Towards Addressing Spatio-Temporal Aspects in Security Games

by

Fei Fang

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

August 2016

Acknowledgements

Having now gone through the process, I would like to express my gratitude to those who helped me reach this point. I have enjoyed the past five years with the privilege to have shared experiences, conservations and relationships with a number of extraordinary people along the way.

I would like to start by thanking my advisor Milind Tambe. Five years ago, I joined Teamcore without much knowledge about research. Thank you for seeing the potential in me and guiding me through every step of doing research. Thank you for the many opportunities that you provided me, including working on various great projects such as PROTECT-Ferry and PAWS, collaborating with researchers within and outside our field, and supporting me for attending conferences all around the world. Your passion and devotion to research have inspired me and is something I will strive to emulate in my career. Your commitment to working on real problems and making a real-world impact has shaped my philosophy of research. I learned so much from you, not only about research but also how to be a good mentor with care and patience. I will be forever grateful for your heartfelt advice for the future career path. Your commitment to your students extends long after graduation, and the Teamcore family network continues to show its strength over time.

I would like to thank my committee members: Shaddin Dughmi, Leana Golubchik, Jelena Mirkovic and Suvrajeet Sen, not only for serving on my qualifying committee but also for providing input and suggestions to my research and future career. Your guidance is so valuable to me, and I am thankful for all your time and efforts on helping and supporting me.

I would like to thank United States Coast Guard personnel Craig Baldwin for your consistent support of the PROTECT project and in particular the PROTECT-Ferry application and for making the application a success. Second, I would like to thank Rob Pickles, Wai Y. Lam and Gopalasamy R. Clements from two non-government organizations Panthera and Rimba for their input and help in making the PAWS application a success including a regular deployment in the field. Also, I would like to thank Arnaud Lyet and Barney Long from World Wildlife Fund for their helpful feedback and Andrew Lemieux for his support for the field test of PAWS. And I would like to thank our partners in the field who made the test and deployment of the PAWS application possible. I am also grateful for the support from the Army Research Office on my research over the years.

During my time at USC, I had the privilege of collaborating with many excellent researchers: Peter Stone, Janusz Marecki, Vincent Conitzer, Heidi J. Albers, Rajiv Maheswaran, Nicole Sintov, Christopher Kiekintveld, Bo An, Albert Jiang, Matthew P. Johnson, Francesco M. Delle Fave, William Haskell, Matthew Brown and Arunesh Sinha. I thank you for your guidance and insights on the projects we worked on as well as the patience you showed in your mentoring.

Being in a large research group, I was fortunate enough to share the Ph.D. experience with a group of bright and talented fellow students who all became my friends: Manish Jain, James Pita, Jun-young Kwak, Rong Yang, Zhengyu Yin, Jason Tsai, Eric Shieh, Thanh Nguyen, Leandro Marcolino, Chao Zhang, Yundi Qian, Debarun Kar, Benjamin Ford, Haifeng Xu, Amulya

Yadav, Aaron Schlenker, Sara Mc Carthy, Yasaman D. Abbasi, Shahrzad Gholami, Bryan Wilder and Elizabeth Orrico. Thank you for making my life full of laugh and excitement and for always being ready to help. I will always remember sharing offices, traveling to conferences and grinding through conference deadlines and attending group seminars and retreats. Special thanks to Manish Jain for all the advice you have given to me over the years and Rong Yang for all your kind help, and especially for your help on my research and life during my first year.

Also, I want to thank all the people that I had the privilege to mentor: Dana Thomas, Brian Schwedock, Amandeep Singh, Matthew Burke, David Liao, Kevin Hoffman and Jewels Kovach. It was a great experience working with you, and I learned a lot from you and the mentoring experience.

I would like to thank Yu Cheng for being a constant source of support and encouragement. This thesis would have never been possible without you. You have shared this entire journey with me, and you have been my rock along the way.

Lastly, I would like to thank my parents for the love and support you have provided me over the years. Thank you for always believing in me, supporting my decisions and sharing my good times and bad times.

