second, I will show Nash equilibria in security games are interchangeable, resolving the defender's equilibrium strategy selection problem in simultaneous-move games. Third, I will show that under a natural restriction on schedules, any SSE strategy for the defender is also a minimax strategy and hence an NE strategy. This resolves the defender's dilemma about whether to play according to SSE or NE when there is uncertainty about attacker's ability to observe the strategy. Finally, for a restricted class of games (ARMOR games), there is a unique SSE/NE defender strategy and a unique attacker NE strategy.

5.1.1 Equivalence of Nash Equilibrium and Minimax

Recall the definition and notation provided in Section 2.6. In this section, I will first prove that any defender's NE strategy is also a minimax strategy. Then for every defender's minimax strategy $\mathbf{X}$ we construct a strategy $\mathbf{a}$ for the attacker such that $\langle \mathbf{X}, \mathbf{a} \rangle$ is an NE profile.

Definition 7. *For a defender's mixed strategy* $\mathbf{X}$*, define the attacker's best response utility by* $E(\mathbf{X}) = \max_{i=1}^{N} v(\mathbf{X}, t_i)$*. Denote the minimum of the attacker's best response utilities over all defender's strategies by* $E^* = \min_{\mathbf{X}} E(\mathbf{X})$*. The set of defender's minimax strategies is defined as:*

$$\Omega_M = \{\mathbf{X} | E(\mathbf{X}) = E^*\}.$$

Define the function f as follows. If $\mathbf{a}$ is an attacker's strategy in which target t_i is attacked with probability a_i, then $f(\mathbf{a}) = \bar{\mathbf{a}}$ is an attacker's strategy such that

$$\bar{a}_i = \lambda a_i \frac{\Delta \mu_i}{\Delta \nu_i},$$

where $\lambda > 0$ is a normalizing constant such that $\sum_{i=1}^{N} \bar{a}_i = 1$. The inverse function $f^{-1}(\bar{\mathbf{a}}) = \mathbf{a}$ is given by the following equation.

$$a_i = \frac{1}{\lambda} \bar{a}_i \frac{\Delta \nu_i}{\Delta \mu_i} \tag{5.1}$$

Lemma 3. *Consider a security game $\mathcal{G}$. Construct the corresponding zero-sum security game $\bar{\mathcal{G}}$ in which the defender's utilities are re-defined as follows.*

$$\mu_i^c = -\nu_i^c,\ \mu_i^u = -\nu_i^u,\ \ \forall i = 1, \ldots, N$$

Then $\langle \mathbf{X}, \mathbf{a} \rangle$ is an NE profile in $\mathcal{G}$ if and only if $\langle \mathbf{X}, f(\mathbf{a}) \rangle$ is an NE profile in $\bar{\mathcal{G}}$.

Proof. Note that the supports of strategies $\mathbf{a}$ and $\bar{\mathbf{a}}$ are the same, and also that the attacker's utility function is the same in games $\mathcal{G}$ and $\bar{\mathcal{G}}$. Thus $\mathbf{a}$ is a best response to $\mathbf{X}$ in $\mathcal{G}$ if and only if $\bar{\mathbf{a}}$ is a best response to $\mathbf{X}$ in $\bar{\mathcal{G}}$.

Denote the utility that the defender gets if profile $\langle \mathbf{X}, \mathbf{a} \rangle$ is played in game $\mathcal{G}$ by $u^{\mathcal{G}}(\mathbf{X}, \mathbf{a})$. To show that $\mathbf{X}$ is a best response to $\mathbf{a}$ in game $\mathcal{G}$ if and only if $\mathbf{X}$ is a best response to $\bar{\mathbf{a}}$ in $\bar{\mathcal{G}}$, it is sufficient to show equivalence of the following two inequalities.

$$u^{\mathcal{G}}(\mathbf{X}, \mathbf{a}) - u^{\mathcal{G}}(\mathbf{X}', \mathbf{a}) \geq 0 \ \Leftrightarrow\ u^{\bar{\mathcal{G}}}(\mathbf{X}, \bar{\mathbf{a}}) - u^{\bar{\mathcal{G}}}(\mathbf{X}', \bar{\mathbf{a}}) \geq 0$$

I will prove the equivalence by starting from the first inequality and transforming it into the second one. On the one hand, from Equation (2.4) we have,

$$u^{\mathcal{G}}(\mathbf{X}, \mathbf{a}) - u^{\mathcal{G}}(\mathbf{X}', \mathbf{a}) = \sum_{i=1}^{N} a_i (x_i - x_i') \Delta \mu_i.$$

Similarly, on the other hand, we have,

$$u^{\bar{\mathcal{G}}}(\mathbf{X}, \bar{\mathbf{a}}) - u^{\bar{\mathcal{G}}}(\mathbf{X}', \bar{\mathbf{a}}) = \sum_{i=1}^{N} \bar{a}_i (x_i - x_i') \Delta \nu_i.$$

Given Equation (5.1) and $\lambda > 0$, we have,

$$\begin{aligned} & u^{\mathcal{G}}(\mathbf{X}, \mathbf{a}) - u^{\mathcal{G}}(\mathbf{X}', \mathbf{a}) \geq 0 \\ \Leftrightarrow \ & \sum_{i=1}^{N} a_i (x_i - x_i') \Delta \mu_i \geq 0 \Leftrightarrow \ \sum_{i=1}^{N} \frac{1}{\lambda} \bar{a}_i \frac{\Delta \nu_i}{\Delta \mu_i} (x_i - x_i') \Delta \mu_i \geq 0 \\ \Leftrightarrow \ & \frac{1}{\lambda} \sum_{i=1}^{N} \bar{a}_i (x_i - x_i') \Delta \nu_i \geq 0 \Leftrightarrow \ \frac{1}{\lambda} \left(u^{\bar{\mathcal{G}}}(\mathbf{X}, \bar{\mathbf{a}}) - u^{\bar{\mathcal{G}}}(\mathbf{X}', \bar{\mathbf{a}}) \right) \geq 0 \\ \Leftrightarrow \ & u^{\bar{\mathcal{G}}}(\mathbf{X}, \bar{\mathbf{a}}) - u^{\bar{\mathcal{G}}}(\mathbf{X}', \bar{\mathbf{a}}) \geq 0 \end{aligned}$$

□

Lemma 4. *Suppose* $\mathbf{X}$ *is a defender NE strategy in a security game. Then* $E(\mathbf{X}) = E^*$*, i.e.,* $\Omega_{NE} \subseteq \Omega_M$.

Proof. Suppose $\langle \mathbf{X}, \mathbf{a} \rangle$ is an NE profile in the security game $\mathcal{G}$. According to Lemma 3, $\langle \mathbf{X}, f(\mathbf{a}) \rangle$ must be an NE profile in the corresponding zero-sum security game $\bar{\mathcal{G}}$. Since $\mathbf{X}$ is an NE strategy in a zero-sum game, it must also be a minimax strategy Fudenberg and Tirole [1991]. Thus $E(\mathbf{X}) = E^*$. □

Lemma 5. *In a security game* $\mathcal{G}$*, any defender's strategy* $\mathbf{X}$ *such that* $E(\mathbf{X}) = E^*$ *is an NE strategy, i.e.,* $\Omega_M \subseteq \Omega_{NE}$.

Proof. $\mathbf{X}$ is a minimax strategy in both $\mathcal{G}$ and the corresponding zero-sum game $\bar{\mathcal{G}}$. Any minimax strategy is also an NE strategy in a zero-sum game Fudenberg and Tirole [1991]. Then there must exist an NE profile $\langle \mathbf{X}, \bar{\mathbf{a}} \rangle$ in $\bar{\mathcal{G}}$. By Lemma 3, $\langle \mathbf{X}, f^{-1}(\bar{\mathbf{a}}) \rangle$ is an NE profile in $\mathcal{G}$. Thus $\mathbf{X}$ is an NE strategy in $\mathcal{G}$. □

Theorem 5.1.1. *In a security game, the set of defender's minimax strategies is equal to the set of defender's NE strategies, i.e.,* $\Omega_M = \Omega_{NE}$.

Proof. Lemma 4 shows that every defender's NE strategy is a minimax strategy, and Lemma 5 shows that every defender's minimax strategy is an NE strategy. Thus the sets of defender's NE and minimax strategies must be equal. □

5.1.2 Interchangeability of Nash Equilibria

I show that Nash Equilibria in security games are interchangeable.

Theorem 5.1.2. *Suppose* $\langle \mathbf{X}, \mathbf{a} \rangle$ *and* $\langle \mathbf{X}', \mathbf{a}' \rangle$ *are two NE profiles in a security game* $\mathcal{G}$. *Then* $\langle \mathbf{X}, \mathbf{a}' \rangle$ *and* $\langle \mathbf{X}', \mathbf{a} \rangle$ *are also NE profiles in* $\mathcal{G}$.

Proof. Consider the corresponding zero-sum game $\bar{\mathcal{G}}$. From Lemma 3, both $\langle \mathbf{X}, f(\mathbf{a}) \rangle$ and $\langle \mathbf{X}', f(\mathbf{a}') \rangle$ must be NE profiles in $\bar{\mathcal{G}}$. By the interchange property of NE in zero-sum games Fudenberg and Tirole [1991], $\langle \mathbf{X}, f(\mathbf{a}') \rangle$ and $\langle \mathbf{X}', f(\mathbf{a}) \rangle$ must also be NE profiles in $\bar{\mathcal{G}}$. Applying Lemma 3 again in the other direction, we get that $\langle \mathbf{X}, \mathbf{a}' \rangle$ and $\langle \mathbf{X}', \mathbf{a} \rangle$ must be NE profiles in $\mathcal{G}$. □

By Theorem 5.1.2, the defender's equilibrium selection problem in a simultaneous-move security game is resolved. The reason is that given the attacker's NE strategy $\mathbf{a}$, the defender must get the same utility by responding with any NE strategy. Next, I will provide some additional insights on the expected utilities of both players when some NE profile is played. In particular, I will first show the attacker's expected utility is the same in all NE profiles. However, the defender may have varying expected utilities corresponding to different attacker's strategies as demonstrated by an example.

Theorem 5.1.3. *Suppose $\langle \mathbf{X}, \mathbf{a} \rangle$ is an NE profile in a security game. Then, $v(\mathbf{X}, \mathbf{a}) = E^*$.*

Proof. From Lemma 4, $\mathbf{X}$ is a minimax strategy and $E(\mathbf{X}) = E^*$. On the one hand,

$$v(\mathbf{X}, \mathbf{a}) = \sum_{i=1}^{N} a_i v(\mathbf{X}, t_i) \leq \sum_{i=1}^{N} a_i E(\mathbf{X}) = E^*.$$

On the other hand, because $\mathbf{a}$ is a best response to $\mathbf{X}$, it should be at least as good as the strategy of attacking $t^* \in \arg\max_t v(\mathbf{X}, t)$ with probability 1, that is,

$$v(\mathbf{X}, \mathbf{a}) \geq v(\mathbf{X}, t^*) = E(\mathbf{X}) = E^*.$$

Therefore we know $v(\mathbf{X}, \mathbf{a}) = E^*$. □

Unlike the attacker who gets the same utility in all NE profiles, the defender may get varying expected utilities depending on the attacker's strategy selection as shown in Example 1.

	$\mathbf{t_1}$		$\mathbf{t_2}$	
	Cov.	Unc.	Cov.	Unc.
Defender	1	0	2	0
Attacker	1	2	0	1

Figure 5.1: A security game where the defender's expected utility varies in different NE profiles

Example 1. *Consider the game shown in Figure 5.1. The defender can choose to cover one of the two targets at a time. The only defender's NE strategy is to cover t_1 with 100% probability, making the attacker indifferent between attacking t_1 and t_2. One attacker's NE response is always* *attacking t_1, which gives the defender an expected utility of* 1*. Another attacker's NE strategy is* $(2/3, 1/3)$*, given which the defender is indifferent between defending t_1 and t_2. In this case, the defender's utility decreases to* $2/3$ *because she captures the attacker with a lower probability.*

5.1.3 SSE and Minimax / NE

We have already shown that the set of defender's NE strategies coincides with her minimax strategies. If every defender's SSE strategy is also a minimax strategy, then SSE strategies must also be NE strategies. The defender can then safely commit to an SSE strategy; there is no selection problem for the defender. Unfortunately as shown in Example 2, if a security game has arbitrary scheduling constraints, an SSE strategy may not be part of any NE profile.

	t_1		**t_2**		**t_3**		**t_4**	
	Cov.	Unc.	Cov.	Unc.	Cov.	Unc.	Cov.	Unc.
Defender	10	9	-2	-3	1	0	1	0
Attacker	2	5	3	4	0	1	0	1

Figure 5.2: A schedule-constrained security game where the defender's SSE strategy is not an NE strategy.

Example 2. *Consider the game in Figure 5.2 with 4 targets $\{t_1, \ldots, t_4\}$, 2 schedules $s_1 = \{t_1, t_2\}$, $s_2 = \{t_3, t_4\}$, and a single defender resource. The defender always prefers that t_1 is attacked, and t_3 and t_4 are never appealing to the attacker. There is a unique SSE strategy for the defender, which places as much coverage probability on s_1 as possible without making t_2 more appealing to the attacker than t_1. The rest of the coverage probability is placed on s_2. The result is that s_1 and s_2 are both covered with probability* 0.5*. In contrast, in a simultaneous-move game, t_3 and t_4 are dominated for the attacker. Thus, there is no reason for the defender to place resources on targets that are never attacked, so the defender's unique NE strategy covers s_1 with probability*

1. That is, the defender's SSE strategy is different from the NE strategy. The difference between the defender's payoffs in these cases can also be arbitrarily large because t_1 is always attacked in an SSE and t_2 is always attacked in a NE.

The above example restricts the defender to protect t_1 and t_2 together, which makes it impossible for the defender to put more coverage on t_2 without making t_1 less appealing. If the defender could assign resources to any subset of a schedule, this difficulty is resolved. More formally, denote the set of schedules that a resource i can cover by S_i, then for any resource $1 \leq i \leq \gamma$, any subset of a schedule in S_i is also a possible schedule in S_i:

$$\forall 1 \leq i \leq \gamma: \; s' \subseteq s \in S_i \Rightarrow s' \in S_i. \tag{5.2}$$

If a security game satisfies Equation (5.2), we say it has the *SSAS* property. This is natural in many security domains, since it is often possible to cover *fewer* targets than the maximum number that a resource could possible cover in a schedule. I will show that this property is sufficient to ensure that the defender's SSE strategy must also be an NE strategy.

Lemma 6. *Suppose* $\mathbf{X}$ *is a defender strategy in a security game which satisfies the* SSAS *property and* $\mathbf{x} = \varphi(\mathbf{X})$ *is the corresponding vector of marginal probabilities. Then for any* $\mathbf{x}'$ *such that* $0 \leq x'_i \leq x_i$ *for all* $t_i \in T$*, there must exist a defender strategy* $\mathbf{X}'$ *such that* $\varphi(\mathbf{X}') = \mathbf{x}'$.

Proof. The proof is by induction on the number of t_i where $x'_i \neq x_i$, denoted by $\delta(\mathbf{x}, \mathbf{x}')$. As the base case, if there is no target i such that $x'_i \neq x_i$, the existence trivially holds because $\varphi(\mathbf{X}) = \mathbf{x}'$. Suppose the existence holds for all $\mathbf{x}, \mathbf{x}'$ such that $\delta(\mathbf{x}, \mathbf{x}') = k$, where $0 \leq k \leq N - 1$. Consider any $\mathbf{x}, \mathbf{x}'$ such that $\delta(\mathbf{x}, \mathbf{x}') = k + 1$. Then for some j, $x'_j \neq x_j$. Since $x'_j \geq 0$ and $x'_j < x_j$, we have $x_j > 0$. There must be a nonempty set of coverage vectors $\mathcal{D}_j$ that cover t_j and receive positive probability in $\mathbf{X}$. Because the security game satisfies the *SSAS* property, for every $\mathbf{d} \in \mathcal{D}_j$, there is a valid $\mathbf{d}^-$ which covers all targets in $\mathbf{d}$ except for t_j. From the defender strategy $\mathbf{X}$, by shifting $\frac{X_{\mathbf{d}}(x_j - x'_j)}{x_j}$ probability from every $\mathbf{d} \in \mathcal{D}_j$ to the corresponding $\mathbf{d}^-$, we get a defender strategy $\mathbf{X}^\dagger$ where $x^\dagger_i = x_i$ for $i \neq j$, and $x^\dagger_i = x'_i$ for $i = j$. Hence $\delta(\mathbf{x}^\dagger, \mathbf{x}') = k$, implying there exists a $\mathbf{X}'$ such that $\varphi(\mathbf{X}') = \mathbf{x}'$ by the induction assumption. By induction, the existence holds for any $\mathbf{x}, \mathbf{x}'$. □

Theorem 5.1.4. *Suppose* $\mathbf{X}$ *is a defender SSE strategy in a security game which satisfies the* SSAS *property. Then* $E(\mathbf{X}) = E^*$*, i.e.,* $\Omega_{SSE} \subseteq \Omega_M = \Omega_{NE}$.

Proof. The proof is by contradiction. First it is impossible that $E(\mathbf{X}) < E^*$ since by definition E^* is the minimum of all possible $E(\mathbf{X})$. Now suppose $\langle \mathbf{X}, g \rangle$ is an SSE profile in a security game which satisfies the *SSAS* property, and $E(\mathbf{X}) > E^*$. Let $T_a = \{t_i | v(\mathbf{X}, t_i) = E(\mathbf{X})\}$ be the set of targets that give the attacker the maximum utility against the defender strategy $\mathbf{X}$. By the definition of SSE, we have

$$u(\mathbf{X}, g(\mathbf{X})) = \max_{t_i \in T_a} u(\mathbf{X}, t_i).$$

Consider a defender mixed strategy $\mathbf{X}^*$ such that $E(\mathbf{X}^*) = E^*$. Then for any $t_i \in T_a$, $v(\mathbf{X}^*, t_i) \leq E^*$. Consider the following vector $\mathbf{x}'$:

$$x'_i = \begin{cases} x^*_i - \dfrac{E^* - v(\mathbf{X}^*, t_i) + \epsilon}{v^u_i - v^c_i}, & t_i \in T_a, \quad (5.3a) \\[2ex] x^*_i, & t_i \notin T_a, \quad (5.3b) \end{cases}$$

where ϵ is an infinitesimal positive number. Since $E^* - v(\mathbf{X}^*, t_i) + \epsilon > 0$, we have $x'_i < x^*_i$ for all $t_i \in T_a$. On the other hand, since for all $t_i \in T_a$,

$$v(\mathbf{x}', t_i) = E^* + \epsilon < E(\mathbf{X}) = v(\mathbf{X}, t_i),$$

we have $x'_i > x_i \geq 0$. Then for any $t_i \in T$, we have $0 \leq x'_i \leq x^*_i$. From Lemma 6, there exists a defender strategy $\mathbf{X}'$ corresponding to $\mathbf{x}'$. The attacker's utility of attacking each target is as follows:

$$v(\mathbf{X}', t_i) = \begin{cases} E^* + \epsilon, & t_i \in T_a, \qquad (5.4a) \\ v(\mathbf{X}^*, t_i) \leq E^*, & t_i \notin T_a. \qquad (5.4b) \end{cases}$$

Thus, the attacker's best responses to $\mathbf{X}'$ are still T_a. For all $t_i \in T_a$, since $x'_i > x_i$, it must be the case that $u(\mathbf{X}, t_i) < u(\mathbf{X}', t_i)$. By definition of attacker's SSE response g, we have,

$$u(\mathbf{X}', g(\mathbf{X}')) = \max_{t_i \in T_a} u(\mathbf{X}', t_i) > \max_{t_i \in T_a} u(\mathbf{X}, t_i) = u(\mathbf{X}, g(\mathbf{X})).$$

It follows that the defender is better off using $\mathbf{X}'$, which contradicts the assumption $\mathbf{X}$ is an SSE strategy of the defender. □

Theorem 5.1.1 and 5.1.4 together imply the following corollary.

Corollary 1. *In security games with the* SSAS *property, any defender's SSE strategy is also an NE strategy.*

We can now answer the original question posed in this chapter: when there is uncertainty over the type of game played, should the defender choose an SSE strategy or a mixed strategy Nash equilibrium or some combination of the two? For domains that satisfy the *SSAS* property, we have proven that any of the defender's SSE strategies is also an NE strategy.

Among our motivating domains, the LAX domain satisfies the *SSAS* property since all schedules are of size 1. Other patrolling domains, such as patrolling a port, also satisfy the *SSAS* property. In such domains, the defender could thus commit to an SSE strategy, which is also now known to be an NE strategy. The defender retains the ability to commit, but is still playing a best-response to an attacker in a simultaneous-move setting (assuming the attacker plays an equilibrium strategy – it does not matter which one, due to the interchange property shown above). However, the FAMS domain does not naturally satisfy the *SSAS* property because marshals must fly complete tours (though in principle they could fly as civilians on some legs of a tour). The question of selecting SSE vs. NE strategies in this case is addressed experimentally in Section 5.3.

5.1.4 Uniqueness in Restricted Games

The previous sections show that SSE strategies are NE strategies in many cases. However, there may still be multiple equilibria to select from (though this difficulty is alleviated by the interchange property). Here I will prove an even stronger uniqueness result for ARMOR games, an important restricted class of security domains. In particular, I consider security games where the defender has homogeneous resources that can cover any single target. The *SSAS* property is trivially satisfied, since all schedules are of size 1 (and "stay home" is allowed). Any vector of coverage probabilities $\mathbf{x} = (x_i)$ such that $\sum_{i=1}^{N} x_i \leq \gamma$ is a feasible strategy for the defender, so we can represent the defender strategy by marginal coverage probabilities. With a minor restriction on the attacker's payoff matrix, the defender always has a unique minimax strategy which is also the unique SSE and NE strategy. Furthermore, the attacker also has a unique NE response to this strategy.

Theorem 5.1.5. *In an* ARMOR *game, if for every target $t_i \in T$, $v_i^c \neq E^*$, then the defender has a unique minimax, NE, and SSE strategy.*

Proof. I first show the defender has a unique minimax strategy. Let $T^* = \{t_i | v_i^u \geq E^*\}$. Define $\mathbf{x}^*$ as

$$x_i^* = \begin{cases} \dfrac{v_i^u - E^*}{v_i^u - v_i^c}, & t_i \in T^*, \quad (5.5a) \\ 0, & t_i \notin T^*. \quad (5.5b) \end{cases}$$

Note that E^* cannot be less than any v_i^c – otherwise, regardless of the defender's strategy, the attacker could always get at least $v_i^c > E^*$ by attacking t_i, which contradicts the fact that E^* is the attacker's best response utility to a defender's minimax strategy. Since $E^* \geq v_i^c$ and we assume $E^* \neq v_i^c$,

$$1 - x_i^* = \frac{E^* - v_i^c}{v_i^u - v_i^c} > 0 \Rightarrow x_i^* < 1.$$

Next, we will prove $\sum_{i=1}^N x_i^* \geq \gamma$. For the sake of contradiction, suppose $\sum_{i=1}^N x_i^* < \gamma$. Consider $\mathbf{x}'$ where $x_i' = x_i^* + \epsilon$. Since $x_i^* < 1$ and $\sum_{i=1}^N x_i^* < \gamma$, we can find $\epsilon > 0$ such that $x_i' < 1$ and $\sum_{i=1}^N x_i' < \gamma$. Then every target has strictly higher coverage in $\mathbf{x}'$ than in $\mathbf{x}^*$, hence $E(\mathbf{x}') < E(\mathbf{x}^*) = E^*$, which contradicts the fact that E^* is the minimum of all $E(\mathbf{x})$.