Contents

List Of Figures

List Of Tables

Abstract

Game theory has been successfully used to handle complex resource allocation and patrolling problems in security and sustainability domains. More specifically, real-world applications have been deployed for different domains based on the framework of security games, where the defender (e.g., security agency) has a limited number of resources to protect a set of targets from an adversary (e.g., terrorist). Whereas the first generation of security games research provided algorithms for optimizing security resources in mostly static settings, my thesis advances the state-of-the-art to a new generation of security games, handling massive games with complex spatio-temporal settings and leading to real-world applications that have fundamentally altered current practices of security resource allocation. Indeed, in many real-world domains, players act in a geographical space over time, and my thesis is then to expand the frontiers of security games and to deal with challenges in domains with spatio-temporal dynamics. My thesis provides the first algorithms and models for advancing key aspects of spatio-temporal challenges in security games, including (i) continuous time; (ii) continuous space; (iii) frequent and repeated attacks; (iv) complex spatial constraints.

First, focusing on games where actions are taken over continuous time (for example games with moving targets such as ferries and refugee supply lines), I propose a new game model that accurately models the continuous strategy space for the attacker. Based on this model, I provide

an efficient algorithm to calculate the defender's optimal strategy using a compact representation for both the defender and the attacker's strategy space. Second, for games where actions are taken over continuous space (for example games with forest land as a target), I provide an algorithm computing the optimal distribution of patrol effort. Third, my work addresses challenges with one key dimension of complexity – frequent and repeated attacks. Motivated by the repeated interaction of players in domains such as preventing poaching and illegal fishing, I introduce a novel game model that deals with frequent defender-adversary interactions and provide algorithms to plan effective sequential defender strategies. Furthermore, I handle complex spatial constraints that arise from the problem of designing optimal patrol strategy given detailed topographical information.

My thesis work has led to two applications which have been deployed in the real world and have fundamentally altered previously used tactics, including one used by the US Coast Guard for protecting the Staten Island Ferry in New York City and another deployed in a protected area in Southeast Asia to combat poaching.

Chapter 1

Introduction

Security and sustainability challenges exist all around the world. These challenges include protecting critical infrastructure and transportation networks, preventing intrusions in cyber systems, as well as protecting environmental resources, saving endangered wildlife from poaching and stopping illegal fishing. A unifying theme in these challenges is the strategic reasoning between the law enforcement agencies and the adversaries such as terrorists and poachers. The law enforcement agencies only have limited resources, and it is not possible to protect everything at all times; and at the same time, the adversaries can conduct surveillance to observe the agencies' actions. Therefore, any deterministic allocation of resources can be exploited by the adversaries, and it is important for the agencies to introduce randomness to the allocation.

Game theory has become a well-established paradigm for modeling complex resource allocation and scheduling problems in security and sustainability domains (Tambe, 2011; Gatti, 2008; Agmon, Kraus, & Kaminka, 2008; Basilico, Gatti, & Amigoni, 2009). One game-theoretic model that has received significant attention is the Stackelberg security game (denoted as SSG) model with two players, the defender, and the attacker. In SSG, the defender needs to allocate and schedule their limited resources to protect a set of targets from the attacker. The defender

commits to a *mixed strategy*, which is a randomized schedule specified by a probability distribution over deterministic schedules; the attacker then observes the distribution and plays the best response (Korzhyk, Conitzer, & Parr, 2010a). Models and algorithms have been proposed to compute the optimal strategy for the defender efficiently to address real-world challenges, forming the first generation of security games (Jain, 2013; Yin, 2013; Pita, 2012; Yang, 2014; Shieh, 2015; Brown, 2015). Decision-support systems based on SSG and the proposed algorithms have been successfully deployed in several domains to assist security agencies (Pita, Jain, Marecki, Ordóñez, Portway, Tambe, Western, Paruchuri, & Kraus, 2008a; Tsai, Rathi, Kiekintveld, Ordonez, & Tambe, 2009a; Shieh, An, Yang, Tambe, Baldwin, DiRenzo, Maule, & Meyer, 2012a; Yin, Jiang, Johnson, Kiekintveld, Leyton-Brown, Sandholm, Tambe, & Sullivan, 2012b).

In most previous work on security games, only limited spatio-temporal aspects have been considered: First, the targets are assumed to be stationary (e.g., airport terminals (Pita, Jain, Marecki, Ordóñez, Portway, Tambe, Western, Paruchuri, & Kraus, 2008b)), or stationary relative to the defender and the attacker (e.g., trains (Yin & Tambe, 2012) and planes (Tsai, Rathi, Kiekintveld, Ordonez, & Tambe, 2009b), where the players can only move along with the targets to protect or attack them). Second, the targets are assumed to be discretized, such as airport terminals and different regions in port. Third, it is assumed that the attackers can conduct long-term surveillance to understand the defender's strategy and then plan for a one-shot attack. Fourth, spatial constraints that restrict the movement of the defender's resources (e.g., patrollers) are often ignored or significantly simplified.