Next, we show that if $\mathbf{x}$ is a minimax strategy, then $\mathbf{x} = \mathbf{x}^*$. By the definition of a minimax strategy, $E(\mathbf{x}) = E^*$. Hence, $v(\mathbf{x}, t_i) \leq E^* \Rightarrow x_i \geq x_i^*$. On the one hand $\sum_{i=1}^N x_i \leq \gamma$ and on the other hand $\sum_{i=1}^N x_i \geq \sum_{i=1}^N x_i^* \geq \gamma$. Therefore it must be the case that $x_i = x_i^*$ for any i. Hence, $\mathbf{x}^*$ is the unique minimax strategy of the defender.

Furthermore, by Theorem 5.1.1, we have that $\mathbf{x}^*$ is the unique defender's NE strategy. By Theorem 5.1.4 and the existence of SSE Basar and Olsder [1995], we have that $\mathbf{x}^*$ is the unique defender's SSE strategy. □

The restriction that $v_i^c \neq E^*$ is equivalent to $v_i^c < E^*$, implying that when a target is completely covered by the defender, it is never appealing to the attacker. It is a reasonable assumption because usually if the attacker knew attacking a particular target would definitely get himself caught, he would never consider to attack it.

Theorem 5.1.6. *In an* ARMOR *game, if for every target $t_i \in T$, $v_i^c \neq E^*$ and $v_i^u \neq E^*$, the attacker has a unique NE strategy.*

Proof. Define $\mathbf{x}^*$ and T^* to be the same as in the proof of Theorem 5.1.5. Given the defender's unique NE strategy $\mathbf{x}^*$, in any attacker's best response, only $t_i \in T^*$ can be attacked with positive probability, because,

$$v(\mathbf{x}^*, t_i) = \begin{cases} E^* & t_i \in T^* \qquad (5.6a) \\ v_i^u < E^* & t_i \notin T^* \qquad (5.6b) \end{cases}$$

Suppose $\langle \mathbf{x}^*, \mathbf{a} \rangle$ forms an NE profile. We have

$$\sum_{t_i \in T^*} a_i = 1 \tag{5.7}$$

For any $t_i \in T^*$, we know from the proof of Theorem 5.1.5 that $x_i^* < 1$. In addition, because $v_i^u \neq E^*$, we have $x_i^* \neq 0$. Thus we have $0 < x_i^* < 1$ for any $t_i \in T^*$. For any $t_i, t_j \in T^*$, necessarily $a_i \Delta\mu_i = a_j \Delta\mu_j$. Otherwise, assume $a_i \Delta\mu_i > a_j \Delta\mu_j$. Consider another defender's strategy $\mathbf{x}'$ where $x_i' = x_i^* + \epsilon < 1$, $x_j' = x_j^* - \epsilon > 0$, and $x_k' = x_k^*$ for any $k \neq i, j$.

$$u(\mathbf{x}', \mathbf{a}) - u(\mathbf{x}^*, \mathbf{a}) = \epsilon(a_i \Delta\mu_i - a_j \Delta\mu_j) > 0$$

Hence, $\mathbf{x}^*$ is not a best response to $\mathbf{a}$, which contradicts the assumption that $\langle \mathbf{x}^*, \mathbf{a} \rangle$ is an NE profile. Therefore, there exists $\beta > 0$ such that, for any $t_i \in T^*$, $a_i \Delta\mu_i = \beta$. Substituting a_i with $\beta / \Delta\mu_i$ in Equation (5.7), we have

$$\beta = \frac{1}{\sum_{t_i \in T^*} \frac{1}{\Delta\mu_i}}$$

Then we can explicitly write down $\mathbf{a}$ as

$$a_i = \begin{cases} \dfrac{\beta}{\Delta\mu_i}, & t_i \in T^*, \quad (5.8a) \\ 0, & t_i \notin T^*. \quad (5.8b) \end{cases}$$

As we can see, $\mathbf{a}$ defined by (5.8a) and (5.8b) is the unique attacker NE strategy. □

The implication of Theorem 5.1.5 and Theorem 5.1.6 is that in the simultaneous-move game, there is a unique NE profile, which as a result, gives each player a unique expected utility.

5.2 Multiple Attacker Resources

To this point I have assumed that the attacker will attack exactly one target. We now extend our security game definition to allow the attacker to use multiple resources to attack multiple targets simultaneously. To keep the model simple, I assume homogeneous resources (for both players) and schedules of size 1. The defender has $\gamma < N$ resources which can be assigned to protect any target, and the attacker has $\zeta < N$ resources which can be used to attack any target. Attacking the same target with multiple resources is equivalent to attacking with a single resource. The defender's pure strategy is again a coverage vector $\mathbf{d} = (d_1, \ldots, d_N) \in \mathcal{D}$, where $d_i \in \{0, 1\}$ represents whether t_i is covered or not. Similarly, the attacker's pure strategy is an attack vector

$\mathbf{q} = (q_1, \ldots, q_N) \in \mathcal{Q}$. We have $\sum_{i=1}^N d_i = \gamma$ and $\sum_{i=1}^N q_i = \zeta$. If $\langle \mathbf{d}, \mathbf{q} \rangle$ is played, the defender gets a utility of

$$u(\mathbf{d}, \mathbf{q}) = \sum_{i=1}^{N} q_i \left(d_i \mu_i^c + (1 - d_i) \mu_i^u \right)$$

while the attacker's utility is given by

$$v(\mathbf{d}, \mathbf{q}) = \sum_{i=1}^{N} q_i \left(d_i \nu_i^c + (1 - d_i) \nu_i^u \right)$$

The defender's mixed strategy is again a vector **X** which specifies the probability of playing each $\mathbf{d} \in \mathcal{D}$. Similarly, the attacker's mixed strategy **A** is a vector of probabilities corresponding to all $\mathbf{q} \in \mathcal{Q}$.

In security games with multiple attacker resources, the defender's SSE strategy may not be part of any NE profile, even if there are no scheduling constraints as shown in the following example.

	$\mathbf{t_1}$		$\mathbf{t_2}$		$\mathbf{t_3}$	
	Cov.	Unc.	Cov.	Unc.	Cov.	Unc.
Defender	0	-1	-100	$-100 - \epsilon$	0	$0 - \epsilon$
Attacker	$100 - \epsilon$	100	0	10	$5 - \epsilon$	5

Figure 5.3: A security game with multiple attacker resources where the defender's SSE strategy is not an NE strategy.

Example 3. *Consider the game shown in Figure 5.3. There are* 3 *targets* t_1, t_2, t_3. *The defender has* 1 *resource, and the attacker has* 2 *resources. Therefore the defender's pure strategy space is the set of targets to protect:* $\{t_1, t_2, t_3\}$, *while the attacker's pure strategy space consists of the pairs of targets:* $\{(t_1, t_2), (t_1, t_3), (t_2, t_3)\}$. *If the defender protects* t_1 *and the attacker attacks* (t_1, t_2), *the defender's utility is* $\mu_1^c + \mu_2^u = -100 - \epsilon$ *and the attacker's utility is* $\nu_1^c + \nu_2^u = 110 - \epsilon$. *In* *this example, t_1 is very appealing to the attacker no matter if it is covered or not, so t_1 is always attacked. If t_2 is attacked, the defender gets a very low utility, even if t_2 is defended. So in the SSE, the defender wants to make sure that t_2 is not attacked. The defender's SSE strategy places at least .5 probability on t_2, so that t_1 and t_3 are attacked instead of t_2 (recall that the attacker breaks ties in the defender's favor in an SSE). The attacker's SSE response is* $\mathbf{A} = (0, 1, 0)$*, i.e., to always attack t_1 and t_3. The other .5 defense probability will be placed on t_1 because $\Delta\mu_1 > \Delta\mu_3$. So, the SSE profile is* $\langle \mathbf{X}, \mathbf{A} \rangle$*, where* $\mathbf{X} = (.5, .5, 0)$.

Next, I show that there is no NE in which the defender plays $\mathbf{X}$. *Suppose there is an NE profile* $\langle \mathbf{X}, \mathbf{A}' \rangle$. *Given* $\mathbf{X}$*, the attacker's utility for attacking t_1 is higher than the utility for attacking t_2, so it must be that t_1 is always attacked in this NE. Therefore, the attacker never plays $\langle t_2, t_3 \rangle$. However, this implies that t_1 is the most appealing target for the defender to cover, because* $u(t_1, \mathbf{A}) > u(t_i, \mathbf{A}), i \in \{2, 3\}$. *So to be a best response, the coverage of t_1 would need to be* 1 *instead of* 0.5*, contradicting the assumption that* $\mathbf{X}$ *is an equilibrium strategy for the defender.*

5.3 Experimental Results

While the theoretical results presented earlier resolve the leader's dilemma for many interesting classes of security games, as we have seen, there are still some cases where SSE strategies are distinct from NE strategies for the defender. One case is when the schedules do not satisfy the *SSAS* property, and another is when the attacker has multiple resources. In this section, I provide experiments to further investigate these two cases, offering evidence about the frequency with which SSE strategies differ from all NE strategies across randomly generated games, for a variety of parameter settings.

The methodology used is as follows. For a particular game instance, I first compute an SSE strategy **X** using Dobss Paruchuri et al. [2008]. Then I use the linear feasibility program below to determine whether or not this SSE strategy is part of some NE profile by attempting to find an appropriate attacker response strategy.

$$A_{\mathbf{q}} \in [0,1], \text{ for all } \mathbf{q} \in Q \tag{5.9}$$

$$\sum_{\mathbf{q}\in Q} A_{\mathbf{q}} = 1 \tag{5.10}$$

$$A_{\mathbf{q}} = 0, \text{ for all } v(\mathbf{X}, \mathbf{q}) < E(\mathbf{X}) \tag{5.11}$$

$$\sum_{\mathbf{q}\in Q} A_{\mathbf{q}} u(\mathbf{d}, \mathbf{q}) \le Z, \text{ for all } \mathbf{d} \in \mathcal{D} \tag{5.12}$$

$$\sum_{\mathbf{q}\in Q} A_{\mathbf{q}} u(\mathbf{d}, \mathbf{q}) = Z, \text{ for all } \mathbf{d} \in \mathcal{D} \text{ with } X_{\mathbf{d}} > 0 \tag{5.13}$$

Here Q is again the set of attacker pure strategies, which is the set of targets when attacker has a single resource. The probability that the attacker plays **q** is denoted by $A_{\mathbf{q}}$ which must be between 0 and 1 (Constraint (5.9)). Constraint (5.10) forces the probabilities to sum to 1. Constraint (5.11) prevents the attacker from placing positive probabilities on pure strategies which give the attacker a utility less than the best response utility $E(\mathbf{X})$. In constraints (5.12) and (5.13), Z is a variable which represents the maximum expected utility the defender can get among all pure strategies given the attacker's strategy **A**, and $X_{\mathbf{d}}$ denotes the probability of playing **d** in **X**. These two constraints require the defender's strategy **X** to be a best response to the attacker's mixed strategy. Therefore, any feasible solution **A** to this linear feasibility program, taken together with the Stackelberg strategy **C**, constitutes a Nash equilibrium. Conversely, if $\langle \mathbf{X}, \mathbf{A} \rangle$ is a Nash equilibrium, **A** must satisfy all of the LP constraints.

In this set of experiments, I varied:

- the number of attacker resources,
- the number of (homogeneous) defender resources,
- the size of the schedules that resources can cover,
- the number of schedules.

For each parameter setting, I generated 1000 games with 10 targets. For each target t_i, a pair of defender payoffs (μ_i^c, μ_i^u) and a pair of defender payoffs (ν_i^u, ν_i^c) were drawn uniformly at random from the set $\{(x, y) \in \mathbb{Z}^2 \; : \; x \in [-10, +10], y \in [-10, +10], x > y\}$. In each game in the experiment, all of the schedules have the same size, except there is also always the empty schedule—assigning a resource to the empty schedule corresponds to the resource not being used. The schedules are randomly chosen from the set of all subsets of the targets that have the size specified by the corresponding parameter.

The results of our experiments are shown in Figure 5.4. The plots show the percentage of games in which the SSE strategy is not an NE strategy, for different numbers of defender and attacker resources, different schedule sizes, and different numbers of schedules. Each row corresponds to a different number of attacker resources, and each column to a different schedule size. The number of defender resources is on the x-axis, and each number of schedules is plotted separately. For each parameter setting, 1000 random games with 10 targets were generated. The *SSAS* property holds in the games with schedule size 1 (shown in column 1); *SSAS* does not hold in the games with schedule sizes 2 and 3 (columns 2 and 3). For the case where there is a single attacker resource and schedules have size 1, the *SSAS* property holds, and the experimental results

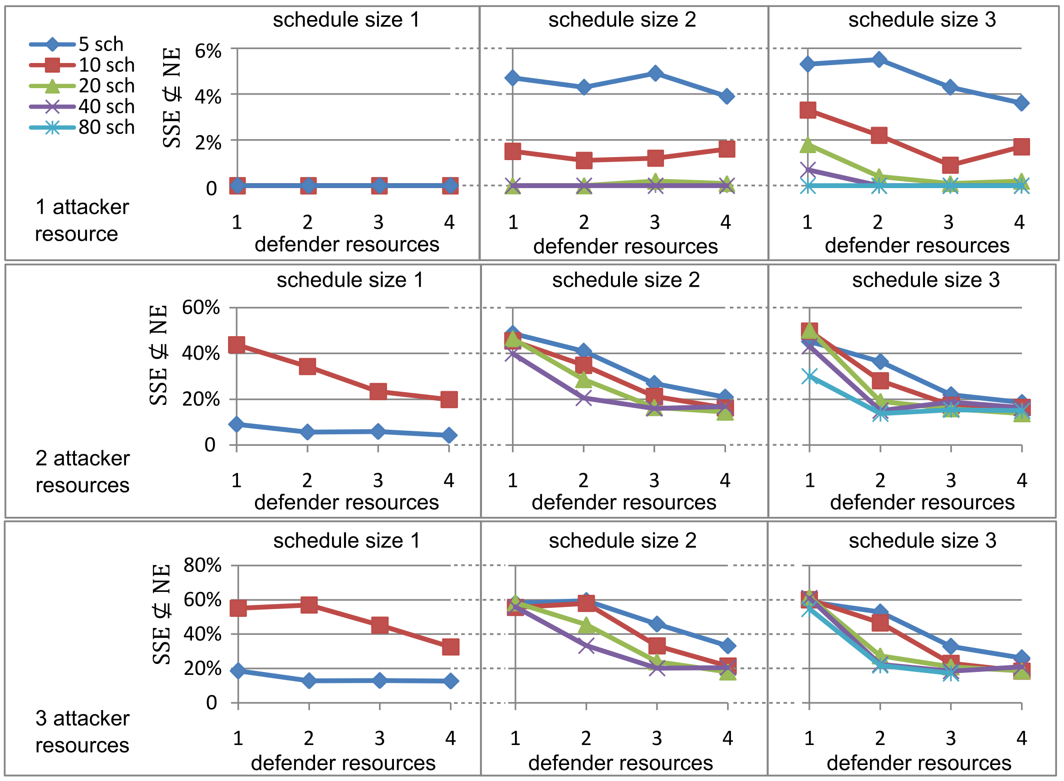


Figure 5.4: The number of games in which the SSE strategy is not an NE strategy, for different parameter settings.

confirm the theoretical result that the SSE strategy is always an NE strategy. If we increase either the number of attacker resources or the schedule size, then the theoretical result no longer holds, and indeed we start to see cases where the SSE strategy is not an NE strategy.

Let us first consider the effect of increasing the number of attacker resources. We can see that the number of games in which the defender's SSE strategy is not an NE strategy increased significantly as the number of attacker resources increased, especially as it went from 1 to 2 (note the different scales on the y-axes). In fact, when there were 2 or 3 attacker resources, the phenomenon that in many cases the SSE strategy is not an NE strategy was consistent across a wide range of values for the other parameters.

Now, let us consider the effect of increasing the schedule size. When the schedule size (with a single attacker resource) increases, the *SSAS* property no longer holds, and so there exist some games where the SSE strategy is not an NE strategy—but the percentage of such games was generally small ($< 6\%$). Also, as more random schedules were generated, the number of games where the SSE strategy is not an NE strategy dropped to zero. This is particularly encouraging for domains like FAMS, where the schedule sizes are relatively small (2 in most cases), and the number of possible schedules is large relative to the number of targets. The effect of increasing the number of defender resources is more ambiguous. When there are multiple attacker resources, increasing the schedule size sometimes increases and sometimes decreases the number of games where the SSE strategy is not an NE strategy.

The main message to take away from the experimental results appears to be that for the case of a single attacker resource, SSE strategies are usually also NE strategies even when *SSAS* does not hold, which appears to further justify the practice of playing an SSE strategy. On the other hand, when there are multiple attacker resources, there are generally many cases where the SSE strategy is not an NE strategy. This strongly poses the question of what should be done in the case of multiple attacker resources (in settings where it is not clear whether the attacker can observe the defender's mixed strategy). While a formal answer to this question is out of the scope of this thesis, it has been studied extensively in the literature. In particular, security games with multiple attacker resources were studied in Korzhyk et al. [2011b]. Korzhyk et. al. also provided an extensive-form game formulation to further address the defender's dilemma in security games where the attacker's observability is uncertain and the defender's SSE strategy is not an NE strategy Korzhyk et al. [2011a].

Chapter 6: Patrolling in Transit Systems

This chapter presents game-theoretic models for patrolling in transit systems, where timing is an integral part of what determines the effectiveness of patrol schedules, in addition to the set of targets being covered. For example, trains, buses and ferries follow specific schedules, and in order to protect them the patroller needs to be at the right place at the right time. As introduced earlier in Section 2.1, a motivating example that I will focus on is the problem of scheduling randomized ticket inspections for fare evasion deterrence in the Los Angeles Metro Rail system.

Patrolling in transit systems introduces significant new challenges in designing and implementing game-theoretic models. First, there can be exponentially many feasible patrols, which are sequences of patrol actions subject to subject to both the spatial and temporal constraints of travel within the underlining system. There can be trillions of feasible ticket inspection patrols in complex real world train systems. Second, there are potentially a large number of opponents. The Los Angeles Metro Rail system serves 300,000 riders everyday among which approximately 6% are evading tickets Booz Allen Hamilton [2007]. These potential fare evaders are of many types, each corresponds to a unique travel pattern. Finally, *execution uncertainty* (errors, emergencies, noise, etc) in transit systems can affect the defender units' ability to carry out their planned schedules in later time steps. The patrols in train system may get interrupted for a variety of reasons such as writing citations, felony arrests, and handling emergencies. Such interruptions can cause the officers to miss the train that they were supposed to take and void the rest of the schedule.

To address the three challenges mentioned above, I provide new game-theoretic models with a focus on the problem of fare evasion deterrence in transit systems. The result of this investigation is a novel application called TRUSTS (**T**actical **R**andomization for **U**rban **S**ecurity in **T**ransit **S**ystems), for fare evasion deterrence in urban transit systems, carried out in collaboration with the Los Angeles Sheriff's Department (LASD). TRUSTS models this problem as a Stackelberg game with one leader (the LASD) and many followers, in which each metro rider (a follower) takes a fixed route at a fixed time. The leader precommits to a mixed patrol strategy (a probability distribution over all pure patrols), and riders observe this mixed strategy before deciding whether to buy the ticket or not (the decision to ride having already been made), in order to minimize their expected total cost, following for simplicity the classic economic analysis of rational crime Becker and Landes [1974]. Both ticket sales and fines issued for fare evasion translate into revenue to the government. Therefore the optimization objective chosen for the leader is to maximize total revenue (total ticket sales plus penalties).

The remainder of this chapter is divided into three parts. In Section 6.1, I introduce the first generation of TRUSTS (TRUSTSv1), assuming the leader has perfect execution. TRUSTSv1 uses the *transition graph*, which captures the spatial as well as temporal structure of the domain, and solves for the optimal (fractional) flow through this graph, using linear programming (LP). Such a flow can be interpreted as a marginal coverage vector from which a mixed strategy of feasible patrols can be extracted. Additionally, I show that a straightforward approach to extracting patrol strategies from the marginals faces important challenges: it can create infeasible patrols that violate the constraint on patrol length, and it can generate patrols that switch too frequently between trains, which can be difficult for patrol personnel to carry out. Thus, I present a novel technique to overcome these difficulties using an extended formulation on a *history-duplicate transition graph* that allows us to specify constraints and preferences on individual patrols.

In Section 6.2, I present TRUSTSv2, generalizing and extending TRUSTSv1 to model execution uncertainty and provide automatic contingency plans when the patrol officer deviates from the original schedule. TRUSTSv2 models execution uncertainty as Markov Decision Processes. Computing a Stackelberg equilibrium for this game presents significant computational challenges due to the exponential dimension in the defender's strategy space. I show that when the utility functions have a certain separable structure, the leader's strategy space can be compactly represented. As a result the problem can be reduced to a polynomial-sized optimization problem, solvable by existing approaches for Bayesian Stackelberg games such as Dobss and Hunter. Furthermore I show that from the compactly represented solution we can generate randomized patrol schedules with contingency plans. Such contingency plans can be implemented as a smart-phone app carried by patrol units, or as a communication protocol with a central operator. Finally, I will show how this approach can be applied to the fare evasion deterrence problem and provide details of a smart-phone app that facilitates the deployment of the TRUSTSv2 in the Los Angeles Metro System.

In Section 6.3, I provide simulation results of TRUSTSv1 and TRUSTSv2 based on actual ridership data provided by the LASD, for four LA Metro train lines (Blue, Gold, Green, and Red). My simulation results on TRUSTSv1 suggest the possibility of significant fare evasion deterrence and hence prevention of revenue loss with very few resources. Field trials conducted by the LASD using patrol schedules generated by TRUSTSv1 show encouraging results but also reveal serious issues. While TRUSTSv1 schedules were more effective in catching fare evaders, they were vulnerable to execution errors and often got interrupted and abandoned before completion. My further simulation results show that execution uncertainty has a significant impact on revenue and TRUSTSv2 significantly outperforms TRUSTSv1 in the presence of execution uncertainty.

6.1 TRUSTSv1: Deterministic Model for Perfect Execution

In this section, I introduce TRUSTSv1, a deterministic game-theoretic model assuming the defender's patrol actions are always executed perfectly. A formal problem setting for the fare evasion deterrence problem is given in Section 6.1.1. Section 6.1.2 introduces the basic and extended linear program formulations for solving the problem.

6.1.1 Formal Model

We model this problem as a leader-follower Stackelberg game with one leader (the LASD) and multiple followers (riders). In this game, a pure leader strategy is a *patrol*, i.e., a sequence of *patrol actions* (defined below), of constant bounded duration. The two possible pure follower strategies are buying and not buying. Each follower observes the strategy the leader commits to and plays a best response. There are many types of followers, one for each source, destination, and departure time triple (corresponding to the set of all riders who take such a trip). In general the leader's strategies will be mixed; the followers are assumed to play pure strategies Conitzer and Sandholm [2006].

Train System: The train system consists of a single line on which trains travel back and forth, in general with multiple trains traveling simultaneously. The system operates according to a fixed daily schedule, with trains arriving at stations at (finitely many) designated times throughout the day. Therefore we can model time as discrete, focusing only on the time steps at which some train arrival/departure event occurs. We use the (directed) *transition graph* $G = \langle V, E \rangle$ to encode the daily timetable of the metro line, where a vertex $v = \langle l, \tau \rangle$ corresponds to some pair of location (train station) l and time point τ. An edge in G represents a possible (minimal) action. In particular, there is an edge from $\langle l, \tau \rangle$ to $\langle l', \tau' \rangle$ if:

- l' is either the predecessor or successor of l in the station sequence and $\langle l, \tau \rangle$ and $\langle l', \tau' \rangle$ are two consecutive stops for some train in the train schedule (*traveling action*), or
- $l' = l$, $\tau < \tau'$, and there is no vertex $\langle l, \tau'' \rangle$ with $\tau < \tau'' < \tau'$ (*staying action*).

We refer to the entire path that a given train takes through G, from the start station to the terminal station, as a *train path*. For simplicity, this model assumes a single train line in the system, however the solution methods presented in this thesis are applicable to extensions of multiple intersecting lines with transfer points.

Patrols: There are a fixed number γ of deployable patrol units, each of which may be scheduled on a patrol of duration at most κ hours. There are two sorts of patrol actions, which a given patrol unit can alternate between on its shift: *on-train* inspections (in which patrollers ride the train, inspecting passengers), and *in-station* inspections (in which they inspect passengers as they exit the station). A pure patrol strategy is represented mathematically as a path in G for each patrol unit, in which an edge e represents an *atomic* patrol action, i.e., inspecting in-station from the time of one train event at that station to the next (at that station) or inspecting on-train as it travels from one station to the next. Each edge e has a duration δ_e equal to the corresponding patrol action duration and an effectiveness value f_e, which represents the percentage of the relevant ridership inspected by this action. For both in-station and on-train inspections, f_e depends on the ridership volume at that location and time of day and on the duration. A valid pure patrol strategy is a set of paths $P_1, ..., P_\gamma$, each of size at most κ, i.e., $\sum_{e \in P_i} \delta_e \leq \kappa$.

Example 4. *A simple scenario with* 3 *stations* (A, B, C) *and* 4 *discrete time points (6pm, 7pm, 8pm, 9pm) is given in Figure 6.1. The dashed lines represent staying actions; the solid lines represent traveling actions. There are* 4 *trains in the system; all edge durations are 1 hour. A sample train path here is* $\langle A, 6pm\rangle \rightarrow \langle B, 7pm\rangle \rightarrow \langle C, 8pm\rangle$. *In this example, if* $\kappa = 2$ *and* $\gamma = 1$, *then the valid pure leader strategies (pure patrol strategies) consist of all paths of length* 2.

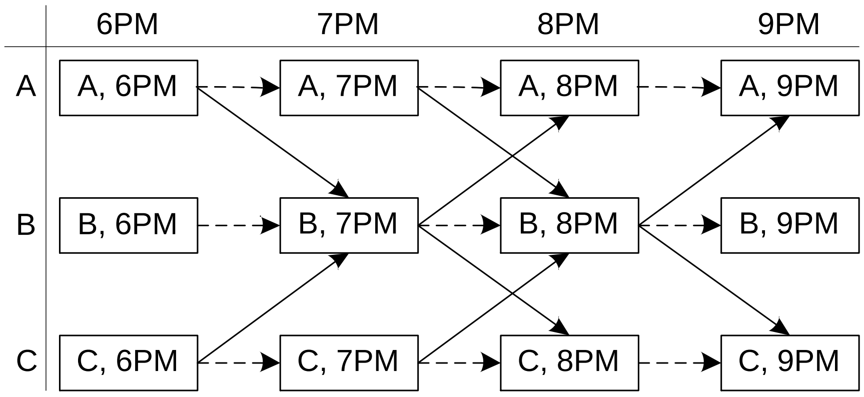


Figure 6.1: The transition graph of a toy problem instance.

Riders: The riders are assumed to be daily commuters who take a fixed route at a fixed time. Horizon research corporation 2002 suggests more than 82% of riders use the system at least 3 days a week. A rider's *type* λ is therefore defined by the path he takes in the graph. Because there is a single train line, riders never linger in stations, i.e., do not follow any "stay" edges (staying at a station) mid-journey; the last edge of every follower type is a (short) stay edge, representing the action of "exiting" the destination station, during which the rider may be subject to in-station inspection. Therefore the space Λ of rider types corresponds to the set of all subpaths of train paths. (When G is drawn as in Figure 6.1, all rider paths are "diagonal" except for the last edge.)

The ticket price that a rider of type λ pays is a nominal fee ρ_λ, with the fine for fare evasion ϱ_λ much greater. As the riders follow the same route every day, they could estimate the likelihood of being inspected, based on which they make a decision as to whether to buy a ticket. The ticket cost is fixed but the possibility of being caught and fined for fare evasion is uncertain; based on the likelihood of being caught, the rider must make a decision as to whether to buy a ticket. We assume the riders know the inspection probability perfectly, and are rational, risk-neutral economic actors Becker and Landes [1974], who make this choice in order to minimize expected cost. (Equivalently, we can assume that some riders are conscientious, but that selfish or rational riders are distributed evenly among all passenger types.)

Given a pure patrol strategy of the γ units, $(P_1, \ldots, P_\gamma)$, the inspection probability for a rider of type $\lambda \in \Lambda$ is:

$$\min\{1, \sum_{i=1}^{\gamma} \sum_{e \in P_i \cap \lambda} f_e\}, \tag{6.1}$$

and therefore his expected utility is the negative of the expected amount he pays: $-\rho_\lambda$ if he buys the ticket and $-\varrho_\lambda \min\{1, \sum_{i=1}^{\gamma} \sum_{e \in P_i \cap \lambda} f_e\}$ otherwise. The inspection probability for a mixed strategy is then the expectation of Equation (6.1), taken over the distribution of pure strategies.

We justify the inspection probability in Equation (6.1) as follows. First, consider on-train inspections. The fraction of the train that is inspected in a given inspection action is determined by f_e (which depends on ridership volume). The key is that in the next inspection action, a patrol will not reinspect the fraction of the train that is already inspected in a previous inspection action. Therefore, unlike in settings where patrollers may repeatedly draw a random sample from the same set train passengers to inspect, in our setting, the probabilities f_e are added rather than multiplied. Now also consider in-station inspections. Since a rider taking a journey only exits a single station, a rider will encounter at most one in-station inspection. Finally, when multiple patrol units cover the same edge e, the inspection probability given by (6.1) is the sum of the contributions from each patrol unit, capped at 1. This is a reasonable assumption when the number of patrol units on each edge e is small, as multiple patrol units on the same train could check different cars or different portions of the same car, and multiple patrol units inspecting at the same station could be checking different exits.

Objective: The leader's utility, equal to total expected revenue, can be decomposed into utilities from bilateral interactions with each individual follower. Hence the game is equivalent to a Bayesian Stackelberg game between one leader with one type and one follower with multiple types. Specifically, we denote the prior probability of a follower type $\lambda \in \Lambda$ (proportional to its ridership volume) by p_λ.

Furthermore, these utility functions imply that the game is zero sum, in which case the Stackelberg equilibrium is equivalent to the maximin solution. Although such zero-sum Bayesian games are solvable by either applying the LP formulation of Ponssard and Sorin [1980] or treating the Bayesian game as a extensive-form game and applying the sequence form LP formulation of Koller et al. [1994], those LP formulations would be impractical here because they explicitly enumerate the exponential number of pure strategies of the leader.

6.1.2 Approach

In this section, I formulate a linear program which finds a maximum-revenue (mixed) patrol strategy. As noted above, the leader's space of *pure* strategies is exponentially large, even with a single patrol unit. (A pure strategy consists of γ pure patrol strategies, one for each patrol unit.) We avoid this difficulty by compactly representing mixed patrol strategies by marginal coverage on edges x_e of the transition graph (the marginal strategy), i.e., by the expected numbers of inspections that will occur on these edges. Subsequently, we construct a mixed strategy (i.e., a probability distribution over pure strategies) consistent with the marginal coverage.

For expository purposes, I will first present a basic LP formulation based on the compactly represented strategy. This basic formulation however may generate infeasible patrols due to a couple of key issues. I will then introduce an extended formulation to address these issues.

6.1.2.1 Basic Formulation

We denote the set of possible starting vertices in the transition graph $G = \langle V, E \rangle$ by $V^+ \subset V$, and the set of possible ending vertices by $V^- \subset V$. For algorithmic convenience, we add to the transition graph a source v^+ with edges to all vertices in V^+ and a sink v^- with edges from all vertices in V^-. We assign these additional dummy edges zero duration and zero effectiveness.

Based on this graph, we provide a linear program (shown in Figure 6.2) to provide an upper bound on the optimal revenue achievable. Here u_λ denotes the expected value paid by a rider of type λ, and so $p_\lambda u_\lambda$ is the expected total revenue from riders of this type; x_e is the expected number of inspections on edge e. Constraint (6.4) bounds the total flow entering and exiting the system by γ, the number of total patrol units allowed. Constraint (6.5) enforces conservation

$$\max_{\mathbf{x},\mathbf{u}} \sum_{\lambda\in\Lambda} p_\lambda u_\lambda \tag{6.2}$$

$$\text{s.t. } u_\lambda \leq \min\{\rho_\lambda,\ \varrho_\lambda \sum_{e\in\lambda} x_e f_e\},\ \text{for all } \lambda\in\Lambda \tag{6.3}$$

$$\sum_{v\in V^+} x_{(v^+,v)} = \sum_{v\in V^-} x_{(v,v^-)} \leq \gamma \tag{6.4}$$

$$\sum_{(v',v)\in E} x_{(v',v)} = \sum_{(v,v^\dagger)\in E} x_{(v,v^\dagger)},\ \text{for all } v\in V \tag{6.5}$$

$$\sum_{e\in E} \delta_e \cdot x_e \leq \gamma\cdot\kappa, 0 \leq x_e \leq \gamma, \forall e\in E \tag{6.6}$$

Figure 6.2: Basic Formulation

of flow, which clearly is satisfied by any mixed patrol strategy. Constraint (6.6) limits the total number of time units to $\gamma\cdot\kappa$, and also bounds x_e for each e by γ.

Finally, let us consider Constraint (6.3), which indicates that the rider will best respond, by bounding the expected cost to a rider of type λ by both the ticket price ρ_λ and $\varrho_\lambda \min\{1, \sum_{e\in\lambda} x_e f_e\} = \min\{\varrho_\lambda, \varrho_\lambda \sum_{e\in\lambda} x_e f_e\}$, the formulation's estimate of the expected fine if the rider chooses not to buy. However, the latter is only an *overestimate* of the actual expected fine of not buying. This is because the expression $\min\{1, \sum_{e\in\lambda} x_e f_e\}$ only caps the expectation (over its pure strategies) of the inspection probability at 1, but allows a pure strategy $(P_1,\ldots,P_\gamma)$ in its support to achieve $\sum_{i=1}^{\gamma}\sum_{e\in P_i\cap\lambda} f_e > 1$, whereas according to (6.1) the inspection probability of each pure strategy should be at most 1. This results in an overestimate of the actual inspection probability (and thus the leader's utility). As a result the solution of this LP provides only an upper bound on the optimal revenue. Fortunately, once we generate the patrols from the marginals we are able to compute the actual best-response utilities of the riders. Our experiments show that the differences between the actual utilities and the upper-bounds given by the LP formulation are small. The remaining task is to construct a γ-unit mixed patrol strategy whose marginals match the marginal strategy $\mathbf{x}$.

Proposition 4. *Given a marginal strategy* $\mathbf{x}$*, a* γ*-unit mixed strategy for the leader that produces the same coverage on each edge* e *as* $\mathbf{x}$ *does can be constructed in polynomial time.*

Proof. First, we construct a set Υ of weighted patrol paths, by extracting distinct source-to-sink flows from $\mathbf{x}$ through the following iterative procedure.

1. Find a path P from v^+ to v^- where $x_e > 0$ for all $e \in P$. If no such path exists, terminate because x_e must then be 0 for all $e \in E$ (due to Constraint (6.5)). Otherwise go to step 2.

2. Let $x^* = \min_{e \in P}\{x_e\}$. Add path P with weight x^* to the set Υ. Deduct x^* from x_e for all $e \in P$. Go to step 1.

Since every iteration removes a complete source-to-sink flow, constraint (6.5) is maintained throughout the execution of this procedure. The procedure's running time is polynomial because at least one new x_e is set to 0 in each iteration.

Finally, we create a mixed strategy of joint patrol paths (with γ units) that matches exactly the set of weighted patrol paths Υ obtained in the procedure above, and thus the marginal strategy $\mathbf{x}$. To do this, we could assign a path of weight x^* to the γ units independently, each with an equal probability of $\frac{x^*}{\gamma}$. Since $x^* \leq \gamma$, we have $\frac{x^*}{\gamma} \leq 1$. □

6.1.2.2 Issues with the Basic Formulation

There are two fundamental issues with the basic formulation. First, the mixed strategy constructed can fail to satisfy the patrol length limit of κ, notwithstanding Constraint (6.6) on the sum of the lengths of all patrols, and hence be infeasible. In fact, the marginal strategy computed in the basic formulation may not correspond to any feasible mixed strategy in which all patrols have length at most κ. Consider the counterexample in Figure 6.3. Edges $v_1 \rightarrow v_2$ and $v_2 \rightarrow v_3$ represent two real atomic actions, each with duration 1. Patrols must start from either v_1 or v_3, but can terminate at any of v_1, v_2 and v_3. This is specified using v^+ and v^-, the dummy source and sink respectively. Let $\kappa = 1$ and $\gamma = 1$. It can be verified that the marginal strategy shown in Figure 6.3 satisfies constraints (6.4) through (6.6). However, the only corresponding mixed strategy is to take $v^+ \rightarrow v_3 \rightarrow v^-$ with 50% probability and $v^+ \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v^-$ with 50% probability. This mixed strategy is infeasible since its second patrol has duration greater than 1. This patrol length violation arises because the basic formulation only constrains the average patrol length, and therefore allows the use of overlong patrols as long as some short patrols are also used.

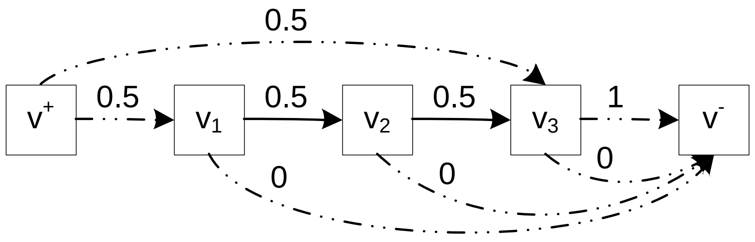


Figure 6.3: Example of an infeasible marginal strategy.

Second, the paths selected according the constructed mixed strategy may *switch* between trains or between in-station and on-train at impractically large number of times, making the patrol path difficult to implement and error-prone. This is an important issue because we want real LASD officers to be able to carry out these strategies. The more switches there are in a patrol strategy, the more instructions the patrol unit has to remember, and the more likely they will miss a switch due to imperfections in the train schedule and/or the unit's mis-execution of the instructions. For example, in Example 4, $\langle A, 6pm\rangle \rightarrow \langle B, 7pm\rangle \rightarrow \langle A, 8pm\rangle$ and

$\langle C, 6pm\rangle \rightarrow \langle B, 7pm\rangle \rightarrow \langle C, 8pm\rangle$ each has 1 switch while $\langle A, 6pm\rangle \rightarrow \langle B, 7pm\rangle \rightarrow \langle C, 8pm\rangle$ and $\langle C, 6pm\rangle \rightarrow \langle B, 7pm\rangle \rightarrow \langle A, 8pm\rangle$ each has 0. Both path pairs cover the same set of edges, however the second pair will be preferred because it is easier to implement.

6.1.2.3 Extended Formulation

Now I present a more sophisticated formulation design to address the two aforementioned issues. The difficulty involved in imposing constraints on the patrol paths (i.e., penalizing or forbidding certain paths) in the marginal representation is that paths themselves are not represented, instead being encoded only as marginal coverage.

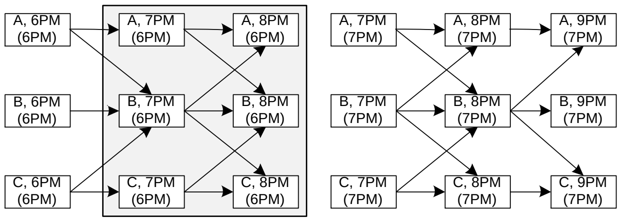
Hence the key idea is to preserve sufficient path history information within vertices to be able to evaluate our constraints, while avoiding the exponential blowup creating a node for every path would cause. To this end, we construct a new graph, called the *History-Duplicate Transition graph* (HDT graph), by creating multiple copies of the original vertices, each corresponding to a unique (partial) patrol history. This duplication is performed only to preserve patrol history information that is necessary as I will show next.

I will first explain how to construct the HDT graph from a transition graph G in order to restrict the length of patrol paths to at most κ. The HDT graph is composed of multiple restricted copies of G (i.e., subgraphs of G), each corresponding to a unique starting time. For the copy corresponding to starting time point τ^*, we only keep the subgraph that can be reached from time τ^*, i.e., vertices $v = \langle l, \tau\rangle \in V$ where $\tau^* \leq \tau \leq \tau^* + \kappa$. Thus, in each restricted copy of G, the length of any path is guaranteed to be less than or equal to κ. Since there are a finite number of distinct possible starting time points (i.e., all distinct discrete time points in V^+), the new graph is a linear expansion of G. It is however often desirable to use fewer starting time points (e.g., one for every hour) to improve runtime efficiency at the cost of small quality loss.

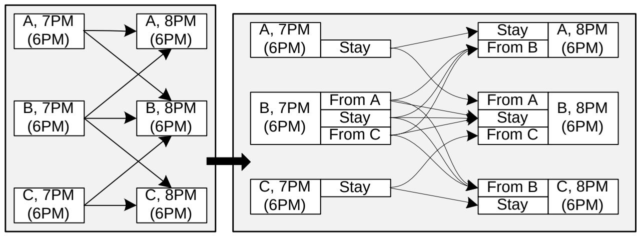
Figure 6.4(a) shows the HDT graph (the shaded portion further explained below) of Example 4 with $\kappa = 2$ and 2 starting time points, $6pm$ and $7pm$. The HDT graph is thus composed of two restricted copies of the original transition graph. In each vertex, the time shown in parenthesis indicates the starting time point. For example, the original vertex $\langle A, 7pm \rangle$ now has two copies $\langle A, 7pm, (6pm) \rangle$ and $\langle A, 7pm, (7pm) \rangle$ in the HDT graph. For the starting time point of $6pm$, the patrol must end at or before $8pm$, hence we do not need to keep vertices whose discrete time point is $9pm$. For the starting time point of $7pm$, the patrol must start at or after $7pm$, hence we do not need to keep vertices whose discrete time point is $6pm$. The two restricted copies *are not two separate graphs* but a single graph that will be tied together by the dummy source and sink.

Next, I will explain how to further extend the HDT graph to penalize complex patrol paths. The idea is to have each vertex encode the last action occurring prior to it. Specifically, we create multiple copies of a vertex v, each corresponding to a different edge (prior action) that leads to it. If v is a possible starting vertex, we create an additional copy representing no prior action. If there is an edge from v to v', we connect all copies of v to the specific copy of v' whose last action was edge (v, v'). A new edge is called a switching edge if the recorded last actions of its two vertices are of different types (e.g., inspecting different trains), unless one of the two vertices is a "no prior action" vertex. As can be verified, the number of switches of a patrol path in the new graph is the number of switching edges it has. To favor simple patrol paths, we demand a cost $\beta > 0$ for using switching edges. Varying the value of β allows us to trade off between revenue and patrol complexity (average number of switches).

In Figure 6.4(b), we show how to apply this extension using the subgraph shown in the shaded box of Figure 6.4(a). Since there is only one edge leading to $\langle A, 7pm, (6pm)\rangle$, we create one copy of it representing the action of staying at A. There are 3 edges leading to $\langle B, 7pm, (6pm)\rangle$, so we create 3 copies of it representing the actions of taking train from A, staying at B, and taking train from C. The original edges are also duplicated. For example, $\langle B, 7pm, (6pm)\rangle \rightarrow \langle B, 8pm, (6pm)\rangle$ has 3 copies connecting the 3 copies of $\langle B, 7pm, (6pm)\rangle$ to the copy of $\langle B, 8pm, (6pm)\rangle$, representing the staying at B action. Among the three copies, only the "Stay" to "Stay" edge is not a switching edge.



(a)



(b)

Figure 6.4: (a) HDT graph of Example 4 with two starting time points. (b) extension storing the last action occurring.

Given the final HDT graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}\rangle$, I provide an extended linear program formulation in Figure 6.5. We still use x_e to represent the marginal coverage on an original edge $e \in E$, but we now also use y_e to represent the marginal coverage on an HDT graph edge $e \in \mathcal{E}$. Let $\Gamma(e) \subset \mathcal{E}$ be the set of copies of e, then $x_e = \sum_{e' \in \Gamma(e)} y_{e'}$. Let $c_e = 1$ if $e \in \mathcal{E}$ is a switching edge

$$\max_{\mathbf{x},\mathbf{y},\mathbf{u}} \sum_{\lambda\in\Lambda} p_\lambda u_\lambda - \beta \sum_{e\in\mathcal{E}} c_e y_e \tag{6.7}$$

$$\text{s.t. } u_\lambda \leq \min\{\rho_\lambda,\ \varrho_\lambda \sum_{e\in\lambda} x_e f_e\},\ \text{ for all } \lambda \in \Lambda \tag{6.8}$$

$$\sum_{v\in\mathcal{V}^+} y_{(v^+,v)} = \sum_{v\in\mathcal{V}^-} y_{(v,v^-)} \leq \gamma \tag{6.9}$$

$$\sum_{(v',v)\in\mathcal{E}} y_{(v',v)} = \sum_{(v,v^\dagger)\in\mathcal{E}} y_{(v,v^\dagger)},\ \text{ for all } v \in \mathcal{V} \tag{6.10}$$

$$x_e = \sum_{e'\in\Gamma(e)} y_{e'}, \forall e \in E, 0 \leq x_e \leq \gamma, \forall e \in E \tag{6.11}$$

Figure 6.5: Extended Formulation

and 0 otherwise. The set of possible starting vertices $\mathcal{V}^+$ is the set of copies of V^+ that are "no prior action" vertices. The set of possible ending vertices $\mathcal{V}^-$ is the set of all copies of V^-. We again add a dummy source v^+ leading to $\mathcal{V}^+$ and a dummy sink that can be reached from $\mathcal{V}^-$. Because the extended formulation enforces stricter restrictions on patrols allowed than the basic formulation, the LP of Figure 6.5, with β set to 0, provides a tighter upper bound on the optimal revenue than the LP of Figure 6.2.

A path in the HDT graph $\mathcal{G}$ trivially corresponds to a path in the transition graph G, since any edge in $\mathcal{G}$ is a duplicate of some edge in G. Therefore from the optimal solution $\mathbf{y}^*$, we can use the same process described for the basic formulation to construct a mixed patrol strategy. As we can see, this mixed patrol strategy does not have the two issues of the basic formulation. First, the length of any patrol path in the HDT graph is bounded by κ. In addition, since the number of switches in a patrol path equals the number of switching edges in it, the average number of switches of the constructed mixed strategy is equal to $\sum_{e\in\mathcal{E}} c_e y_e^*$, which is penalized in the objective function.