However, as the security game model has evolved, there has been a push towards increasingly complex security domains where these simple assumptions might not be sufficient. My thesis focuses on such domains and addresses the research challenges raised by introducing more

(a) Ferry protection

(b) Forest protection

(c) Wildlife protection

(d) Fishery protection

Figure 1.1: Different domains with complex spatio-temporal aspects.

complex spatio-temporal aspects. The first part of my thesis considers spatio-temporal continuity of targets and players' action space in security games. The second part of my thesis focuses on spatio-temporal dynamics in "green (environmental)" security domains such as preventing poaching and illegal fishing, where frequent and repeated attacks and complex spatial constraints are involved. The technical contributions of my thesis include models and algorithms for addressing these spatio-temporal aspects.

1.1 Spatio-temporal Continuity

The first part of my thesis focuses on spatio-temporal continuity in security games. In many real-world scenarios, players act in a geographical space over time even considering a one-shot attack, and these scenarios would lead to a need of considering the continuous time and space.

One important scenario is moving target protection, where the defender has mobile resources to protect targets that are moving according to a known daily schedule and with changing utility values. This scenario captures domains such as protecting ferry systems from potential attacks and protecting refugee supply lines. The changing position of the targets makes it necessary to reason about the continuous space and time, which leads to significant challenges in representing the players' strategies and calculating the equilibrium. For example, the action set for the attacker has infinite actions, as he can choose not only which target to attack, but also when to attack. Such infinite action space due to continuous time and space was not considered in previous work in security games.

Another important scenario where continuous space needs to be considered is protecting a large area from intruders. The intruders' reward is not only measured by whether or not he intrudes the area without being captured by the patroller, but also how far he intrudes. This scenario abstracts the problem of protecting a forest land from excessive fuel-wood extraction where the rangers aim to deter extractors who intrude the forest from the boundary of the protected land. Calculating the optimal strategies for the players is a significant challenge as the action sets for both players have infinite cardinality: the extractors need to decide how far to intrude, and the rangers need to plan their patrol effort within the whole area.

To handle the infinite action space introduced by spatio-temporal continuity in security games, one approach that is often used is discretization (Yang, Ford, Tambe, & Lemieux, 2014a). In some problems, the discretization needs to be sufficiently fine-grained to ensure the solution quality. However, a fine-grained discretization can still lead to a large action space for the players, making the calculation of optimal strategies computationally prohibitive. Also, in some problems, no matter how fine-grained the discretization is, it may still lead to sup-optimal solutions. My

main contributions in this part include exploiting the spatio-temporal structure to handle the infinite action space directly and to handle the large action space after discretization efficiently. The key ideas include reducing the number of actions to be considered based on the dominance relationship or equivalence relationship and analyzing the properties of the optimal strategy to reduce the search space. Both ideas aim to abstract the players' action space before calculating the equilibrium strategy.

In dealing with the problem of moving target protection, I introduce a novel game model that considers an SSG with a continuous set of actions for the attacker. In contrast, while the defender's action space is also continuous, it is discretized for three reasons. Firstly, if the defender's action space is continuous, the space of mixed strategies for the defender would then have infinite dimensions, which makes exact computation infeasible. Secondly, in practice, the defender's mobile resources (i.e., patrol boats for protecting ferries) may not be able to have infinitely fine-grained control over their movement, which makes the actual defender's action space effectively a discrete one. Finally, the discretized defender action space is a subset of the original continuous defender action space, so the optimal defender strategy calculated in the discretized defender action space is a feasible solution in the original game and gives a lower-bound guarantee for the defender.