6.2 TRUSTSv2: Stochastic Model for Imperfect Execution

A major drawback of TRUSTSv1 is its vulnerability to execution uncertainty. In real world trials carried out by the LASD, a significant fraction of the executions of pre-generated schedules got interrupted, for a variety of reasons such as writing citations, felony arrests, and handling emergencies. Such interruptions can cause the officers to miss the train that they were supposed to take as part of the schedule. As a result the solution of TRUSTSv1 may not provide instructions on what to do after an interruption occurs. Furthermore, since the TRUSTSv1 model does not take into account such execution uncertainty in its optimization formulation, the quality guarantee of its solution is no longer valid in real world settings.

In this section, I will present TRUSTSv2, the second generation of TRUSTS to address the challenge of execution uncertainty. In Section 6.2.1, I will first present a formal general game-theoretic model for patrolling with dynamic execution uncertainty. Section 6.2.2 provides a solution method for problems where the utilities have additional *separable* structure. Finally, I will explain the details of applying this model to the fare evasion deterrence problem in Section 6.2.3.

6.2.1 Formal Model

A *patrolling game with execution uncertainty* is a two-player Bayesian Stackelberg game, between a leader (the defender) and a follower (the adversary). The leader has γ patrol units, and commits to a randomized daily patrol schedule for each unit. A (naive) patrol schedule consists of a list of commands to be carried out in sequence. Each command is of the form: at time τ, the unit should be at location l, and should execute patrol action a. The patrol action a of the current command, if executed successfully, will take the unit to the location and time of the next command. Each unit faces uncertainty in the execution of each command: delays, or being called to deal with emergencies (possibly at another location). As a result the unit may end up at a location and a time that is different from the intended outcome of the action.

We use Markov Decision Processes (MDPs) as a compact representation to model each individual defender unit's execution of patrols. We emphasize that these MDPs are not the whole game: they only model the defender's interactions with the environment when executing patrols; we will later describe the interaction between the defender and the adversary. Formally, for each defender unit $i \in \{1, \dots, \gamma\}$ we define an MDP (S_i, A_i, T_i, R_i), where

- S_i is a finite set of states. Each state $s_i \in S_i$ is a tuple (l, τ) of the current location of the unit and the current discretized time. We denote by $l(s_i)$ and $\tau(s_i)$ the location and time of s_i, respectively.
- A_i is a finite set of actions. Let $A_i(s_i) \subseteq A_i$ be the set of actions available at state s_i.
- For each $s_i \in S_i$ and each action $a_i \in A_i(s)$, the *default next state* $n(s_i, a_i) \in S_i$ is the intended next state when executing action a_i at s_i. We call a transition (s_i, a_i, s_i') a default transition if $s_i' = n(s_i, a_i)$ and a non-default transition otherwise.
- $T_i(s_i, a_i, s_i')$ is the probability of next state being s_i' if the current state is s_i and the action taken is a_i.
- $R_i(s_i, a_i, s_i')$ is the immediate reward for the defender from the transition (s_i, a_i, s_i'). For example, being available for emergencies (such as helping a lost child) is an important function of the police, and we can take this into account in our optimization formulation by using R_i to give positive rewards for such events.

We assume that the MDP is acyclic: $T_i(s_i, a_i, s'_i)$ is positive only when $\tau(s'_i) > \tau(s_i)$, i.e., all transitions go forward in time. $S_i^+, S_i^- \subseteq S_i$ are two subsets of states where a patrol could start and end respectively. For convenience, we add a dummy source state $s_i^+ \in S_i$ that has actions with deterministic transitions going into each of the states in S_i^+, and analogously a dummy sink state $s_i^- \in S_i$. Thus each patrol of defender i starts at s_i^+ and ends at s_i^-. A patrol execution of i is specified by its *complete trajectory* $t_i = (s_i^+, a_i^+, s_i^1, a_i^1, s_i^2, \ldots, s_i^-)$, which records the sequence of states visited and actions performed. A joint complete trajectory, denoted by $\mathbf{t} = (t_1, \ldots, t_\gamma)$, is a tuple of complete trajectories of all units. Let $\mathcal{X}$ be the finite space of joint complete trajectories.

The immediate rewards R_i are not all the utility received by the defender. The defender also receives rewards from interactions with the adversary. The adversary can be of a set Λ of possible *types* and has a finite set of actions $\mathcal{A}$. The types are drawn from a known distribution, with p_λ the probability of type $\lambda \in \Lambda$. The defender does not know the instantiated type of the adversary, while the adversary does and can condition his decision on his type.

In this general game model, the utilities resulting from defender-adversary interaction could depend arbitrarily on the complete trajectories of the defender units. Formally, for a joint complete trajectory $\mathbf{t}$, the realized adversary type $\lambda \in \Lambda$, and an action of the adversary $\alpha \in \mathcal{A}$, the defender receives utility $\mu(\mathbf{t}, \lambda, \alpha)$, while the adversary receives $\nu(\mathbf{t}, \lambda, \alpha)$.

We are interested in finding the Strong Stackelberg Equilibrium (SSE) of this game, in which the defender commits to a randomized policy which we define next, and the adversary plays a best response to this randomized policy. It is sufficient to consider only pure strategies for the adversary Conitzer and Sandholm [2006]. Finding one SSE is equivalent to the following optimization problem:

$$\max_{\pi} \sum_{\lambda \in \Lambda} p_\lambda E_{\mathbf{t}\sim\pi}[\mu(\mathbf{t}, \lambda, \alpha_\lambda) + \sum_i R_i(t_i)] \tag{6.12}$$

$$\text{s.t. } \alpha_\lambda \in \arg\max_{\alpha_\lambda} E_{\mathbf{t}\sim\pi}[\nu(\mathbf{t}, \lambda, \alpha_\lambda)], \forall \lambda \in \Lambda \tag{6.13}$$

where $R_i(t_i)$ is the total immediate reward from the trajectory t_i, and $E_{\mathbf{t}\sim\pi}[\cdot]$ denotes the expectation over joint complete trajectories induced by defender's randomized policy π.

Whereas MDPs always have Markovian and deterministic optimal policies, in our game the defender's optimal strategy may be non-Markovian because the utilities depend on trajectories, and may be randomized because of interactions with the adversary. The execution of patrols can be potential coupled and decoupled. In coupled execution, patrol units can coordinate with each other; that is, the behavior of unit i at s_i could depend on the earlier joint trajectory of all units. Formally, let $\mathcal{T}_i$ be the set of unit i's *partial trajectories* $(s_i^+, a_i^+, s_i^1, a_i^1, \ldots, s_i')$. A coupled randomized policy is a function $\pi : \prod_i \mathcal{T}_i \times \prod_i A_i \to \boldsymbol{R}$ that specifies a probability distribution over joint actions of units for each joint partial trajectory. Denote by $\varphi(\mathbf{t}; \pi) \in \boldsymbol{R}$ the probability that joint complete trajectory $\mathbf{t} \in \mathcal{X}$ is instantiated under policy π. In decoupled execution, patrol units do not communicate with each other. Formally, a decoupled randomized policy $\pi = (\pi_1, \ldots, \pi_\gamma)$ where for each unit i, $\pi_i : \mathcal{T}_i \times A_i \to \boldsymbol{R}$ specifies a probability distribution over i's actions given each partial trajectory of i. Thus a decoupled randomized policy $(\pi_1, \ldots, \pi_\gamma)$ can be thought of as a coupled randomized policy π' where $\pi'(\mathbf{t}, (a_1, \ldots, a_\gamma)) = \prod_i \pi_i(t_i, a_i)$.

Coupled execution potentially yields higher expected utility than decoupled execution. Suppose the defender wants to protect an important target with at least one unit, and unit 1 is assigned that task. Then if she knows unit 1 is dealing with an emergency and unable to reach that target, she can reroute unit 2 to cover the target. However, coordinating among units presents significant logistical and (as I will explain later) computational burden.

6.2.2 Approach

Since the defender's optimal strategy may be coupled and non-Markovian, i.e., the policy at s could depend on the entire earlier trajectories of all units rather than the current state s. This makes solving the game computationally difficult—the dimension of the space of mixed strategies is exponential in the number of states.

Nevertheless, in many domains, the utilities have additional structure. In Section 6.2.2.1 I will show that under the assumption that the utilities have *separable* structure, it is possible to efficiently compute an SSE of patrolling games with execution uncertainty. In Section 6.2.2.2 I will discuss generating patrol schedules from solutions described in Section 6.2.2.1. In Section 6.2.2.3 I will consider a more general case with *partially separable* utilities.

6.2.2.1 Efficient Computation on Separable Utilities

Consider a coupled strategy π. Denote by $x_i(s_i, a_i, s'_i)$ the marginal probability of defender unit i reaching state s_i, executing action a_i, and ending up at next state s'_i. Formally,

$$x_i(s_i, a_i, s'_i) = \sum_{\mathbf{t} \in \mathcal{X}} \varphi(\mathbf{t}; \pi)\theta(t_i, s_i, a_i, s'_i), \tag{6.14}$$

where the value of the membership function $\theta(t_i, s_i, a_i, s'_i)$ is equal to 1 if trajectory t_i contains transition (s_i, a_i, s'_i) and is equal to 0 otherwise. Let $\mathbf{x} \in \boldsymbol{R}^M$ be the vector of these marginal probabilities, where $M = \sum_i |S_i|^2|A_i|$. Similarly, let $w_i(s_i, a_i)$ be the marginal probability of unit i

reaching s_i and taking action a_i. Let $\mathbf{w} \in \boldsymbol{R}^{\sum_i |S_i||A_i|}$ be the vector of these marginal probabilities. I will show that $\mathbf{w}$ and $\mathbf{x}$ satisfy the linear constraints:

$$x_i(s_i, a_i, s_i') = w_i(s_i, a_i) T_i(s_i, a_i, s_i'), \forall s_i, a_i, s_i' \tag{6.15}$$

$$\sum_{s_i', a_i'} x_i(s_i', a_i', s_i) = \sum_{a_i} w_i(s_i, a_i), \forall s_i \tag{6.16}$$

$$\sum_{a_i} w_i(s_i^+, a_i) = \sum_{s_i', a_i'} x_i(s_i', a_i', s_i^-) = 1, \tag{6.17}$$

$$w_i(s_i, a_i) \geq 0, \forall s_i, a_i \tag{6.18}$$

Lemma 7. *For all coupled randomized policy π, the resulting marginal probabilities $w_i(s_i, a_i)$ and $x_i(s_i, a_i, s_i')$ satisfy constraints* (6.15), (6.16), (6.17), (6.18).

Proof. Constraint (6.15) holds by the definition of transition probabilities of MDPs. Constraint (6.16) holds because both *lhs* and *rhs* equal the marginal probability of reaching state s. Constraint (6.17) holds because by construction, the marginal probability of reaching s_i^+ is 1, and so is the marginal probability of reaching s_i^-. Constraint (6.18) holds because $w_i(s_i, a_i)$ is a probability. $\square$

Intuitively, if we can formulate utilities in terms of $\mathbf{w}$ and $\mathbf{x}$, which have dimensions polynomial in the sizes of the MDPs, this will lead to a much more compact representation of the SSE problem compared to (6.12). It turns out this is possible if the game's utilities are *separable*, which intuitively means that given the adversary's strategy, the utilities of both players are sums of contributions from individual units' individual transitions:

Definition 8. *A patrolling game with execution uncertainty as defined in Section 6.2.1 has* **separable** *utilities if there exist utilities $U_\lambda(s_i, a_i, s_i', \alpha)$ and $V_\lambda(s_i, a_i, s_i', \alpha)$ for each unit i, transition (s_i, a_i, s_i'), $\lambda \in \Lambda$, $\alpha \in \mathcal{A}$, such that for all $\mathbf{t} \in \mathcal{X}$, $\lambda \in \Lambda$, $\alpha \in \mathcal{A}$, the defender's and the adversary's utilities can be expressed as $\mu(\mathbf{t}, \lambda, \alpha) = \sum_i \sum_{s_i, a_i, s_i'} \theta(t_i, s_i, a_i, s_i') U_\lambda(s_i, a_i, s_i', \alpha)$ and $\nu(\mathbf{t}, \lambda, \alpha) = \sum_i \sum_{s_i, a_i, s_i'} \theta(t_i, s_i, a_i, s_i') V_\lambda(s_i, a_i, s_i', \alpha)$, respectively.*

Let $U_\lambda, V_\lambda \in \boldsymbol{R}^{M \times |\mathcal{A}|}$ be the corresponding matrices. Then U_λ, V_λ completely specifies the utility functions μ and ν.

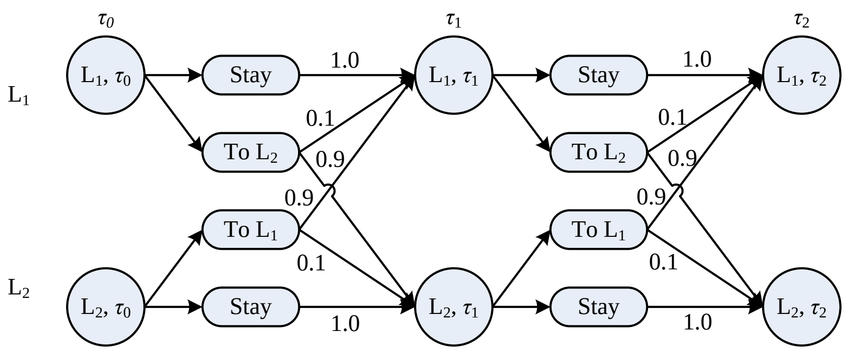


Figure 6.6: Example game with separable utilities.

Example 5. *Consider the following simple example game with one defender unit, whose MDP is illustrated in Figure 6.6. There are six states, shown as circles in the figure, over two locations L_1, L_2 and three time points τ_0, τ_1, τ_2. From states at τ_0 and τ_1, the unit has two actions: to stay at the current location, which always succeeds, and to try to go to the other location, which with probability 0.9 succeeds and with probability 0.1 fails (in which case it stays at the current location). There are 12 transitions in total, which is fewer than the number of complete trajectories (18). There is a single type of adversary who chooses one location between L_1 and L_2 and one time point between τ_1 and τ_2 to attack (τ_0 cannot be chosen). If* *the defender is at that location at that time, the attack fails and both players get zero utility. Otherwise, the attack succeeds, and the adversary gets utility 1 while the defender gets* -1*. In other words, the attack succeeds if and only if it avoids the defender unit's trajectory. It is straightforward to verify that this game has separable utilities: for any transition* (s_i, a_i, s'_i) *in the MDP, let* $V_\lambda(s_i, a_i, s'_i, \alpha)$ *be 1 if* α *coincides with* s'_i *and 0 otherwise. For example, the utility expression for the adversary given trajectory* $((L_1, \tau_0), \mathit{To}\ L_2, (L_1, \tau_1), \mathit{To}\ L_2, (L_2, \tau_2))$ *is* $V_\lambda((L_1, \tau_0), \mathit{To}\ L_2, (L_1, \tau_1), \alpha) + V_\lambda((L_1, \tau_1), \mathit{To}\ L_2, (L_2, \tau_2), \alpha)$*, which gives the correct utility value for the adversary: 1 if* α *equals* (L_1, τ_1) *or* (L_2, τ_2) *and 0 otherwise.*

It is straightforward to show the following.

Lemma 8. *Consider a game with separable utilities. Suppose* $\mathbf{x}$ *is the vector of marginal probabilities induced by the defender's randomized policy* π*. Let* $\mathbf{y}_\lambda \in \mathbb{R}^{|\mathcal{A}|}$ *be a vector describing the mixed strategy of the adversary of type* λ*, with* $y_\lambda(\alpha)$ *denoting the probability of choosing action* α*. Then the defender's and the adversary's expected utilities from their interactions are* $\sum_\lambda p_\lambda \mathbf{x}^T U_\lambda \mathbf{y}_\lambda$ *and* $\sum_\lambda p_\lambda \mathbf{x}^T V_\lambda \mathbf{y}_\lambda$*, respectively.*

In other words, given the adversary's strategy, the expected utilities of both players are linear in the marginal probabilities $x_i(s_i, a_i, s'_i)$. Lemma 8 also applies when (as in an SSE) the adversary is playing a pure strategy, in which case $\mathbf{y}_\lambda$ is a 0-1 integer vector with $y_\lambda(\alpha) = 1$ if α is the action chosen. We can thus use this compact representation of defender strategies to rewrite the formulation for SSE (6.12) as a polynomial-sized optimization problem.

$$\max_{\mathbf{w},\mathbf{x},\mathbf{y}} \sum_{\lambda\in\Lambda} p_\lambda \mathbf{x}^T U_\lambda \mathbf{y}_\lambda + \sum_{i=1}^{\gamma} \sum_{s_i,a_i,s_i'} x_i(s_i,a_i,s_i')R_i(s_i,a_i,s_i') \tag{6.19}$$

$$\text{s.t. constraints (6.15), (6.16), (6.17), (6.18)}$$

$$\sum_{\alpha} y_\lambda(\alpha) = 1, \quad y_\lambda(\alpha) \in \{0,1\} \tag{6.20}$$

$$\mathbf{y}_\lambda \in \arg\max_{\mathbf{y}'_\lambda} \mathbf{x}^T V_\lambda \mathbf{y}'_\lambda \tag{6.21}$$

As I will show in Section 6.2.2.2, given a solution $\mathbf{w}, \mathbf{x}$ to (6.19), we can calculate a *decoupled* policy that matches the marginals $\mathbf{w}, \mathbf{x}$. Compared to (6.12), the optimization problem (6.19) has exponentially fewer dimensions; in particular the numbers of variables and constraints are polynomial in the sizes of the MDPs. Furthermore, existing methods for solving Bayesian Stackelberg games can be directly applied to (6.19) such as DOBSS Paruchuri et al. [2008] or HUNTER in this thesis Yin and Tambe [2012].

For the special case of $U_\lambda + V_\lambda = \mathbf{0}$ for all λ, i.e., when the interaction between defender and adversary is zero-sum, the above SSE problem can be formulated as a linear program (LP):

$$\max_{\mathbf{w},\mathbf{x},\mathbf{u}} \sum_{\lambda\in\Lambda} p_\lambda u_\lambda + \sum_{i} \sum_{s_i,a_i,s_i'} x_i(s_i,a_i,s_i')R_i(s_i,a_i,s_i') \tag{6.22}$$

$$\text{s.t. constraints (6.15), (6.16), (6.17), (6.18)}$$

$$u_\lambda \le \mathbf{x}^T U_\lambda \mathbf{e}_\alpha, \forall \lambda \in \Lambda,\ \alpha \in \mathcal{A}, \tag{6.23}$$

where $\mathbf{e}_\alpha$ is the basis vector corresponding to adversary action α. This LP is similar to the maximin LP for a zero-sum game with the utilities given by U_λ and V_λ, except that an additional term $\sum_i \sum_{s_i,a_i,s'_i} x_i(s_i, a_i, s'_i)R_i(s_i, a_i, s'_i)$ representing defender's expected utilities from immediate rewards is added to the objective. One potential issue arises: because of the extra defender utilities from immediate rewards, the entire game is no longer zero-sum. Is it still valid to use the above maximin LP formulation? It turns out that the LP is indeed valid, as the immediate rewards do not depend on the adversary's strategy.

Proposition 5. *If the game has separable utilities and $U_\lambda + V_\lambda = 0$ for all λ, then a solution of the LP* (6.22) *is an SSE.*

Proof. We can transform this game to an equivalent zero-sum Bayesian game whose LP formulation is equivalent to (6.22). Specifically, given the non-zero-sum Bayesian game Γ specified above, consider the Bayesian game Γ' with the following "meta" type distribution for the second player: for all $\lambda \in \Lambda$ of Γ there is a corresponding type $\lambda' \in \Lambda'$ in Γ', with probability $p_{\lambda'} = 0.5p_\lambda$, with the familiar utility functions; and there is a special type $\phi \in \Lambda'$ with probability $p_\phi = 0.5$, whose action does not affect either player's utility. Specifically the utilities under the special type ϕ are $\mu(\mathbf{t}, \phi, \alpha) = \sum_i \sum_{s_i,a_i,s'_i} \theta(t_i, s_i, a_i, s'_i)R_i(s_i, a_i, s'_i)$ and $\nu(\mathbf{t}, \phi, \alpha) = -\sum_i \sum_{s_i,a_i,s'_i} \theta(t_i, s_i, a_i, s'_i)R_i(s_i, a_i, s'_i)$. The resulting game Γ' is zero-sum, with the defender's utility exactly half the objective of (6.22). Since for zero-sum games maximin strategies and SSE coincide, a solution of the LP (6.22) is an optimal SSE marginal vector for the defender of Γ'. On the other hand, if we compare the induced normal forms of Γ and Γ', the only difference is that for the adversary the utility $-0.5\sum_{e\in E^*} U_e x_e$ is added, which does not depend on the adversary's strategy. Therefore Γ and Γ' have the same set of SSE, which implies that a solution of the LP is an SSE of Γ. $\square$

6.2.2.2 Generating Patrol Schedules

The solution of (6.19) does not yet provide a complete specification of what to do. We ultimately want an explicit procedure for generating the patrol schedules. We define a *Markov strategy* π to be a decoupled strategy $(\pi_1, \ldots, \pi_\gamma)$, $\pi_i : S_i \times A_i \to \boldsymbol{R}$, where the distribution over next actions depends only on the current state. Proposition 6 below shows that given $\mathbf{w}, \mathbf{x}$, there is a simple procedure to calculate a Markov strategy that matches the marginal probabilities. This implies that if $\mathbf{w}, \mathbf{x}$ is the optimal solution of (6.19), then the corresponding Markov strategy π achieves the same expected utility. I have thus shown that for games with separable utilities it is sufficient to consider Markov strategies.

Proposition 6. *Given* $\mathbf{w}, \mathbf{x}$ *satisfying constraints* (6.15) *to* (6.18)*, construct a Markov strategy* π *as follows: for each* $s_i \in S_i$*, for each* $a_i \in A_i(s_i)$*,* $\pi_i(s_i, a_i) = \frac{w_i(s_i,a_i)}{\sum_{a'_i} w_i(s_i,a'_i)}$*. Suppose the defender plays* π*, then for all unit* i *and transition* (s_i, a_i, s'_i)*, the probability that* (s_i, a_i, s'_i) *is reached by* i *equals* $x_i(s_i, a_i, s'_i)$.

Sketch. Such a Markov strategy π induces a Markov chain over the states S_i for each unit i. It can be verified by induction that the resulting marginal probability vector matches $\mathbf{x}$. $\square$

In practice, directly implementing a Markov strategy requires the unit to pick an action according to the randomized Markov strategy at each time step. This is possible when units can consult a smart-phone app that stores the strategy, or can communicate with a central command. However, in certain domains such requirement on computation or communication at each time step places additional logistical burden on the patrol unit. To avoid unnecessary computation or communication at every time step, it is desirable to have a deterministic schedule (i.e., a pure strategy) from the Markov strategy. Without execution uncertainty, a pure strategy can be specified by the a complete trajectory for each unit. However, this no longer works in the case with execution uncertainty.

I will thus begin by defining a Markov pure strategy, which specifies a deterministic choice at each state.

Definition 9. *A* **Markov pure strategy q** *is a tuple* $(q_1, \ldots, q_\gamma)$ *where for each unit i,* $q_i : S_i \rightarrow A_i$.

Given a Markov strategy π, we sample a Markov pure strategy **q** as follows: for each unit i and state $s_i \in S_i$, sample an action a_i as $q_i(s_i)$ according to π_i. This procedure is correct since each state in i's MDP is visited at most once and thus q_i exactly simulates a walk from s_i^+ on the Markov chain induced by π_i.

To directly implement a Markov pure strategy, the unit needs to remember the entire mapping **q** or receives the action from the central command at each time step. A logistically more efficient way is for the central command to send the unit a trajectory assuming perfect execution, and only after a non-default transition happened does the unit communicates with the central command to get a new trajectory starting from the current state. Formally, given $s_i \in S_i$ and q_i, we define the *optimistic* trajectory from s_i induced by q_i to be $(s_i, q_i(s_i), n(s_i, q_i(s_i)), \ldots s^-)$, i.e, the trajectory assuming it always reaches its default next state. Given a Markov pure strategy **q**, the following procedure for each unit i exactly simulates **q**: (i) central command gives unit i the optimistic trajectory from s^+ induced by q_i; (ii) unit i follows the trajectory until the terminal state s^- is reached or some unexpected event happens and takes i to state s'_i; (iii) central command sends the new optimistic trajectory from s'_i induced by q_i to unit i and repeat from step (ii).

6.2.2.3 Coupled Execution: Cartesian Product MDP

Without the assumption of separable utilities, it is no longer sufficient to consider decoupled Markov strategies of individual units' MDPs. We therefore need to create a new MDP that captures the joint execution of patrols by all units. For simplicity of exposition, we look at the case with two defender units. Then a state in the new MDP corresponds to the tuple (location of unit 1, location of unit 2, time). An action in the new MDP corresponds to a tuple (action of unit 1, action of unit 2). Formally, if unit 1 has an action a_1 at state $s_1 = (l_1, \tau)$ that takes her to $s'_1 = (l'_1, \tau')$ with probability $T_1(s_1, a_1, s'_1)$, and unit 2 has an action a_2 at state $s_2 = (l_2, \tau)$ that takes her to $s'_2 = (l'_2, \tau')$ with probability $T_2(s_2, a_2, s'_2)$, we create in the new MDP an action $a_\times = (a_1, a_2)$ from state $s_\times = (l_1, l_2, \tau)$ that transitions to $s'_\times = (l'_1, l'_2, \tau')$ with probability $T_\times(s_\times, a_\times, s'_\times) = T_1(s_1, a_1, s'_1)T_2(s_2, a_2, s'_2)$. The immediate rewards $R_\times$ of the MDP are defined analogously. We call the resulting MDP $(S_\times, A_\times, T_\times, R_\times)$ the Cartesian Product MDP.

An issue arises when at state $s_\times$ the individual units have transitions of different time durations. For example, unit 1 rides a train that takes 2 time steps to reach the next station while unit 2 stays at a station for 1 time step. During these intermediate time steps only unit 2 has a "free choice". How do we model this on the Cartesian Product MDP? One approach is to create new states for the intermediate time steps. For example, suppose at location L_A at time 1 a non-default transition takes unit 1 to location L_A at time 3. We modify unit 1's MDP so that this transition ends at a new state $(L^1_A, 2) \in S_1$, where L^1_A is a "special" location specifying that the unit will become alive again at location L_A in one more time step. There is only one action from $(L^1_A, 2)$, with only one possible next state $(L_A, 3)$. Once we have modified the individual units' MDPs so that all transitions take exactly one time step, we can create the Cartesian Product MDP as described in the previous paragraph.

Like the units' MDPs, the Cartesian Product MDP is also acyclic. Therefore we can analogously define marginal probabilities $w_\times(s_\times, a_\times)$ and $x_\times(s_\times, a_\times, s'_\times)$ on the Cartesian Product MDP. Let $\mathbf{w}_\times \in \boldsymbol{R}^{|S_\times||A_\times|}$ and $\mathbf{x}_\times \in \boldsymbol{R}^{|S_\times|^2|A_\times|}$ be the corresponding vectors. Utilities generally cannot be expressed in terms of $\mathbf{w}_\times$ and $\mathbf{x}_\times$. We consider a special case in which utilities are *partially separable*:

Definition 10. *A patrolling game with execution uncertainty has* **partially separable** *utilities if there exist $U_\lambda(s_\times, a_\times, s'_\times, \alpha)$ and $V_\lambda(s_\times, a_\times, s'_\times, \alpha)$ for each transition $(s_\times, a_\times, s'_\times)$, $\lambda \in \Lambda$, $\alpha \in \mathcal{A}$, such that for all $\mathbf{t} \in \mathcal{X}$, $\lambda \in \Lambda$, $\alpha \in \mathcal{A}$, the defender's and the adversary's utilities can be expressed as $\mu(\mathbf{t}, \lambda, \alpha) = \sum_{s_\times, a_\times, s'_\times} \theta_\times(\mathbf{t}, s_\times, a_\times, s'_\times) U_\lambda(s_\times, a_\times, s'_\times, \alpha)$ and $\nu(\mathbf{t}, \lambda, \alpha) = \sum_{s_\times, a_\times, s'_\times} \theta_\times(\mathbf{t}, s_\times, a_\times, s'_\times) V_\lambda(s_\times, a_\times, s'_\times, \alpha)$, respectively.*

Partially separable utilities is a weaker condition than separable utilities, as now the expected utilities may not be sums of contributions from individual units. When utilities are partially separable, we can express expected utilities in terms of $\mathbf{w}_\times$ and $\mathbf{x}_\times$ and find an SSE by solving an optimization problem analogous to (6.19). From the optimal $\mathbf{w}^*_\times$, we can get a Markov strategy $\pi^*_\times(s_\times, a_\times) = \frac{w^*_\times(s_\times, a_\times)}{\sum_{a'_\times} w^*_\times(s, a'_\times)}$, which is provably the optimal coupled strategy.

This approach cannot scale up to a large number of defender units, as the size of $S_\times$ and $A_\times$ grow exponentially in the number of units. In particular the dimension of the Markov policy $\pi_\times$ is already exponential in the number of units. To overcome this we will need a more compact representation of defender strategies. One approach is to use decoupled strategies. Although no longer optimal in general, I will show in Section 6.2.3 that decouple strategies can provide a good approximation in the fare evasion deterrence problems.

6.2.3 Application to Fare Evasion Deterrence

I will now explain how the techniques proposed in Section 6.2.2 can be applied to the fare evasion deterrence problem in transit systems. As we will see in Section 6.2.3.1, although the utilities in this domain are not separable, we are able to upper bound the defender utilities by separable utilities, allowing efficient computation. The solution given in Section 6.2.3.1 is a Markov strategy which can be used to sample Markov pure strategies as described earlier in Section 6.2.2.2. However implementation of a Markov pure strategy with tens of thousands of states is nontrivial in practice. In Section 6.2.3.2, I will demonstrate a smart-phone app solution that facilitates TRUSTSv2 deployment in real-world transit systems.

6.2.3.1 Linear Program Formulation

Similar to the extended formulation on *history-duplicate transition* graph given in Section 6.1.2, a state here comprises the current station and time of a unit, as well as necessary history information such as starting time and prior patrol action. At any state, a unit may stay at her current station to conduct an *in-station* operation for some time or she can ride a train to conduct an *on-train* operation when her current time coincides with the train schedule. Due to execution uncertainty, a unit may end up at a state other than the intended outcome of the action. For ease of analysis, I assume a single type of unexpected event which delays a patrol unit for some time beyond the intended execution time. Specifically, I assume for any fare check operation taken, there is a probability η that the operation will be delayed, i.e., staying at the same station (for *in-station* operations) or on the same train (for *on-train* operations) involuntarily for some time. Furthermore, I assume that units will be involved with events unrelated to fare enforcement and thus will not check fares during any delayed period of an operation. Intuitively, a higher chance of delay leads to less time spent on fare inspection.

The riders (adversaries) and the objective remain unchanged from TRUSTSv1. Recall riders have multiple types, each corresponds a fixed route. A rider observes the likelihood of being checked and makes a binary decision between buying and not buying the ticket. If the rider of type λ buys the ticket, he pays a fixed ticket price ρ_λ. Otherwise, he rides the train for free but risks the chance of being caught and paying a fine of $\varrho_\lambda > \rho_\lambda$. The LASD's objective is set to maximize the overall revenue of the whole system including ticket sales and fine collected, essentially forming a zero-sum game.

Recall in TRUSTSv1, we define the fare check effectiveness f for each atomic patrol action represented by an edge in the transition graph. However, in TRUSTSv2 the fare check operation performed is determined by the actual transition rather than the action taken. Therefore we will define the effectiveness of a transition (s, a, s') against a rider type λ, $f_\lambda(s, a, s')$, as the percentage of riders of type λ checked by transition (s, a, s'). Note $f_\lambda(s, a, s')$ is non-zero if and only if the actual operation in transition (s, a, s') intersects with the route λ takes. Following the same argument as in Section 6.1.1, the probability that a joint complete trajectory $\mathbf{t}$ detects evader λ is the sum of f_λ over all transitions in $\mathbf{t} = (t_1, \ldots, t_\gamma)$ capped at one:

$$\Pr(\mathbf{t}, \lambda) = \min\{\sum_{i=1}^{\gamma} \sum_{s_i, a_i, s'_i} f_\lambda(s_i, a_i, s'_i)\theta(t_i, s_i, a_i, s'_i), 1\}. \tag{6.24}$$

For type λ and joint trajectory $\mathbf{t}$, the LASD receives ρ_λ if the rider buys the ticket and $\varrho_\lambda \cdot \Pr(\mathbf{t}, \lambda)$ otherwise. The utilities in this domain are indeed not separable — even though multiple units (or even a single unit) may detect a fare evader multiple times, the evader can only be fined once. As a result, neither players' utilities can be computed directly using marginal probabilities $\mathbf{x}$ and $\mathbf{w}$. Instead, we upper bound the defender utility by overestimating the detection probability using marginals as the following:

$$\widehat{\Pr(\mathbf{x}, \lambda)} = \sum_{i=1}^{\gamma} \sum_{s_i, a_i, s_i'} f_\lambda(s_i, a_i, s_i) x_i(s_i, a_i, s_i'). \tag{6.25}$$

Equation (6.25) leads to the following upper bound LP for the fare evasion deterrence problem:

$$\max_{\mathbf{x},\mathbf{w},\mathbf{u}} \sum_{\lambda \in \Lambda} p_\lambda u_\lambda + \sum_{i=1}^{\gamma} \sum_{s_i, a_i, s_i'} R_i(s_i, a_i, s_i') \tag{6.26}$$

$$\text{s.t.} \quad \text{constraints (6.15), (6.16), (6.17), (6.18)}$$

$$u_\lambda \leq \min\{\rho_\lambda,\ \varrho_\lambda \cdot \widehat{\Pr(\mathbf{x}, \lambda)}\}, \text{ for all } \lambda \in \Lambda \tag{6.27}$$

We prove the claims above by the following two propositions.

Proposition 7. $\widehat{\Pr(\mathbf{x}, \lambda)}$ *is an upper bound of the true detection probability of any coupled strategy with marginals* $\mathbf{x}$.

Proof. Consider a coupled strategy π. Recall that $\varphi(\mathbf{t}; \pi) \in \boldsymbol{R}$ is the probability that joint trajectory $\mathbf{t} \in \mathcal{X}$ is instantiated. For rider type λ, the true detection probability is $\Pr(\pi, \lambda) =$

$\sum_{\mathbf{t}\in\mathcal{X}} \varphi(\mathbf{t};\pi)\Pr(\mathbf{t},\lambda)$. Relaxing $\Pr(\mathbf{t},\lambda)$ by removing the cap at 1 in Equation (6.24) and applying Equation (6.14) we have,

$$\begin{aligned}\Pr(\pi,\lambda) &\le \sum_{\mathbf{t}\in\mathcal{X}} \varphi(\mathbf{t};\pi) \sum_{i=1}^{\gamma} \sum_{s_i,a_i,s_i'} f_\lambda(s_i,a_i,s_i')\theta(t_i,s_i,a_i,s_i') \\ &= \sum_{i=1}^{\gamma} \sum_{s_i,a_i,s_i'} f_\lambda(s_i,a_i,s_i') \sum_{\mathbf{t}\in\mathcal{X}} \varphi(\mathbf{t};\pi)\theta(t_i,s_i,a_i,s_i') \\ &= \sum_{i=1}^{\gamma} \sum_{s_i,a_i,s_i'} f_\lambda(s_i,a_i,s_i') x_i(s_i,a_i,s_i') = \widehat{\Pr(\mathbf{x},\lambda)}. \quad \square\end{aligned}$$

$\square$

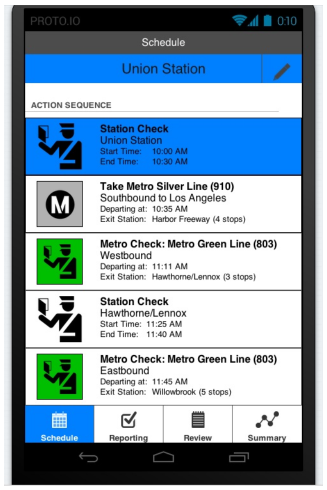
Proposition 8. *LP* (6.26) *provides an upper bound of the optimal coupled strategy.*

Proof. Let $\mathbf{x}^*$ and $\mathbf{w}^*$ be the marginal coverage and u_λ^* be the value of the patroller against rider type λ in the optimal coupled strategy π^*. It suffices to show that $\mathbf{x}^*$, $\mathbf{w}^*$, and $\mathbf{u}^*$ is a feasible point of the LP. From Lemma 7, we already know $\mathbf{x}^*$ and $\mathbf{w}^*$ must satisfy constraints (6.15) to (6.18). Furthermore, we have $u_\lambda^* \le \rho_\lambda$ since the rider pays at most the ticket price. Finally, $u_\lambda^* \le \varrho_\lambda \cdot \widehat{\Pr(\mathbf{x},\lambda)}$ since $\widehat{\Pr(\mathbf{x},\lambda)}$ is an overestimate of the true detection probability. $\square$

Intuitively, LP (6.26) relaxes the utility functions by allowing an evader to be fined multiple times (instead of only once in reality) during a single trip. The relaxed utilities are indeed separable and thus the relaxed problem can be efficiently solved. Since the solution returned $\mathbf{x}^*$ and $\mathbf{w}^*$ satisfy constraints (6.15) to (6.18), we can construct a Markov strategy from $\mathbf{w}^*$ as described in Section 6.2.2.2. The Markov strategy provides an approximate solution to the original problem, whose actual value can be evaluated using Monte Carlo simulation.

6.2.3.2 Metro App: Smart-Phone Implementation

In order to implement the TRUSTSv2 approach in real-world transit systems, the Metro App presented in this section is being developed to work in accordance with TRUSTSv2 to (i) provide officers with patrol policies generated by TRUSTSv2, (ii) provide recovery from schedule interruptions, and (iii) collect patrol data. In this section, I will present how the Metro App will interface with the user and TRUSTSv2 component to provide patrol officers with real-time TRUSTSv2-generated patrol schedules and collect reporting data from the patrol officer's shift. Moreover I will discuss the features of Metro App and user interface design, and the benefits expected from deploying TRUSTSv2 in the Los Angeles Metro System.



Union Station

ACTION SEQUENCE

Station Check
Union Station
Start Time: 10:00 AM
End Time: 10:30 AM

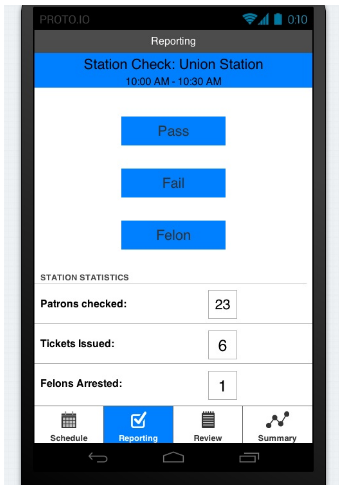
Take Metro Silver Line (910)
Southbound to Los Angeles
Departing at: 10:35 AM
Exit Station: Harbor Freeway (4 stops)

Metro Check: Metro Green Line (803)
Westbound
Departing at: 11:11 AM
Exit Station: Hawthorne/Lennox (3 stops)

Station Check
Hawthorne/Lennox
Start Time: 11:25 AM
End Time: 11:40 AM

Metro Check: Metro Green Line (803)
Eastbound
Departing at: 11:45 AM
Exit Station: Willowbrook (5 stops)

(a) Schedule view

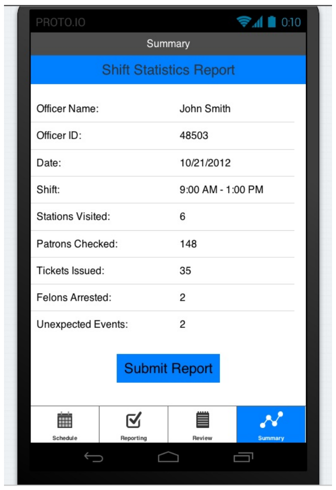


Station Check: Union Station
10:00 AM - 10:30 AM

STATION STATISTICS

Patrons checked:	23
Tickets Issued:	6
Felons Arrested:	1

(b) Reporting view



Shift Statistics Report

Officer Name:	John Smith
Officer ID:	48503
Date:	10/21/2012
Shift:	9:00 AM - 1:00 PM
Stations Visited:	6
Patrons Checked:	148
Tickets Issued:	35
Felons Arrested:	2
Unexpected Events:	2

(c) Summary view

Figure 6.7: Metro App user interface.

The Metro App is a software agent carried by each patrol officer that provides an interface for interaction between the user and TRUSTSv2. The Metro App provides three principal features: a TRUSTSv2-generated patrol schedule for the current shift, a tracking system for reporting passenger violations, and a shift statistics summary report. At the beginning of an officer's shift, the

METRO APP queries the database for the user's patrol strategy (a Markov pure strategy) for the current shift. From the patrol strategy, the METRO APP displays a schedule of the user's current and upcoming patrol actions in "Schedule View", shown in Figure 6.7(a). Implementing recovery from unexpected events in the real world that cause the officer to fall off schedule, "Schedule View" allows the officer to manually set their current location, triggering the app to dynamically update their schedule based on the officer's location. The new updated schedule is obtained from the Markov pure strategy assuming no unexpected events will happen as I have explained in Section 6.2.2.2.

The METRO APP also allows patrol officer to view and record passenger violations, such as fare evasion, for the current patrol action using Reporting View, as shown in Figure 6.7(b). Officers can also view and edit the passenger violations reported for past actions in Summary View, shown in Figure 6.7(c). In Summary View, the officer can also view and submit their METRO APP-generated shift statistics summary report, including all unexpected events and violations reported by the officer throughout the shift, to the TRUSTS database. Through analysis on this collected patrol data, we expect to gain valuable insight on the Los Angeles Metro patrolling domain, such as follower behavior patterns, and better evaluate the effectiveness of TRUSTS deployment in a real transit system. In addition, as many transit system security departments manually enter violations data, METRO APP can eliminate this inefficiency by automatically submitting the collected data to the security department. Furthermore, this collected data will also benefit transit system security departments that conduct their own analysis on patrol system performance and the transit system.

6.3 Experimental Results

In this Section, I will present experimental evaluation of TRUSTS based on real metro schedules and rider traffic data provided by the LASD. For both TRUSTSv1 and TRUSTSv2, I solved the LP with history duplication using CPLEX 12.2 on a standard 2.8GHz machine with 4GB memory. I will first describe the data sets I used, followed by simulation results.

6.3.1 Data Sets

I created four data sets, each based on a different Los Angeles Metro Rail line: Red (including Purple), Blue, Gold, and Green. For each line, I created its transition graph using the corresponding timetable from `http://www.metro.net`. Implementing TRUSTSv1 and TRUSTSv2 requires a fine-grain ridership distribution of potential fare evaders (recall that a rider type corresponds to a 4-tuple of boarding station / time and disembarking station / time).

In my experiments, I assumed that potential fare evaders were evenly distributed among the general population and created the required fine-grained rider distribution using hourly boarding and alighting counts provided by the Los Angeles Sheriff Department. Suppose the percentage of riders boarding in hour i is d_i^+ and the percentage of riders alighting in hour i is d_i^-. Denote the set of those that board in hour i by Λ_i^+ and that alight in hour i by Λ_i^-. Then it is necessary to compute a fine-grained ridership distribution $\mathbf{p}$ to match the hourly boarding and alighting percentages, i.e., to find a point within the following convex region Ω,

$$\Omega = \{\mathbf{p} | \mathbf{p} \geq \mathbf{0} \wedge \sum_{\lambda \in \Lambda_i^+} p_\lambda = d_i^+ \wedge \sum_{\lambda \in \Lambda_i^-} p_\lambda = d_i^-, \forall i\}.$$

For simplicity, I estimated the fare evader distribution by finding the analytic center of Ω, i.e., $\mathbf{p}^* = \arg\min_{\mathbf{p}\in\Omega} \sum_{\lambda\in\Lambda} -\log(p_\lambda)$, which can be efficiently computed.

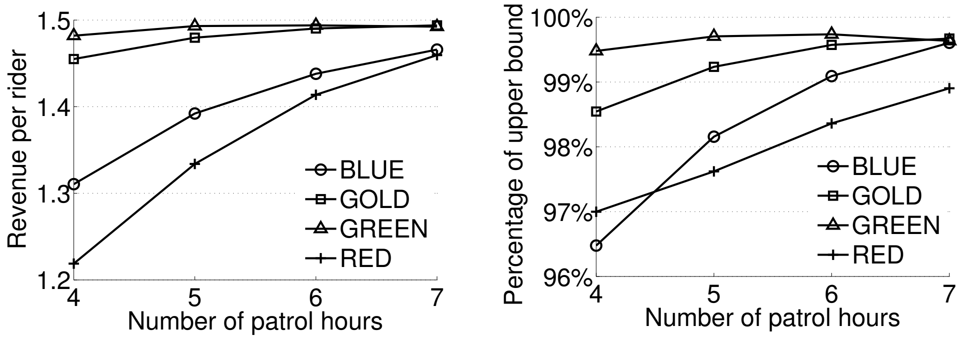
The inspection effectiveness f of a patrol action was adjusted according to the ridership volume intersected. f is capped at 0.5 to capture the fact that the inspector cannot switch between cars while the train is moving. (Trains contain at least two cars.) In the initial batch of experiments on TRUSTSv1 (Section 6.3.2), f was estimated based on the assumption that 10 passengers can be inspected per minute. In subsequent experiments on TRUSTSv2 (Section 6.3.4), this inspection rate was reduced to 3 passengers per minute to account for longer inspection time on tap card users. While this modeling discrepancy makes the results in the two subsections not directly comparable, the experiments conducted within each section were self-consistent and the comparison between TRUSTSv1 and TRUSTSv2 with exactly the same modeling parameters was given in Section 6.3.4. The ticket fare was set to \$1.5 (the actual current value) while the fine was set to \$100. (Fare evaders in Los Angeles can be fined \$200, but they also may be issued warnings.) If we could increase the fine dramatically the riders would have much less incentive for fare evasion, and we could achieve better revenue. However a larger fine is infeasible legally. Table 6.1 summarizes the detailed statistics for the Metro lines.

Line	Stops	Trains	Daily Riders	Types
Red	16	433	149991.5	26033
Blue	22	287	76906.2	46630
Gold	19	280	30940.0	41910
Green	14	217	38442.6	19559

Table 6.1: Statistics of Los Angeles Metro lines.

6.3.2 Simulation Results of TRUSTSv1

Throughout this set of experiments, I fixed γ to 1. In the first set of experiments, I fixed penalty β to 0 (no penalty for using patrol paths with more switches), and varied the maximum number of hours that an inspector can patrol from 4 to 7 hours. To create the HDT graph, I took one starting time point every hour.



(a) (b)

Figure 6.8: Solution quality of TRUSTSv1: (a) Per passenger revenue of the computed mixed strategy (b) Percentage of the solution value compared to the LP upper bound.