Given this model, I propose CASS (Solver for Continuous Attacker Action Set), an efficient linear program to exactly solve the proposed game model. In this solver, I represent the defender's mixed strategies as marginal probability variables, and this compact representation significantly reduces the number of variables needed to describe a defender strategy under discretization. In addition, I handle the infinite attacker actions over continuous time directly by partitioning the attacker's action set. I show that due to the spatio-temporal structure of the problem, only one time

point could potentially be chosen by a best-responding attacker for each target in each partitioned subset. An attacker is best-responding if he tries to attack a target at a time point that gives him the highest expected utility. Therefore, it is sufficient to consider a finite number of attacker actions when calculating the optimal defender strategy. Additional contributions include equilibrium refinement in the game and patrol route sampling. The game may have multiple equilibria and the defender strategy found by CASS can be suboptimal with respect to uncertainties in the attacker's model, e.g. if the attacker can only attack during certain time intervals. I propose two heuristic equilibrium refinement approaches. The first, *route-adjust*, iteratively computes a defender strategy that dominates earlier strategies. The second, *flow-adjust*, is a linear-programming-based approach. An application built based on the work has been deployed for protecting the Staten Island Ferry and is used by US Coast Guard since April 2013.

For area protection, the solution stems from the cost-benefit analysis, which leads to the key observation that in equilibrium, the attacker chooses a distance with the highest net benefit and the defender allocates the patrol resources to make the attacker's *marginal* net benefit be zero beyond the equilibrium distance. That is, the attacker's every additional step beyond the equilibrium distance would lead to zero net benefit. We prove that there exists one optimal strategy that satisfies this property, and we aim to find such strategy. With this analysis, the search space for the optimal patrol strategy is reduced significantly since the strategy can be characterized by the distance from which the defender starts to allocate patrol resources. Our work provides an efficient algorithm to find the optimal patrol strategy, a simple and practical approximation with a theoretical bound, as well as closed-form solutions for special cases.

1.2 Novel Challenges in Green Security Domains

The second part of my thesis focuses on spatio-temporal dynamics in green security domains such as protecting wildlife and preventing illegal fishing. Same as infrastructure security domains, the law enforcement agencies have limited resources, and game theory can be brought into the field to fight against the illegal activities such as poaching and illegal fishing. However, these green security domains are very different compared to infrastructure security for several main reasons. First, frequent and repeated attacks are involved. For example, poachers place snares on the ground to trap animals. More than one thousand snares can be found annually in a conservation area in Uganda. If we consider the problem as a game, it is no longer a one-shot game. In addition, the frequent attacks would bring in more attack data that can be exploited by the defender. The second main difference is in attacker's decision making. The attacks take place frequently, so it's impossible for the attacker to conduct long-term surveillance before each of the attacks. The attacker may even have a lagged understanding of the defender's strategy. Also, due to frequent attacks and the relatively low cost of failure, the attacker will take less effort in planning the attacks and may be boundedly rational in their decision making. Third, the feasibility of a patrol route is often constrained by various spatial constraints. For example, it is important to take into account the geographical information and design routes that can be followed by patrollers.

With all these differences, new models are needed for green security domains and the defender faces a much more complex optimization problem. My contributions in this part include models and algorithms to handle challenges introduced by these difference.

I proposed Green Security Games (GSGs), a novel game model that considers frequent and repeated attacks and a novel behavior model for the attacker. In green security domains, the

attackers can repeatedly and frequently perform attacks. On the other hand, this makes it possible for the defender to exploit the attackers' temporal attack pattern and benefit from changing her strategy from time to time. GSG generalizes the standard Stackelberg assumption in security games and instead assumes that the attackers' understanding of the defender strategy can be approximated as a convex combination of the defender strategies used in several recent rounds. Based on this model, I proposed two sets of algorithms that plan ahead, providing defender strategies in each round and I further provide a novel framework that incorporates learning with our planning framework.

In bringing the defender strategy to the real world, one important aspect to be considered is spatial constraints. For example, efforts have been made by law enforcement agencies in many countries to protect endangered animals; the most commonly used approach is conducting foot patrols. However, for human patrollers who move in conservation areas with complex terrain, a defender strategy in the form of coverage probability is not sufficient. More guidance should be provided, and a desirable defender strategy should include a set of complete and detailed patrol routes that are compatible with the terrain. In my thesis, I incorporate the topographical information into the game model and handle the spatial constraints brought by it.