Figure 6.8(a) shows the expected revenue per rider of the mixed patrol strategy generated by TRUSTSv1, which is the total revenue divided by the number of daily riders. Since the LP only returns an upper bound of the attainable revenue, the true expected revenue of the mixed patrol strategy was computed by evaluating the riders' best responses for all rider types. A rider can always pay the ticket price for \$1.5 and will only evade the ticket when the expected fine is lower. Hence the theoretical maximum achievable value is \$1.5, which is achieved when every rider purchases a ticket. As we can see, the per-rider revenue increases as the number of patrol hours increases, almost converging to the theoretical upper bound of \$1.5 for the Gold and Green line. Specifically, a 4-hour patrol strategy already provides reasonably good expected value: 1.31 for the Blue line (87.4% of the maximum), 1.45 for the Gold line (97.0%), 1.48 for the Green line

(98.8%), and 1.22 for the Red line (81.3%). Among the four lines, the Red line has the lowest revenue per rider. This is because the effectiveness of fare inspection decreases as the volume of daily riders increases, and the Red line has significantly higher number of daily riders than the other lines.

I depict in Figure 6.8(b) the percentage of the true expected revenue vs. the theoretical upper bound returned by the LP. Strategies generated by our method are near optimal; for example, our 4-hour strategies for the Blue, Gold, Green, and Red lines provided expected revenues of 96.5%, 98.5%, 99.5%, and 97.0% of the upper bound (and thus at least as much of the optimum), respectively.

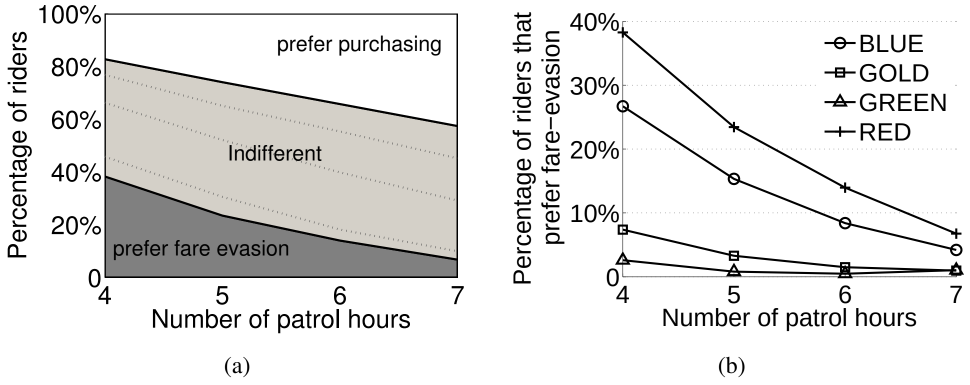


Figure 6.9: Fare evasion analysis of TRUSTSv1: (a) Evasion tendency distribution of the Red line (b) Percentage of riders that prefer fare evasion.

To study riders' responses to the computed strategy, I partitioned the entire population of riders into three groups depending on their expected fine if fare-evading: riders who prefer purchasing tickets (expected fine is greater than 1.7—13.3% above the ticket price), riders who prefer fare evasion (expected fine is less than 1.3—13.3% below the ticket price), and indifferent riders (expected fine is between 1.3 and 1.7). In Figure 6.9(a), I show the distribution of the three groups against the strategies computed for the Red line. The three dashed lines inside the region of indifferent riders represent, from top to bottom, the percentages of riders whose expected fine is less than 1.6, 1.5, and 1.4, respectively. As the number of patrol hours increases from 4 to 7, the percentage of riders who prefer fare evasion decreases from 38% to 7%, the percentage of riders who prefer purchasing tickets increases from 17% to 43%, and the percentage of indifferent riders remains stable between 45% and 50%.

Zooming in on the fare evasion, Figure 6.9(b) shows the percentage of riders who preferred fare evasion against the patrol strategies computed. As we can see, this percentage decreased almost linearly in the number of additional patrol hours beyond 4. Our 7-hour patrol strategy lowered this percentage to 4.2% for the Blue line, 0.01% for the Gold line, 0.01% for the Green line, and 6.8% for the Red line. Again, due to having the highest daily volume, the Red line had the highest percentage of riders who preferred fare evasion.

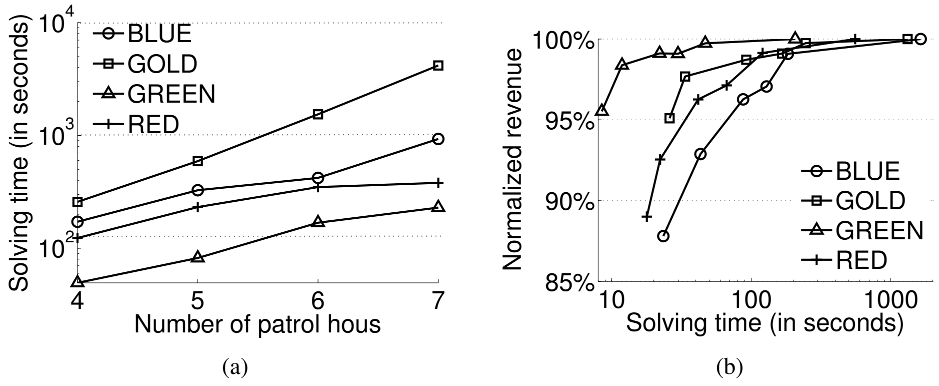


Figure 6.10: Runtime analysis of TRUSTSv1: (a) Runtime of solving the LP by CPLEX (b) Tradeoffs between optimality and runtime.

Figure 6.10(a) shows the runtime required by CPLEX to solve the LPs created. As we can see, the runtime increased as the number of patrol hours increased for all the metro lines. This is because the size of the HDT graph constructed is roughly proportional to the maximum length of the patrols, and a larger HDT graph requires an LP with more variables and constraints. Among the four lines, the Red and the Green lines have significantly fewer types, and are thus easier to solve than the other two lines.

To further study the tradeoff between solution quality and runtime efficiency, I varied the interval of taking starting time points. I fixed the patrol length κ to 4 hours and penalty parameter β to 0. For each line, I tested 6 interval settings ranging from 0.5 hour to 4 hours. In Figure 6.10(b), the x-axis is the runtime (in log-scale) and the y-axis is the normalized revenue against the expected revenue of 0.5-hour interval within each line. For each line, a data point from left to right corresponds to 4, 3, 2, 1.5, 1, and 0.5 hour(s) interval respectively. Increasing the runtime always led to a better solution; however, the quality gain was diminishing. For example, for the Blue line, it took 20 seconds of additional runtime to increase the solution quality from 87.9% (4 hours) to 92.9% (3 hours), whereas it took 1456 seconds of additional runtime to increase the solution quality from 99.1% (1 hour) to 100% (0.5 hour).

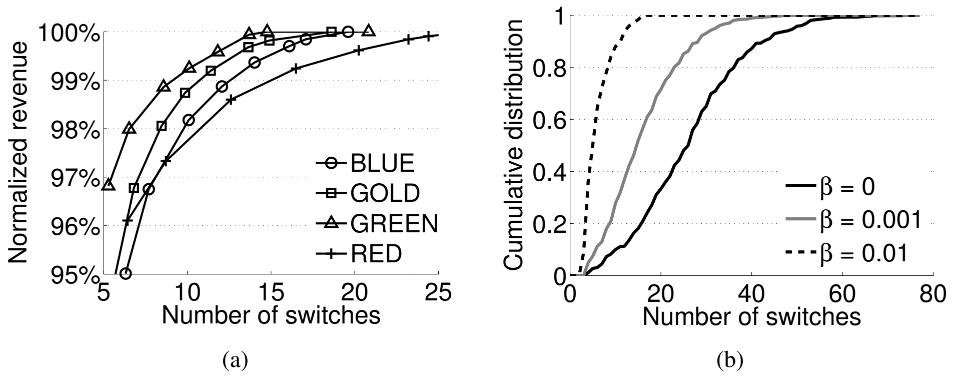


Figure 6.11: Reducing number of switches: (a) Tradeoffs between optimality and patrol preference (b) Cumulative probability distribution of the number of switches for the Red line.

In the final experiment, I varied the penalty β, trading off between the solution quality and the average number of switches. I fixed the patrol length κ to 4 hours and starting time interval δ to one hour. For each line, I tested 7 penalty settings from $\beta = 0$ to $\beta = 0.01$. Figure 6.11(a) plots the average number of switches against the normalized revenue against the expected revenue of $\beta = 0$ within each line. For all lines, higher β values led to both lower solution quality and fewer number of switches. For example, the average number of switches in the solution of the highest revenue ($\beta = 0$) ranged from 18.6 (Gold line) to 26.7 (Red line). However, by allowing 3% quality loss, this number could be lowered to less than 10 for all the four lines.

To further understand the patrol paths returned in these solutions, I show, in Figure 6.11(b), the cumulative probability distributions of the number of switches for the Red line given 3 settings of β: 0, 0.001, and 0.01. Choosing a lower β tended to lead to more complex patrol paths. For example, the solution of $\beta = 0$ used patrol paths whose number of switches is greater than 20 with 68.9% probability; the solution of $\beta = 0.001$ (99.7% of the optimum) only used such paths with 31.2% probability. And the solution of $\beta = 0.01$ (97.0% of the optimum) never used patrol paths that had more than 20 switches.

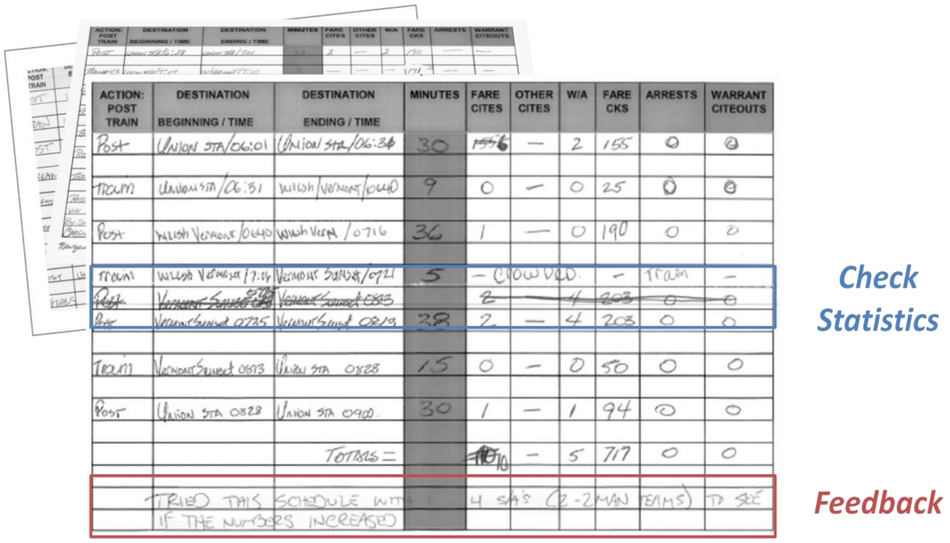
6.3.3 Field Trials of TRUSTSv1

In addition to simulations, some real world trials of TRUSTSv1-generated schedules have been conducted by the Los Angeles Sheriff's Department to further validate the approach. In particular, the LASD conducted two 4-hour patrol shifts on Jan. 4 and Jan. 5, 2012 as initial trials, followed by ten 3-hour shifts on seven distinct dates in May and June and twelve more 3-hour shifts on Sep. 21 and 24, 2012. These shifts were all conducted on the Red line. Figure 6.12 shows one example of a patrol shift given to the LASD where "Post" represents a fare inspection in the given station and "Train" represents a fare inspection on the given train. The LASD followed the given shift as much as they could and for each inspection period they collected statistics including the number of patrons checked, warned, cited and arrested. They were also encouraged to provide

Post	UNION 15:06 → 15:41 (35 mins)
Train	UNION 15:41 → WILSHIRE/VERMONT 15:50 (9 mins)
Post	WILSHIRE/VERMONT 15:50 → 16:44 (54 mins)
Train	WILSHIRE/VERMONT 16:44 → 7TH/METRO CENTER 16:48 (4 mins)
Post	7TH/METRO CENTER 16:48 → 17:23 (35 mins)
Train	7TH/METRO CENTER 17:23 → UNION 17:28 (5 mins)
Post	UNION 17:28 → 17:58 (30 mins)

Figure 6.12: Example of a fare inspection patrol shift.

feedback on the schedules especially when they were unable to follow the patrol completely. An example sheet of collected statistics and feedback is given in Figure 6.13.


ACTION: POST TRAIN	DESTINATION BEGINNING / TIME	DESTINATION ENDING / TIME	MINUTES	FARE CITES	OTHER CITES	W/A	FARE CKS	ARRESTS	WARRANT CITEOUTS
Post	Union STA/06:01	Union STA/06:34	30	~~155~~6	—	2	155	0	0
Train	Union STA/06:31	Wilsh/Vermont/0640	9	0	—	0	25	0	0
Post	Wilsh Vermont/0640	Wilsh Verm /0716	36	1	—	0	190	0	0
Train	Wilsh Vermont/7:16	Vermont Sunset/0721	5	— CROWDED.			-	Train	—
~~Post~~	~~Vermont Sunset 0735~~	~~Vermont Sunset 0813~~		~~2~~		~~4~~	~~203~~	~~0~~	~~0~~
Post	Vermont Sunset 0735	Vermont Sunset 0813	38	2	—	4	203	0	0
Train	Vermont Sunset 0813	Union STA 0828	15	0	—	0	50	0	0
Post	Union STA 0828	Union STA 0900.	30	1	—	1	94	0	0
		TOTALS =		~~10~~10	—	5	717	0	0

TRIED THIS SCHEDULE WITH 4 SA'S (2-2MAN TEAMS) TO SEE IF THE NUMBERS INCREASED

Figure 6.13: Example of shift statistics and feedback provided by the LASD.

Table 6.2 summarizes the comparison between TRUSTSv1 shifts and LASD's regular shifts. It worths noting that TRUSTSv1 shifts were more effective than regular shifts—officers were able to check more patrons and catch more fare evaders following the TRUSTSv1 schedules. TRUSTSv1 shifts also detected a higher citation rate (i.e. fare evasion rate) than regular shifts. A plausible explanation of this observation is that regular shifts decided by human schedulers may be limited to certain spatio-temporal patterns and can fail to intersect high evasion rate traffic emerged consequently. Being fully machine-randomized and optimized, TRUSTS on the other

(Daily Average)	TRUSTSv1	Regular
Checks per officer	403.3	317.8
Citations per officer	6.72	3.54
Citation rate	1.67%	1.11%

Table 6.2: Comparison between TRUSTSv1 shifts and LASD's regular shifts.

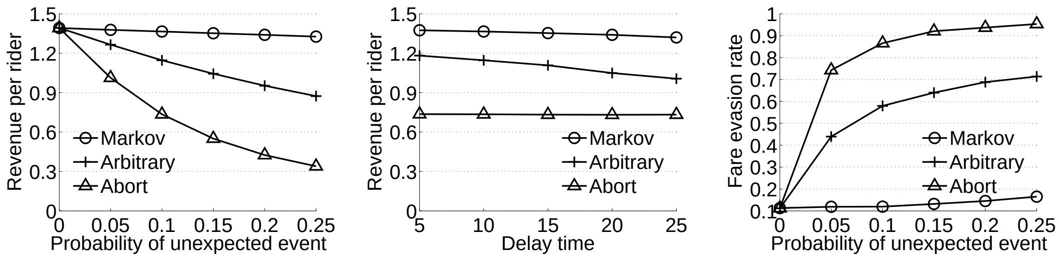
hand is able to identify location and time pairs where fare inspection can be the most effective yet avoids being predictable.

While field trials of TRUSTSv1 showed encouraging results, serious issues also emerged from the feedback given by the LASD. First, 8 out of 24 patrols were explicitly reported as executed with errors for various reason including felon arrests, train delays, backup requests, and etc. The observation that execution error affected at least one third of the TRUSTSv1 schedules motivates the need of moving the TRUSTS system towards TRUSTSv2 to provide recoverable patrols. Moreover, the feedback written on paper requires significant human effort to digitalize the statistics for further systematic analysis. More importantly, this digitalization process is subject to significant noise—counting errors, typos, bad handwriting, and/or incorrect data entry can all corrupt the valuable data collected. The mobile phone application proposed in my thesis is indeed motivated by resolving these important issues by providing user-friendly interface for TRUSTSv2 schedules and simplifying the task of data collection, transmission, and formatting.

6.3.4 Simulation Results of TRUSTSv2

I studied the performance of the Markov strategies generated by TRUSTSv2 under a variety of settings. As mentioned earlier, to better fit the reality, the experiments in this section assumed that the inspection rate was 3 passengers per minute instead of 10 (which was used in the experiments described in Section 6.3.2). Throughout the settings that I have tested, the Markov strategy was close to optimal with revenue always above 99% of the LP upper bound. Therefore in the remainder of this subsection I will report values of the Markov strategy without mentioning the LP upper bound.

In the first set of experiments, I compared, under execution uncertainty, the performance of the Markov strategy against pre-generated schedules given by TRUSTSv1, a deterministic model assuming perfect execution. However, actions to take after deviations from the original plan are not well-defined in TRUSTSv1 schedules, making a direct comparison inapplicable. Therefore, I augmented these pre-generated schedules with two naive contingency plans indicating the actions to follow after a unit deviates from the original plan. The first plan, "Abort", is to simply abandon the entire schedule and return to the base. The second plan, "Arbitrary", is to pick an action uniformly randomly from all available actions at any decision point after the deviation.



(a) (b) (c)

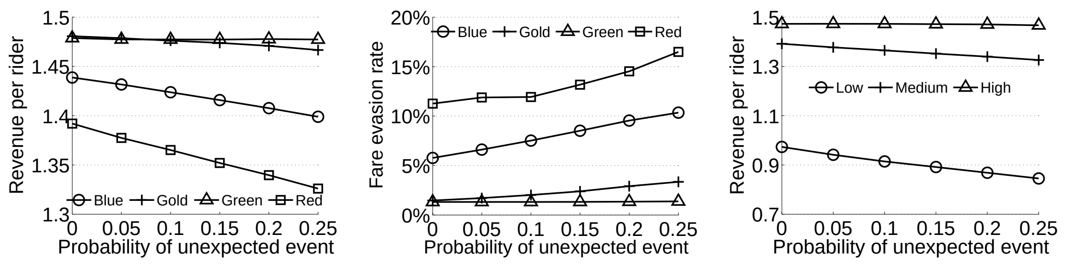
Figure 6.14: Markov strategy (TRUSTSv2) vs. pre-generated strategy (TRUSTSv1): (a) revenue per rider of varying η (b) revenue per rider of varying delay time (c) evasion rate of varying η

In this experiment, I fixed the number of units to 6 and the patrol length to 3 hours, and presented the results on the Red line (experiments on other lines showed similar results). I first fixed the delay time to 10 minutes and varied the delay probability η from 0% to 25%. As we can see in Figure 6.14(a), both "Abort" and "Arbitrary" performed poorly in the presence of execution uncertainty. With increasing values of η, the revenue of "Abort" and "Arbitrary" decayed much faster than the Markov strategy. For example, when η was increased from 0% to 25%, the revenue of “Abort” and “Arbitrary” decreased 75.4% and 37.0% respectively while that of the Markov strategy decreased only 3.6%.

In addition to revenue, Figure 6.14(c) showed the fare evasion rate of the three policies with increasing η. Following the same categorization as in Figure 6.9(a), I considered a rider to prefer fare evasion if and only if his expected penalty from fare evasion is less than \$1.3. As we can see, “Abort” and “Arbitrary” showed extremely poor performance in evasion deterrence with even a tiny probability of execution error. In particular, when η was increased from 0% to 5%, the evasion rate of the Markov strategy barely increased while that of “Abort” and “Arbitrary” increased from 11.2% both to 74.3% and 43.9% respectively.

Then I fixed η to 10% and varied the delay time from 5 to 25 minutes. Figure 6.14(b) showed that both “Abort” and “Arbitrary” performed worse than the Markov strategy. With increasing delay time, the revenue of “Abort” remained the same as the time of the delay really did not matter if the unit was to abandon the schedule after the first unexpected event. The revenue of “Arbitrary”, however, decayed in a faster rate than the Markov strategy. When the delay time was increased from 5 to 25 minutes, the revenue of “Abort” remained the same while that of “Arbitrary” and the Markov strategy decreased 14.4% and 3.6% respectively.

An important observation here is that the revenue of “Abort”, a common practice in fielded operations, decayed extremely fast with increasing η — even with a 5% probability of delay, the revenue of “Abort” was only 73.5% of that of the Markov strategy. With a conservative estimate of 6% potential fare evaders Booz Allen Hamilton [2007] and $300,000$ daily riders in the LA Metro Rail system, the 26.5% difference implies a daily revenue loss of \$$6,500$ or \$2.4 million annually.



(a) (b) (c)

Figure 6.15: Simulation results of TRUSTSv2: (a) Revenue per rider of Markov strategy (b) Evasion rate of Markov strategy (c) Revenue decay with varying coverage levels.

In the second set of experiments, I showed that the Markov strategy performed well consistently in all of the four lines with increasing delay probability η. I fixed the number of units to 6 and the patrol length to 3 hours, but varied η from 0% to 25%. Figure 6.15(a) and Figure 6.15(b) showed the revenue per rider and the evasion rate of the four lines respectively[1]. As we can see, the revenue decreased and the evasion rate increased with increasing η. However, the Markov strategy was able to effectively allocate resources to counter the effect of increasing η in terms of both revenue maximization and evasion deterrence. For example, the ratio of the revenue of $\eta = 25\%$ to that of $\eta = 0\%$ was 97.2%, 99.1%, 99.9%, 95.3% in the Blue, Gold, Green and Red line respectively. Similarly, when η was increased from 0% to 25%, the evasion rate of the Blue, Gold, Green and Red line was increased by 4.6, 1.9, 0.1, 5.2 percentage points respectively.

The next experiment showed that the revenue decay of the Markov strategy with respect to delay probability η could be affected by the amount of resources devoted to fare enforcement. In Figure 6.15(c), I presented the revenue per rider with increasing η on the Red line only, but the same trends were found on the other three lines. In this experiment, I considered 3, 6 and 9 patrol units, representing three levels of fare enforcement: low, medium, and high respectively.

[1]The revenue of the Red line was significantly lower than the other lines because fare check effectiveness f defined in Section 6.2.3.1 was set inversely proportional to the ridership volume.

Intuitively, with more resources, the defender could better afford the time spent on handling unexpected events without sacrificing the overall revenue. Indeed, as we can see, the rate of revenue decay with respect to η decreased as we increased the level of fare enforcement from low to high. For example, when η was increased from 0% to 25%, the revenue drop in the low, medium and high enforcement setting was 13.2%, 4.7%, and 0.4% respectively.