The main technical challenge of addressing the spatial constraints is that considering detailed topographical information leads to a need for a fine-grained discretization, which makes the calculation of optimal defender strategy computationally prohibitive. To address this challenge, I use a hierarchical modeling approach and combine grid-based discretization and graph representation. I first apply a coarse grid-based discretization for the whole area, and then represent each discretized region as a sub-graph and connect the sub-graphs to get a large graph – a virtual street map of the area. More specifically, I build the sub-graphs of the discretized regions based on the

terrain features such as ridgeline and streams, as they are important conduits for animals' movement and patrollers should focus on these features during their patrols. With the street map, the number of patrol routes for the defender is significantly reduced, making it possible to calculate the optimal defender strategy efficiently.

1.3 Thesis Overview

The structure of the thesis is organized as follows: Chapter 2 discusses background material for Stackelberg security games. Chapter 3 reviews related work to provide the context for the contributions of the thesis. Chapter 4 considers security games continuous time (for example games with moving targets such as ferries and refugee supply lines). Chapter 5 investigates how to reason about continuous space in security games (for example games with forest land as a target). Chapter 6 explores the problem of frequent and repeated attacks, motivated by green security domains such as preventing poaching and illegal fishing. Chapter 7 examines how to handle complex spatial constraints that arise from the problem of designing optimal patrol strategy given complex topographical information. Chapter 8 summarizes the thesis and presents possible directions for future work.

Chapter 2

Background

2.1 Security Games

A security game (Conitzer & Sandholm, 2006; Kiekintveld, Jain, Tsai, Pita, Ordóñez, & Tambe, 2009a; Paruchuri, Pearce, Marecki, Tambe, Ordonez, & Kraus, 2008) is a two-player game between a defender and an attacker, where the defender must protect a set of N targets from the attacker. The defender tries to prevent attacks using K defender resources. A pure strategy for the defender is typically an assignment of the K resources to either patrols or targets (depending on the type of the game) while a pure strategy for the adversary is typically the target that is to be attacked. Denote the k^{th} defender pure strategy as A_k, which is an assignment of all the security resources. A_k is represented as a column vector $A_k = \langle A_{ki} \rangle^T$, where A_{ki} indicates whether target i is covered by A_k. For example, in a game with four targets and two resources, $A_k = \langle 1, 1, 0, 0 \rangle$ represents the pure strategy of assigning one resource to target 1 and another to target 2. Each target $i \in [N]$ is assigned a set of payoffs $\{P_i^a, R_i^a, P_i^d, R_i^d\}$: If an attacker attacks target i and it is protected by a defender resource, the attacker gets utility P_i^a (P stands for penalty) and the defender gets utility R_i^d (R stands for reward). If target i is not protected, the

attacker gets utility R_i^a and the defender gets utility P_i^d. In order to be a valid security game, it must hold that $R_i^a > P_i^d$ and $R_i^d > P_i^d$, which means that assigning a resource to cover a target more often is always beneficial for the defender and disadvantageous for the adversary. Careful planning by the defender is necessary as the amount of available security resources is limited, i.e., $K < N$, and not all targets can be covered.

Most work on security games has used the Stackelberg assumption, i.e., the defender commits to a strategy first. The adversary is then able to conduct surveillance and thus learn the defender's strategy before selecting their strategy. The game is then denoted as a Stackelberg security game (SSG) and the standard solution concept the Strong Stackelberg Equilibrium (SSE), in which the defender selects an optimal strategy based on the assumption that the adversary will choose an optimal response, breaking ties in favor of the defender. In an SSG, the optimal resource allocation strategy for the defender will usually be a mixed (randomized) strategy $\mathbf{a}$, which is a distribution over the set of pure defender strategies A, as any deterministic defender strategy would easily be exploited by the adversary. The defender's mixed strategy can then be represented as a vector $\mathbf{a} = \langle a_k \rangle$, where $a_k \in [0, 1]$ is the probability of choosing pure strategy A_k. There is also a more compact "marginal" representation for defender strategies. Let $\boldsymbol{x}$ be the marginal strategy, where $x_i = \sum_{A_k \in \mathcal{A}} a_k A_{ki}$ is the probability that target i is covered. Thus, depending on the particular type of security game, the defender is trying to find either the optimal mixed strategy $\mathbf{a}$ or marginal strategy $\mathbf{x}$.