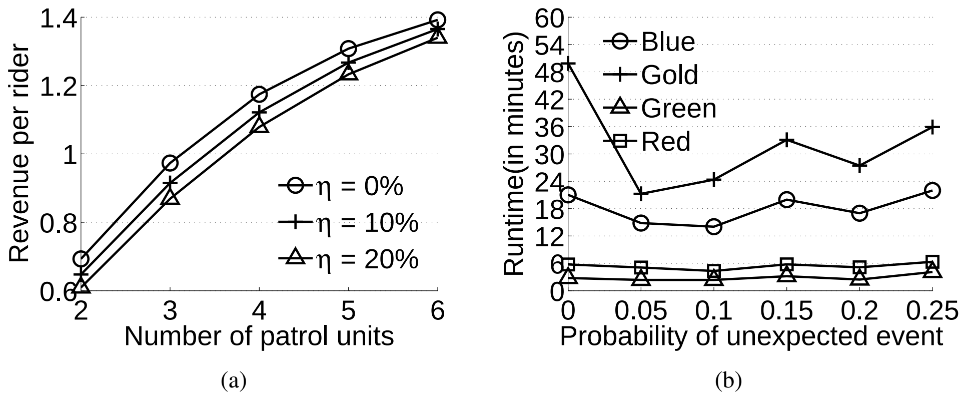


Figure 6.16: Simulation results of TRUSTSv2: (a) Revenue per rider with increasing coverage (b) Worst-case LP runtime.

Next, we demonstrate the usefulness of our Markov strategy in distributing resources under different levels of uncertainty. I showed results on the Red line with a fixed patrol length of 3 hours. Three delay probabilities η = 0%, 10%, and 20% were considered, representing increasing levels of uncertainty. Figure 6.16(a) showed the revenue per rider with increasing number of units from 2 to 6. As I increased the number of units, the revenue increased towards the maximal achievable value of \$1.5 (ticket price). For example, when $\eta = 10\%$, the revenue per rider was \$0.65, \$1.12, and \$1.37 with 2, 4, and to 6 patrol units respectively.

Finally, Figure 6.16(b) plotted the worst-case runtime (over 10 runs) of the LP with increasing η for the four metro lines. The number of units was fixed to 3 and the patrol length per unit was fixed to 3 hours. As we can see, TRUSTSv2 was able to solve all of the problems within an hour.

The runtime varied among the four Metro lines and correlated to their number of states and types. For example, when $\eta = 10\%$, the runtime for the Blue, Gold, Green, and Red line was 14.0, 24.3, 2.4, and 4.3 minutes respectively. Surprisingly, for all of the four lines, stochastic models with $\eta = 5\%$ took less time to solve than deterministic models ($\eta = 0\%$). Overall no significant correlation between the runtime and delay probability η was found.

Chapter 7: Related Work

Dealing with uncertainty and finding robust equilibrium has long been an active topic in game theory traditionally with a focus on simultaneous-move games. Numerous models and approaches were proposed such as the classic Bayesian game model Harsanyi [1967], robust game theory Aghassi and Bertsimas [2006b], and various equilibrium refinement concepts Selten [1975]; McKelvey and Palfrey [1995]; Beja [1992]. My thesis focuses on Stackelberg games, which have received a lot of recent attention due to their real world deployment—the ARMOR program Pita et al. [2008] has been deployed at the Los Angeles International Airport since 2007. Since then, many new uncertainty models, robust techniques, and human bias models for Stackelberg games were developed focusing on security applications Tambe [2011].In addition to research on uncertainty in game theory, another line of research that is related to my thesis and in particular the TRUSTS application aims at finding efficient algorithms for solving complex graph-based patrolling games, either optimally or approximately.

Therefore, in this chapter, I will describe research related to my thesis in the following three categories: (i) work on addressing uncertainty in simultaneous-move games, (ii) uncertainty modeling and robust solutions for Stackelberg games and security games, (iii) efficient solutions for solving complex graph patrolling games.

7.1 Uncertainty in Simultaneous-move Games

In game theory, uncertainty about data (such as players' payoffs) is typically considered in simultaneous-move games with a focus on finding robust Nash equilibria. Harsanyi 1967 modeled incomplete information games (i.e., games with payoff uncertainty) as Bayesian games, encoding such payoff uncertainty in players' type information. He showed any Bayesian game is equivalent to an extensive-form game with complete, but imperfect information. This extensive-form game, in turn, is known to have a strategic-form representation. This modeling technique requires the availability of the full prior distributional information for all uncertain parameters.

Robust game theory Aghassi and Bertsimas [2006b], alternatively, employed an uncertainty model analogous to robust optimization Ben-Tal et al. [2008] and provided a distribution-free equilibrium concept called robust-optimization equilibrium. They showed that computing a robust-optimization equilibrium for finite games with bounded polyhedral payoff uncertainty sets is equivalent to identifying a Nash equilibrium of finite games with complete information. My work Recon Yin et al. [2011] employs a similar framework in terms of optimizing the worst-case utility with bounded uncertainty sets, but solves for the Stackelberg equilibrium, where the work of Aghassi and Bertisimas 2006b is not applicable.

Other than payoff uncertainty, work in the game theory community also investigates execution error or bounded rationality separately, such as trembling-hand perfect equilibria Selten [1975], quantal response equilibria McKelvey and Palfrey [1995] and imperfect equilibria Beja [1992]. The goal of these works is to refine notions of equilibrium. In each of these works, players are assumed to make errors in choosing which pure strategy to play. The deviations to the intended actions of the players are correlated with the expected payoff of each of the actions. Execution error was also studied in repeated games with incomplete information such as Archibald and Shoham [2011].

While the uncertainty models and equilibrium refinement concepts can often be adapted in the Stackelberg setting as seen in Paruchuri et al. [2008]; Yin et al. [2011]; Yang et al. [2012], the algorithms developed for simultaneous-move games are generally inapplicable for Stackelberg games. Consequently, many works focusing on Stackelberg games have been proposed separately.

7.2 Uncertainty in Stackelberg Games

7.2.1 Algorithms for Bayesian Stackelberg Games

For leader-follower Stackelberg games, the Bayesian extension analogous to that of simultaneous-move games has received much recent research interest due to their uses in deployed security applications Tambe [2011]. In particular, Conitzer and Sandholm 2006 proved that finding an optimal leader's mixed strategy in two-player Bayesian Stackelberg games is NP-hard, and provided a solution method by solving multiple linear programs (possibly exponentially many). Parachuri et al. 2008 provided Dobss— a single mixed integer linear program formulation that solves the problem. Jain et al. 2011b employed a branch-and-bound search algorithm in which they computed heuristic upper and lower bounds by solving smaller restricted problems. Despite the algorithmic advancement, none of these techniques can handle games with more than 50 types, even when the number of actions per player is as few as 5. Beyond discrete uncertainty, Kiekintveld et al. 2011 modeled continuously distributed uncertainty over preferences of the follower as Bayesian Stackelberg games with infinite types, and proposed an algorithm to generate approximate solutions for such games.

My work advances this line of research in two aspects: (i) I provided a novel Bayesian Stackelberg game solver Hunter which runs orders of magnitudes faster than previous methods, (ii) I extended the Bayesian Stackelberg games to model the leader's execution and the follower's observation uncertainty (potentially continuous) in a unified framework.

7.2.2 Robust Solutions

As an alternative to the Bayesian model of uncertainty, there have been works on security games that compute robust solutions without explicitly modeling the uncertainty. Cobra Pita et al. [2010] assumes that the follower has bounded rationality and may not strictly maximize expected value. As a result, the follower may select an ϵ-optimal response strategy, i.e., the follower may choose any of the responses within ϵ of his optimal strategy. Cobra attempts to maximize the leader's expect value for the worst-case scenario that fall within this ϵ-bound of the optimal response.

In contrast, Match Pita et al. [2012] employs an idea of graduated robust optimization, which constrains the impact of the follower's deviations depending on the magnitude of the deviation. In particular, Match bounds the leader's loss for a potential deviation of the follower by an adjustable fraction of the follower's loss for the deviation from the expected-value-maximizing strategy.

In addition to robust optimization formulations, An et al. 2011 provided refinement methods to strong Stackelberg equilibrium in security games to achieve "free" additional robustness against potential off-equilibrium actions played by an adversary due to his capability limitations.

My work Recon complements the works above by explicitly considering two major causes of real world uncertainty: the leader's execution error and the follower's observation noise, and therefore provide solutions that are robust to such uncertainty. Recon is able to utilize partial knowledge about the uncertainty such as the noise levels at different targets when such information is available. In contrast, Cobra and Match are incapable of taking advantage of such uncertainty knowledge due to their limited parameter space.

7.2.3 Against Suboptimal Opponents

Another line of research studied systematic biases and bounded rationality of human opponents in the context of security games. Pita et. al. 2010 suggested an anchoring-bias of humans decision makers and incorporate this human modeling component in their algorithmic contribution Cobra. Yang et. al. have designed algorithms Yang et al. [2011, 2012] to compute solutions for Stackelberg games based on the prospect theory model Kahneman and Tversky [1979] and the quantal response model McKelvey and Palfrey [1995]. Shieh et. al. showed that quantal response model could also provide robustness against execution and observation errors as an effect of smoothing out the follower's response Shieh et al. [2012].

These works have focused on creating accurate human decision making models using controlled human subject experiments. When sufficient domain data is available, integrating human decision models with Bayesian Stackelberg game model can be a valuable future research topic. However finding perfect models of human decision making is difficult and requires a large amount of data which can be limited in certain security applications. When data is limited, it can be beneficial for the security agency to use robust optimization framwork such as Recon, Cobra, or Match.

7.2.4 Observability and Commitment

In terms of the follower's observation uncertainty, there has been significant interest in understanding the interaction of observability and commitment in general Stackelberg games. Bagwell 1995 questioned the value of commitment to pure strategies given noisy observations by followers; but the ensuing and on-going debate illustrated that the leader retains her advantage in case of commitment to mixed strategies Huck and Mller [2000]; van Damme and Hurkens [1997]. The value of commitment for the leader when observations are costly was studied in Morgan and Vardy [2007]. Secrecy and deception in Stackelberg games were also considered in Zhuang and Bier [2011]. In contrast, my work focused on real-world security games, providing theoretical properties Yin et al. [2010] that are non-existent in general Stackelberg games studied previously.

In the context of security games, limited follower's observability has been investigated both theoretically assuming that the follower updates his belief according to Bayes' rule An et al. [2012] and empirically through human subject experiments Pita et al. [2010]. Both investigations stick to a modified Stackelberg paradigm where the follower is assumed to infer the leader's strategy through a limited number of observations. Such approaches are however sensitive to the follower's observability model such as the number of observations allowed, which is difficult to estimate in certain security applications. My work, alternatively, established a theoretical partial equivalence between the leader's strategies in the Stackelberg (perfect observability) and simultaneous-move (no observability) models, suggesting that playing a Stackelberg equilibrium strategy is optimal for the leader regardless of the follower's observability. As a followup on my work, Korzhyk et. al. 2011a studied the problem when the follower observes the leader's strategy perfectly with a known probability and does not observe at all otherwise.

7.2.5 Markov Decision Process and Stochastic Games

The MDP model used in TRUSTSv2 Jiang et al. [2013] for modeling execution uncertainty resembles the transition independent DEC-MDP (TI-DEC-MDP) Becker et al. [2003]. TRUSTSv2 with the coupled execution of multiple teams is analogous to the TI-DEC-MDP model with full communication, where the optimal joint policy is sought. Decoupled execution on the other hand corresponds to the TI-DEC-MDP model with no communication. However, two major distinctions exist, presenting unique computational challenges. First, TRUSTSv2 considers the strategic interaction against adversaries and focus on equilibrium computation. Second, utility functions in TRUSTSv2 are non-Markovian which depend on the entire trajectories as opposed to only state and action pairs in typical DEC-MDP models.

The game model of TRUSTS in my thesis can be considered as a special case of extensive-form Stackelberg games with chance nodes, or as a special case of stochastic Stackelberg games Basar and Olsder [1995]. The state in this special stochastic Stackelberg game is essentially the patroller's physical state (location and time) and the transitions between states are purely dependent on the patroller's actions. The follower (rider) in this special game can only choose one action (i.e., buy or not buy the ticket) in the initial state and stick to that action in all future states. The general cases of both games were shown to be NP-hard Letchford and Conitzer [2010]; Letchford et al. [2012]. Vorobeychik and Singh 2012 provided mixed integer linear programs for finding optimal and approximate Markov stationary strategy in general-sum stochastic Stackelberg games. However, their approach does not handle multiple adversary types and their MILP formulation lacks the scalability to a large number of states—inapplicable to the Los Angeles Metro problems studied in my thesis.

7.3 Solving Complex Graph Patrolling Games

There has been research on a wide range of problems related to game-theoretic patrolling on graphs that are related to TRUSTS presented in this thesis. One line of work considers games in which one player, the patroller, patrols the graph to detect and catch the other player, the evader, who tries to minimize the detection probability. This includes work on *hider-seeker games* Halvorson et al. [2009] for the case of mobile evaders and *search games* Gal [1979] for the case of immobile evaders.

Another line of research considers games in which the patroller deploys resources (static or mobile) on the graph to prevent the other player, the attacker, from reaching certain target vertices. There are a few variations depending on the set of possible sources and targets of the attacker. *Infiltration games* Alpern [1992] considered one source and target. *Asset protection* problems Dickerson et al. [2010] and *Network interdiction* Washburn and Wood [1995] consider multiple sources and multiple equally weighted targets.

In the context of Stackelberg games for security, there have been numerous works related to solving large-scale graph-related problems such as protecting commercial flights Tsai et al. [2009]; Kiekintveld et al. [2009]; Jain et al. [2010], protecting urban road network Tsai et al. [2010]; Jain et al. [2011a], port security Shieh et al. [2012], hostile area transit Vanek et al. [2011], malicious packet detection Vanek et al. [2012b], preventing illegal extraction of forest resources Johnson et al. [2012], and etc. Large scale games on graphs often involve combinatorial size of pure strategies which grows exponentially with increasing problem sizes. Therefore general purpose Stackelberg game solvers such as Dobss and Hunter can be extremely inefficient when applied directly. Efficient solution approaches for large scale problems can be generally divided into three categories: (i) exact solution method using *oracle-based algorithms* Jain et al. [2010, 2011a]; Vanek et al. [2011]; Tsai et al. [2012], (ii) approximate solution method utilizing submodular objective functions Krause et al. [2011]; Vanek et al. [2012b], (iii) approximate (relaxed) solution method via compact strategy representation Kiekintveld et al. [2009]; Tsai et al. [2010].

Oracle-based algorithms start with a small subset of pure strategies of the full game and search for an equilibrium iteratively in a succession of increasingly larger subgames of the full game. In each iteration the best response for the current subgame is provided by an oracle and added to the current pure strategy sets of the respective player. The performance of the oracle plays an important role in the overall performance of the algorithm. Jain et. al. 2010 presented Aspen for scheduling air marshals to protect commercial flights (FAMS), which combines a branch-and-price approach and a single best-response oracle for generating the defender's pure strategies. Double oracle algorithms, one for each player, were used in zero-sum games where both players have large pure strategy spaces. Jain et. al. 2011a provided a double oracle algorithm for scheduling checkpoints in urban road network where the defender chooses a combination of road segments to set up checkpoints and the attacker chooses a path in the network to reach a desired target.

In certain security domains that the leader's utility function has the submodularity property, i.e., a natural effect of diminishing returns, there exist approximation algorithms with provable quality guarantees. Specifically, the leader is considered to have a submodular utility function if the marginal utility of deploying additional resources helps more if few resources have been deployed and less if many resources have been deployed. Submodularity has been exploited in optimizing sensor allocations in adversarial environment Krause et al. [2011] and in randomizing deep packet inspections for malicious packet detection within computer networks Vanek et al. [2012b].

The final line of research focused on finding approximate solutions by utilizing compact strategy representation. Eraser-C Kiekintveld et al. [2009] is an approximate algorithm for the FAMS problem, representing the defender's mixed strategy as a marginal coverage vector. This representation relaxes the original strategy space and therefore may fail to generate a feasible solution in cases where arbitrary schedules with more than two flights (i.e., multi-city tours) are allowed. Eraser-C avoids enumerating joint schedules to gain runtime efficiency, but loses the ability to correctly model arbitrary schedules. Similar to Eraser-C, Ranger Tsai et al. [2010] solves the urban network security problem using a marginal coverage representation of the defender's allocation strategy of road checkpoints and provides a couple of sampling approaches to create feasible pure allocations from the marginal strategy generated. Although the sampling approaches are always guaranteed to match the marginal coverage vector, the defender's utility function being optimized in Ranger is overestimated and therefore Ranger may not find the optimal solution.

TRUSTS presented in this thesis, however, introduces unique computational challenges. First, unlike in existing work on graph patrolling games and previous security applications for counterterrorism, the followers to influence in TRUSTS are potentially very many: large numbers of train riders might plausibly consider fare evasion. Booz Allen Hamilton 2007 estimates that 6% of riders are ticketless in the metro system overall; anecdotal reports suggest that on some lines this percentage could be far greater, even a majority. Second, the patrols in TRUSTS correspond to all the feasible trips within the transit network subject to restrictions and preferences that were nonexistent in previous applications. Similar to Eraser-C Kiekintveld et al. [2009] and Ranger Tsai et al. [2010], the patrol strategies in TRUSTS were compactly represented as a marginal coverage vector. But unlike the FAMS problem where a patrol consists of very limited number of flights (often a pair of flights) and unlike the urban network security problem where checkpoints can be placed arbitrarily on any edges in the graphs without any constraints, TRUSTS allows much more complex patrol constraints using a novel compact representation based on *history-duplicate transition graphs*. Moreover, in contrast to ERASER-C which may fail to provide a feasible solution, the approximate solutions given by TRUSTS are always feasible with near-optimal performance on real datasets.

Chapter 8: Conclusions

Game-theoretic approaches have shown their usefulness in deployed security applications such as ARMOR for the Los Angeles International Airport Pita et al. [2008], IRIS for the Federal Air Marshal Service Tsai et al. [2009], GUARDS for the Transportation Security Administration Pita et al. [2011], PROTECT for the Boston Coast Guard Shieh et al. [2012], and TRUSTS for the Los Angeles Metro Rail System Yin et al. [2012a]. At the core of the these applications is the Stackelberg game model. Despite its recent success in real world deployments, the Stackelberg game paradigm is often questioned due to unrealistic assumptions such as (i) the security agency has a complete knowledge about the adversary, (ii) the security agency can perfectly execute the planned security activities, and (iii) the adversary can observe the exact mixed strategy of the security agency, i.e., a probability distribution over actions.

Given the huge growth of recent research interest at the intersection between computer science and game theory, there has been heated discussions about “Does game theory actually work?”. The answer is not as straightforward as one might think, and vastly depends on how game theory here is interpreted. Wooldridge 2012 gave two interpretations: a descriptive interpretation which views game theory as predicting how (human) players will behave in strategic settings, and a normative interpretation which views game theory as a tool to recommend action for players. My thesis focuses on applying game theory to real world problems (in particular security randomization), where the grand task is to make game theory work better under both its descriptive and normative interpretations.

To this end, my thesis on the one hand augments the existing game-theoretic framework to model and address real world uncertainty such as those in preference, execution, and observation, providing better descriptive models and the corresponding solution methods for these real world problems. On the other hand, my thesis also addresses various real world challenges that arise from public transit domains such as scheduling constraints, human preferences, patrol interruptions, and so on, providing practical and usable recommendations to human users. In particular, my thesis has the following four key contributions.

8.1 Contributions

- HUNTER is a new algorithm for solving discrete finite Bayesian Stackelberg games, combining five key ideas:

  - efficient pruning via a best-first search in the follower's strategy space;
  - a novel linear program for computing tight upper bounds for this search;
  - using Bender's decomposition for solving the upper bound linear program efficiently;
  - efficient inheritance of Bender's cuts from parent to child;
  - an efficient heuristic branching rule.

  My experimental results suggest that HUNTER could provide orders of magnitude speedups over the best existing methods for Bayesian Stackelberg games Conitzer and Sandholm

[2006]; Paruchuri et al. [2008]; Jain et al. [2011b]. Moreover, as verified by my experiments, HUNTER's efficiency can be exploited in the *sample average approximation* approach to handling execution and observation uncertainty in both discrete and continuous forms in a unified framework.

- RECON is a robust optimization framework to address execution and observation uncertainty of unknown distribution, with a focus on security games motivated by the ARMOR application. RECON is suitable for security applications where full distributional knowledge about the uncertainty is difficult or impossible to acquire. In the absence of the precise uncertainty distribution, RECON models the uncertainty boundary as a hyper-rectangle, and correspondingly computes the optimal risk-averse strategy for the leader. I provide experimental analysis comparing the performance of various security game strategies including those generated by RECON and HUNTER in simulated uncertainty settings, showing the value of RECON and HUNTER under different assumptions.

- Stackelberg vs. Nash: This work answers a fundamental question in game-theoretic modeling of security applications: what should the security agency do if it is uncertainty whether or not the adversary will conduct surveillance. I provide theoretical and experimental analysis of this problem, focusing on security games motivated by the ARMOR and IRIS applications. In particular, I show that in security games that satisfy the *SSAS* property (such as ARMOR games), any Stackelberg game equilibrium strategy for the defender is also a Nash equilibrium strategy. In this case, the defender is therefore best-responding with a Stackelberg equilibrium strategy regardless of the follower's ability to observe. On the other hand, counter-examples to this (partial) equivalence between the Stackelberg and

Nash equilibrium strategies exist when the *SSAS* property does not hold. However, my experiments show that in this case, the fraction of games where the Stackelberg equilibrium strategy is not in any Nash equilibrium is vanishingly small with increasing problem sizes, especially for the IRIS games which have small schedule size and a large number of schedules.

- TRUSTS is a new application for scheduling inspection patrols in public transit systems for fare evasion deterrence, which presents new challenges in game-theoretic modeling and execution uncertainty handling. In particular, security activities in TRUSTS are carried out as sequences of actions in different place and time subject to strict restrictions imposed by the underlining train system and preferences expressed by human patrollers. Execution uncertainty in such spatiotemporal domains needs an entirely different treatment than earlier applications such as ARMOR and IRIS since an execution error can affect the security officers' ability to carry out their planned schedules in later time steps. The novel contributions of TRUSTS are the following:
  - a general Bayesian Stackelberg game model for spatiotemporal patrolling with execution uncertainty where the execution uncertainty is represented as Markov Decision Processes,
  - a compact strategy representation when the utility functions have a certain separable structure, which reduces the problem to a polynomial-sized linear optimization problem,
  - a novel history-duplicate approach to encode constraints on feasible patrols within the compact representation,

- a smart phone app implementation of the generated patrol schedules with contingency plans,
- simulations and real world experiments on the Los Angeles Metro Rail system in collaboration with the Los Angeles Sheriff Department.

My simulation results show that TRUSTS can provide near-optimal solutions for large scale problems within reasonable runtime requirement. Initial real world trials show encouraging results, indicating that TRUSTS schedules can be more effective than human-created schedules in catching fare evaders.

To summarize, my thesis contributes multiple uncertainty models for Stackelberg games focusing on security applications where the leader's execution and follower's observation are imperfect. These contributions allow the security agency to utilize different amounts of information available about the uncertainty and generate reliable or robust strategies or even strategies with contingency plans in the case where execution errors at earlier steps may void plans after.

8.2 Future Work

In the future one can imagine game-theoretic approaches to be applied in a large spectrum of applications far beyond counterterrorism. The growing list of such applications ranges from ensuring safety in public facilities such as transportation hubs, parks, and sports stadiums to protecting natural resources such as forests, animals, fishes, and etc. More and more new applications are emerging at a rapid rate, bringing significant challenges in scalability and modeling that require future research endeavors.

While this thesis presented algorithmic advancement in Bayesian Stackelberg games that allows problems with significantly more types to be solved, the scalability of the algorithm is still limited, inadequate for large scale problems that may involve tens of thousands of opponents such as riders in public transit system[1], cars in road network, and criminals in large metropolitan areas. In addition, a straightforward Stackelberg game model may no longer be suitable for new applications where some adversaries may be opportunistic without deliberate planning and intelligent inferences. For example, patrols in the Los Angeles Metro Rail system serve multiple purposes including ticket enforcement, ensuring public safety by suppressing crimes, and counterterrorism. Compared to terrorists who are careful planners with surveillance and fare evaders who are informed decision makers, pickpockets on trains who snatch smart phones are more opportunistic. Finally, a pure mathematical model without correct numbers cannot work in practical problems. It thus requires significant research and engineering effort in creating accurate and predictive models using quantitative methods.

In the short run, my goal is to develop algorithms for Stackelberg games with a large number types to meet the needs of future applications. To this end, I plan to further improve the scalability of my Bayesian Stackelberg game solver HUNTER by exploring different relaxation techniques and search heuristics. Moreover, I plan to design new approximation schemes to provide high quality solutions to large scale problems that cannot be solved to optimality. Finally, another interesting direction to pursue is to integrate human decision models into the Bayesian framework, allowing the use of multiple types of human adversary each characterized by a different human decision model.

[1]TRUSTS has to stick to a zero-sum model to avoid the computational complexity for solving a general-sum model, which prohibits modeling of human biases and risk adjustments.

In the long run, it will be important to devise mathematical models for applications with both deliberate and opportunistic adversaries. One possible way is to create a mixture of a Stackelberg game model against deliberate saboteurs and a partial differential equation model for modeling the dynamics of opportunistic crimes Short et al. [2010]. It will also be important to employ quantitative methods to create game-theoretic models using real world data systematically collected from security operations. As shown in Section 6.2.3.2, mobile applications can improve law enforcement agencies' efficiency and effectiveness in carrying out their daily duties as well as collect and format patrol data automatically. Such data along with certain statistical inference methods will help future researchers to create more accurate models, and in turn improve the effectiveness of game-theoretic methods. For example it can help creating more accurate distributions over different types of adversary, yielding better solution of Hunter. It can also help creating more accurate ridership distributions and MDP transition models for the TRUSTS system, and in turn improves the effectiveness of the fare inspection operations.

Bibliography

Michele Aghassi and Dimitris Bertsimas. Robust game theory. *Math. Program.*, 107:231–273, June 2006a.

Michele Aghassi and Dimitris Bertsimas. Robust game theory. *Math. Program.*, 107:231–273, June 2006b.

Noa Agmon, Vladimir Sadov, Gal A. Kaminka, and Sarit Kraus. The Impact of Adversarial Knowledge on Adversarial Planning in Perimeter Patrol. In *AAMAS*, volume 1, 2008.

Shabbir Ahmed, Alexander Shapiro, and Er Shapiro. The sample average approximation method for stochastic programs with integer recourse. *SIAM Journal of Optimization*, 12:479–502, 2002.

S. Alpern. Infiltration games on arbitrary graphs. *Journal of Mathematical Analysis and Applications*, 163(1):286 – 288, 1992.

B. An, D. Kempe, C. Kiekintveld, E. Shieh, S. Singh, M. Tambe, and Y. Vorobeychik. Security games with limited surveillance: An initial report. In *AAAI Spring Symposium on Game Theory for Security, Sustainability and Health*, 2012.

Bo An, Milind Tambe, Fernando Ordonez, Eric Shieh, and Christopher Kiekintveld. Refinement of strong Stackelberg equilibria in security games. In *AAAI*, 2011.

Christopher Archibald and Yoav Shoham. Hustling in repeated zero-sum games with imperfect execution. In *IJCAI*, 2011.

Kyle Bagwell. Commitment and observability in games. *Games and Economic Behavior*, 8: 271–280, 1995.

Egon Balas. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics*, 89(1-3):3 – 44, 1998.

Jonathan F. Bard. *Practical Bilevel Optimization: Algorithms and Applications (Nonconvex Optimization and Its Applications)*. Springer-Verlag New York, Inc., 2006.

Tamer Basar and Geert Jan Olsder. *Dynamic Noncooperative Game Theory*. Academic Press, San Diego, CA, 2nd edition, 1995.

Gary Becker and WilIiam Landes. *Essays in the Economics of Crime and Punishment*. Columbia University Press, 1974.

R. Becker, S. Zilberstein, V. Lesser, and C.V. Goldman. Transition-independent decentralized markov decision processes. In *AAMAS*, pages 41–48. ACM, 2003.

Avraham Beja. Imperfect equilibrium. *Games and Economic Behavior*, 4(1):18 – 36, 1992.

Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*. Princeton University Press, 2008.

John R. Birge and Franois V. Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34(3):384 – 392, 1988.

Booz Allen Hamilton. Faregating analysis. Report commissioned by the LA Metro, `http://boardarchives.metro.net/Items/2007/11_November/20071115EMACItem27.pdf`, 2007.

M. Breton, A. Alj, and A. Haurie. Sequential stackelberg equilibria in two-person games. *Optimization Theory and Applications*, 59(1):71–94, 1988.

V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *EC: Proceedings of the ACM Conference on Electronic Commerce*, 2006.

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 *AAMAS*, 2010.

Drew Fudenberg and Jean Tirole. *Game Theory*. MIT Press, October 1991.

Shmuel Gal. Search games with mobile and immobile hider. *SIAM Journal on Control and Optimization*, 17(1):99–122, 1979.

Nicola Gatti. Game theoretical insights in strategic patrolling: Model and algorithm in normal-form. In *ECAI-08*, pages 403–407, 2008.

Andrew Gilpin and Tuomas Sandholm. Information-theoretic approaches to branching in search. *Discrete Optimization*, 8(2):147 – 159, 2011. ISSN 1572-5286.

E. Halvorson, V. Conitzer, and R. Parr. Multi-step multi-sensor hider-seeker games. In *IJCAI*, 2009.

J.C. Harsanyi. Games with incomplete information played by "Bayesian" players, i-iii. part i. the basic model. *Management science*, 14(3):159–182, 1967.

Horizon Research Corporation. Metropolitan transit authority fare evasion study. `http://libraryarchives.metro.net/DPGTL/studies/2002_horizon_fare_evasion_study.pdf`, 2002.

Steffen Huck and Wieland Mller. Perfect versus imperfect observability–an experimental test of Bagwell's result. *Games and Economic Behavior*, 31(2):174 – 190, 2000.

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

Manish Jain, Dmytro Korzhyk, Ondrej Vanek, Vincent Conitzer, Michal Pechoucek, and Milind Tambe. A double oracle algorithm for zero-sum security games on graphs. In *AAMAS*, 2011a.

Manish Jain, Milind Tambe, and Christopher Kiekintveld. Quality-bounded solutions for finite bayesian stackelberg games: Scaling up. In *AAMAS*, 2011b.

Albert Xin Jiang, Zhengyu Yin, Chao Zhang, Sarit Kraus, and Milind Tambe. Game-theoretic randomization for security patrolling with dynamic execution uncertainty. In *AAMAS*, 2013.

Matthew P. Johnson, Fei Fang, and Milind Tambe. Patrol strategies to maximize pristine forest area. In *Conference on Artificial Intelligence (AAAI)*, 2012.

D. Kahneman and A. Tversky. Prospect theory: An analysis of decision under risk. *Econometrica: Journal of the Econometric Society*, 47(2):263–291, 1979.

Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Milind Tambe, and Fernando Ordóñez. Computing optimal randomized resource allocations for massive security games. In *AAMAS*, 2009.

Christopher Kiekintveld, Janusz Marecki, and Milind Tambe. Approximation methods for infinite bayesian stackelberg games: Modeling distributional payoff uncertainty. In *AAMAS*, 2011.

D. Koller, N. Megiddo, and B. von Stengel. Fast algorithms for finding randomized strategies in game trees. In *STOC: Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 750–759, 1994.

Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Solving stackelberg games with uncertain observability. In *AAMAS*, pages 1013–1020, 2011a.

Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Security games with multiple attacker resources. In *IJCAI*, pages 273–279, 2011b.

A. Krause, A. Roper, and D. Golovin. Randomized sensing in adversarial environments. In *IJCAI*, 2011.

G. Leitmann. On generalized Stackelberg strategies. *Optimization Teory and Applications*, 26 (4):637–643, 1978.

Joshua Letchford and Vincent Conitzer. Computing optimal strategies to commit to in extensive-form games. In *EC*, 2010.

Joshua Letchford, Liam MacDermed, Vincent Conitzer, Ronald Parr, and Charles L. Isbell. Computing optimal strategies to commit to in stochastic games. In *AAAI*, 2012.

Wai-Kei Mak, David P. Morton, and R. Kevin Wood. Monte carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, 24(1-2):47 – 56, 1999. ISSN 0167-6377.

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

John Morgan and Felix Vardy. The value of commitment in contests and tournaments when observation is costly. *Games and Economic Behavior*, 60(2):326–338, 2007.

H. Moulin and J. P. Vial. Strategically zero-sum games: The class of games whose completely mixed equilibria cannot be improved upon. *International Journal of Game Theory*, 7(3-4): 201–221, 1978.

P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S Kraus. Playing games with security: An efficient exact algorithm for Bayesian Stackelberg games. In *AAMAS*, 2008.

J. Pita, Manish Jain, Craig. Western, Christopher Portway, Milind Tambe, Fernando Ordonez, Sarit Kraus, and Praveen Paruchuri. Deployed ARMOR protection: The application of a game theroetic model for security at the los angeles international airport. In *AAMAS*, 2008.

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.

James Pita, Milind Tambe, Chris Kiekintveld, Shane Cullen, and Erin Steigerwald. GUARDS - game theoretic security allocation on a national scale. In *AAMAS*, 2011.

James Pita, Richard John, Rajiv Maheswaran, Milind Tambe, and Sarit Kraus. A robust approach to addressing human adversaries in security games. In *ECAI*, 2012.

J. P. Ponssard and S. Sorin. The lp formulation of finite zero-sum games with incomplete information. *International Journal of Game Theory*, 9:99–105, 1980. ISSN 0020-7276. URL `http://dx.doi.org/10.1007/BF01769767`. 10.1007/BF01769767.

R. Selten. Reexamination of the perfectness concept for equilibrium points in extensive games. *International Journal of Game Theory*, 4:25–55, 1975.

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 *AAMAS*, 2012.

M.B. Short, A.L. Bertozzi, and P.J. Brantingham. Nonlinear patterns in urban crime: Hotspots, bifurcations, and suppression. *SIAM Journal on Applied Dynamical Systems*, 9(2):462–483, 2010.

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

Jason Tsai, Shyamsunder Rathi, Christopher Kiekintveld, Fernando Ordóñez, and Milind Tambe. IRIS - a tool for strategic security allocation in transportation networks. In *AAMAS - Industry Track*, 2009.

Jason Tsai, Zhengyu Yin, Jun-young Kwak, David Kempe, Christopher Kiekintveld, and Milind Tambe. Urban security: Game-theoretic resource allocation in networked physical domains. In *AAAI*, 2010.

Jason Tsai, Thanh H. Nguyen, and Milind Tambe. Security games for controlling contagion. In *AAAI*, 2012.

Eric van Damme and Sjaak Hurkens. Games with imperfectly observable commitment. *Games and Economic Behavior*, 21(1-2):282 – 308, 1997.

Ondrej Vanek, Michal Jakob, Viliam Lisy, Branislav Bosansky, and Michal Pechoucek. Iterative game-theoretic route selection for hostile area transit and patrolling. In *AAMAS*, 2011.

Ondrej Vanek, Zhengyu Yin, Manish Jain, Branislav Bosansky, Milind Tambe, and Michal Pechoucek. Game-theoretic resource allocation for malicious packet detection in computer networks. In *AAMAS*, 2012a.

Ondrej Vanek, Zhengyu Yin, Manish Jain, Branislav Bosansky, Milind Tambe, and Michal Pechoucek. Game-theoretic resource allocation for malicious packet detection in computer networks. In *AAMAS*, 2012b.

Heinrich von Stackelberg. *Marktform und Gleichgewicht*. Springer, 1934.

B. von Stengel and S. Zamir. Leadership with commitment to mixed strategies. In *CDAM Research Report LSE-CDAM-2004-01, London School of Economics*, 2004.

Yevgeniy Vorobeychik and Satinder Singh. Computing stackelberg equilibria in discounted stochastic games. In *AAAI*, 2012.

W. A. Wagenaar. Generation of random sequences by human subjects: A critical survey of literature. *Psychological Bulletin*, 77(1):65–72, 1972.

Alan Washburn and Kevin Wood. Two-person zero-sum games for network interdiction. *Operations Research*, 43(2):243–251, 1995.

M. Wooldridge. Does game theory work? *Intelligent Systems, IEEE*, 27(6):76–80, Nov.-Dec. 2012.

Rong Yang, Christopher Kiekintveld, Fernando Ordóñez, Milind Tambe, and Richard John. Improved computational models of human behavior in security games. In *International Conference on Autonomous Agents and Multiagent Systems (Ext. Abstract)*, 2011.

Rong Yang, Fernando Ordonez, and Milind Tambe. Computing optimal strategy against quantal response in security games. In *AAMAS*, 2012.

Zhengyu Yin and Milind Tambe. A unified method for handling discrete and continuous uncertainty in Bayesian Stackelberg games. In *AAMAS*, 2012.

Zhengyu Yin, Dmytro Korzhyk, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe. Stackelberg vs. nash in security games: Interchangeability, equivalence, and uniqueness. In *AAMAS*, 2010.

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 Xin Jiang, Matthew P. Johnson, Milind Tambe, Christopher Kiekintveld, Kevin Leyton-Brown, Tuomas Sandholm, and John Sullivan. TRUSTS: Scheduling randomized patrols for fare inspection in transit systems. In *IAAI*, 2012a.

Zhengyu Yin, Albert Xin Jiang, Matthew P. Johnson, Milind Tambe, Christopher Kiekintveld, Kevin Leyton-Brown, Tuomas Sandholm, and John Sullivan. TRUSTS: Scheduling randomized patrols for fare inspection in transit systems using game theory. *AI Magazine*, 33(4), 2012b.

Jun Zhuang and Vicki Bier. Secrecy and deception at equilibrium, with applications to anti-terrorism resource allocation. *Defence and Peace Economics*, 22:43–61, 2011.

Appendix A: Bender's Decomposition

Benders' decomposition, named after Jacques F. Benders, is a technique in mathematical programming that allows the solution of very large linear programming problems that have the following special block structure (this structure often occurs in applications such as stochastic programming):

$$\begin{array}{rllllr}
\max_{\mathbf{x},\mathbf{y_1},\dots,\mathbf{y_k}} & \mathbf{c}^{\mathrm{T}}\mathbf{x} & +\mathbf{f_1}^{\mathrm{T}}\mathbf{y_1} & +\dots & +\mathbf{f_k}^{\mathrm{T}}\mathbf{y_k} & \\
s.t. & A\mathbf{x} & & & & \leq \mathbf{b} \\
 & B_1\mathbf{x} & +D_1\mathbf{y_1} & & & \leq \mathbf{d_1} \\
 & B_2\mathbf{x} & & D_2\mathbf{y_2} & & \leq \mathbf{d_2} \\
 & \vdots & & \ddots & & \vdots \\
 & B_k\mathbf{x} & & & D_k\mathbf{y_k} & \leq \mathbf{d_k} \\
 & \mathbf{x}, & \mathbf{y_1}, & \dots, & \mathbf{y_k} & \geq \mathbf{0}
\end{array} \tag{A.1}$$

where $\mathbf{x}$, $\mathbf{y_1},\dots,\mathbf{y_k}$ are all vectors of continuous variables having arbitrary dimensions, A, $B_1,\dots,B_k$ are matrices, and $\mathbf{b}$, $\mathbf{d_1},\dots,\mathbf{d_k}$ are vectors of appropriate dimensions. Due to the special structure, the problem becomes significantly easier to solve if $\mathbf{x}$ is fixed—we can solve for each $\mathbf{y_i}$ separately. Bender's decomposition partitions problem (A.1) into a master problem that contains only the $\mathbf{x}$-variables, and k subproblems where the i-th subproblem contains variables $\mathbf{y_i}$. In particular, problem (A.1) can be partitioned into the master problem:

$$\begin{array}{rl}
\max_{\mathbf{x}} & \mathbf{c}^{\mathrm{T}}\mathbf{x} + \sum_{i=1}^{k} \phi_i(\mathbf{x}) \\
s.t. & A\mathbf{x} \leq \mathbf{b} \\
 & \mathbf{x} \geq \mathbf{0}
\end{array} \tag{A.2}$$

and k subproblems where for every $i = 1,\dots,k$, the i-th subproblem is:

$$\phi_i(\mathbf{x}) = \begin{array}{rl}
\max_{\mathbf{y_i}} & \mathbf{f_i}^{\mathrm{T}}\mathbf{y_i} \\
s.t. & D_i\mathbf{y_i} \leq \mathbf{d_i} - B_i\mathbf{x} \\
 & \mathbf{y_i} \geq \mathbf{0}
\end{array} \tag{A.3}$$

Formulation (A.3) is a linear program for any given $\mathbf{x}$. Note that if (A.3) is unbounded for some i and some $\mathbf{x}$ in the feasible region of problem (A.2), then (A.2) is also unbounded, which in turn implies the original problem (A.1) is unbounded. Assuming boundedness of (A.3), we can also calculate the value of $\phi_i(\mathbf{x})$ by solving its dual. Let $\boldsymbol{\pi}_i$ be the dual variables for constraints $D_i\mathbf{y_i} \leq \mathbf{d_i} - B_i\mathbf{x}$. Then the dual of (A.3) is:

$$\phi_i(\mathbf{x}) = \begin{array}{ll} \min_{\boldsymbol{\pi}_i} & (\mathbf{d_i} - B_i\mathbf{x})^{\mathrm{T}}\boldsymbol{\pi}_i \\ s.t. & D_i^{\mathrm{T}}\boldsymbol{\pi}_i \geq \mathbf{f_i} \\ & \boldsymbol{\pi}_i \geq \mathbf{0} \end{array} \tag{A.4}$$

The key observation is that the feasible region of the dual formulation (A.4) does not depend on the values of variables $\mathbf{x}$, which only affects the objective function. If the dual feasible region of (A.4) is empty, then either the primal problem (A.3) is unbounded for some $\mathbf{x}$ and hence the original problem (A.1) is unbounded, or the primal feasible region of (A.3) is also empty for all $\mathbf{x}$ and hence the original problem (A.1) is infeasible.

Now let us consider the non-trivial case where the feasible region of (A.4) is not empty for any $i = 1, \ldots, k$. Then we can enumerate all extreme points $(\boldsymbol{\pi}_i^1, \ldots, \boldsymbol{\pi}_i^{P_i})$, and all extreme rays $(\boldsymbol{\pi}_i^1, \ldots, \boldsymbol{\pi}_i^{R_i})$ of the feasible region in (A.4), where P_i and R_i are the number of extreme points and extreme rays of the i-th subproblem respectively. Then for a given $\mathbf{x}$, the i-th dual problem can be solved by (i) checking whether $(\mathbf{d_i} - B_i\mathbf{x})^{\mathrm{T}}\boldsymbol{\pi}_i^r < 0$ for some extreme ray $\boldsymbol{\pi}_i^r$, in which case (A.4) is unbounded and the primal formulation is infeasible, and (ii) finding an extreme point $\boldsymbol{\pi}_i^p$ that minimizes the value of the objective function $(\mathbf{d_i} - B_i\mathbf{x})^{\mathrm{T}}\boldsymbol{\pi}_i^p$, in which case both the primal and dual formulations have finite optimal solutions. Then the dual problem (A.4) can be reformulated as follows:

$$\phi_i(\mathbf{x}) = \begin{array}{ll} \max_{\phi_i} & \phi_i \\ s.t. & (\mathbf{d_i} - B_i\mathbf{x})^{\mathrm{T}}\boldsymbol{\pi}_i^r \geq 0, \ \ \forall r = 1, \ldots, R_i \\ & (\mathbf{d_i} - B_i\mathbf{x})^{\mathrm{T}}\boldsymbol{\pi}_i^p \geq \phi_i, \ \ \forall p = 1, \ldots, P_i \end{array} \tag{A.5}$$

We can replace $\phi_i(\mathbf{x})$ in (A.2) with (A.5) and obtain a reformulation of the original problem in terms of $\mathbf{x}$ and $\phi_1, \ldots, \phi_k$:

$$\begin{array}{ll} \max_{\mathbf{x},\phi_1,\ldots,\phi_k} & \mathbf{c}^{\mathrm{T}}\mathbf{x} + \sum_{i=1}^k \phi_i \\ s.t. & A\mathbf{x} \leq \mathbf{b}, \ \ \mathbf{x} \geq \mathbf{0} \\ & (\mathbf{d_i} - B_i\mathbf{x})^{\mathrm{T}}\boldsymbol{\pi}_i^r \geq 0, \ \ \forall i = 1, \ldots, k \ \ \forall r = 1, \ldots, R_i \\ & (\mathbf{d_i} - B_i\mathbf{x})^{\mathrm{T}}\boldsymbol{\pi}_i^p \geq \phi_i, \ \ \forall i = 1, \ldots, k \ \ \forall p = 1, \ldots, P_i \end{array} \tag{A.6}$$

Since there are typically an exponential number of extreme points and extreme rays of the dual formulation (A.4), generating all constraints for (A.6) is not realistic. Instead Bender's decomposition starts with a subset of these constraints, and solves a relaxed master problem, which yields a candidate optimal solution $(\mathbf{x}^*, \phi_1^*, \ldots, \phi_k^*)$. Then we can solve the dual subproblem (A.4) to calculate $\phi_i(\mathbf{x}^*)$. If for any $i = 1, \ldots, k$, the i-th subproblem has an optimal solution such that $\phi_i(\mathbf{x}^*) = \phi_i^*$, then the algorithm stops and $\mathbf{x}^*$ is the optimal solution of the original problem (A.1).

Otherwise, there exists at least one subproblem i such that (A.4) is unbounded or (A.4) is bounded with $\phi_i(\mathbf{x}) < \phi_i^*$. If the i-th dual subproblem is unbounded, then an extreme ray $\boldsymbol{\pi}_i^r$ is obtained and therefore the constraint $(\mathbf{d_i} - B_i\mathbf{x})^{\mathrm{T}}\boldsymbol{\pi}_i^r \geq 0$ should be added to the relaxed master problem. This type of constraints is referred to as the Bender's feasibility cuts because they enforce necessary conditions for feasibility of the primal subproblems (A.3). On the other hand, if the i-th dual subproblem has an optimal solution such that $\phi_i(\mathbf{x}) < \phi_i^*$, then an extreme point $\boldsymbol{\pi}_i^p$ is obtained and the constraint $(\mathbf{d_i} - B_i\mathbf{x})^{\mathrm{T}}\boldsymbol{\pi}_i^p \geq \phi_i$ should be added to the relaxed master problem.

This type of constraints is referred to as the Bender's optimality cuts because they enforce the necessary conditions for optimality of the subproblems.

Mutliple constraints can be generated in each iteration if there are multiple subproblems that are unbounded or have $\phi_i(\mathbf{x}) < \phi_i^*$. After adding these constraints, we solve the new relaxed master problem and repeat the process. Since P_i and R_i are finite for each subproblem i and at least one new Bender's cut is generated in each iteration, it can be concluded that the algorithm will converge in a finite number of iterations, i.e., at most $\sum_{i=1}^{k} P_i + \sum_{i=1}^{k} R_i$ iterations. In practice, the number of iterations needed until convergence is orders of magnitude smaller than the total number of extreme points and extreme rays, and therefore applying Bender's decomposition by solving (A.2) and (A.4) iteratively is often significantly more efficient than solving the original linear program (A.1) directly.