There have been many algorithms and models developed to solve SSGs, including DOBSS (Paruchuri et al., 2008) which solves SSGs using a mixed-integer linear program, ORIGAMI (Kiekintveld et al., 2009a) which provides a polynomial time algorithm for SSGs that contain no

scheduling constraints. However, these algorithms do not apply to security games with spatio-temporal aspects associated with moving targets, continuous target, and frequent adversary interaction.

2.2 Human Behavior Models

One of the strongest assumptions in classic game theory is that the players are perfectly rational utility maximizer. However, it is well-understood that human beings are boundedly rational, and thus, the attackers may not always choose to attack the target with the highest expected utility, i.e., best respond to the defender's strategy. Incorporating human behavioral models (McKelvey & Palfrey, 1995) in decision making into security games has been demonstrated to improve the performance of defender patrol strategies in both simulations and human subject experiments (Pita, Jain, Ordonez, Tambe, & Kraus, 2010; Yang, Ordonez, & Tambe, 2012; Nguyen, Yang, Azaria, Kraus, & Tambe, 2013c).

(Yang et al., 2012) was the first to address human adversaries in security games by incorporating quantal response (QR) model (McKelvey & Palfrey, 1995) from the behavioral economics literature. Instead of choosing the action with the highest utility, QR model predicts a probability distribution over adversary actions where actions with higher utility have a greater chance of being chosen.

(Nguyen et al., 2013c) extended the QR model by proposing that humans use "subjective utility", a weighted linear combination of features (such as defender coverage, adversary reward, and adversary penalty), to make decisions. This subjective utility quantal response (SUQR) model

was shown to outperform QR in human subject experiments. As a result, most subsequent research on boundedly rational human adversaries in security games has focused on the SUQR model. In this model, an attacker's choice is based on an evaluation of key properties of each target, including the coverage probability, the reward and the penalty, represented by the parameter vector $\omega = (\omega_1, \omega_2, \omega_3)$. If the attackers respond to defender strategy η, the probability that an attacker with parameter ω attacks target i is

$$q_i(\omega, \eta) = \frac{e^{\omega_1 \eta_i + \omega_2 R_i^a + \omega_3 P_i^a}}{\sum_j e^{\omega_1 \eta_j + \omega_2 R_j^a + \omega_3 P_j^a}} \tag{2.1}$$

When there are multiple attackers, Bayesian SUQR model (Yang et al., 2014a) is proposed based on the SUQR model. It captures the heterogeneity of a group of attackers and assumes that different attackers have different parameters.

Chapter 3

Related Work

3.1 Stackelberg Security Games

Stackelberg games have been widely applied to security domains, although most of this work has considered static targets (e.g., Korzhyk et al., 2010a; Krause, Roper, & Golovin, 2011; Letchford & Vorobeychik, 2012; Kiekintveld, Islam, & Kreinovich, 2013). (Agmon et al., 2008) proposed algorithms for computing mixed strategies for setting up a perimeter patrol in adversarial settings with mobile robot patrollers. Similarly, (Basilico et al., 2009) computed randomized leader strategies for robotic patrolling in environments with arbitrary topologies. Even when both of the players are mobile, e.g., in hider-seeker games (Halvorson, Conitzer, & Parr, 2009), infiltration games (Alpern, 1992) or search games (Gal, 1980), the targets (if any) were assumed to be static. (Tsai et al., 2009b) applied Stackelberg games to the domain of scheduling federal air marshals on board flights. The targets (i.e., flights) in this domain are mobile, but the players are restricted to move along the targets to protect or attack them. This stationary nature leads to discrete game models with finite numbers of pure strategies.

(Bošanský, Lisý, Jakob, & Pěchouček, 2011) and (Vaněk, Jakob, Hrstka, & Pěchouček, 2011) studied the problem of protecting moving targets. However, they both considered a model in which the defender, the attacker, and the targets have discretized movements on a directed graph. Such discretization of attacker strategy spaces can introduce suboptimality in the solutions as we have shown with DASS. We, in our work, generalize the strategy space of the attacker to the continuous realm and compute optimal strategies even in such a setting. Furthermore, while we provide an efficient and scalable linear formulation, (Bošanský et al., 2011) presented a formulation with non-linear constraints, which faced problems scaling up to larger games even with a single defender resource.

(Yin & Tambe, 2012) considered the domain of patrolling in public transit networks (such as the LA Metro subway train system) in order to catch fare evaders. Because the players ride trains that follow a fixed schedule, the domain is inherently discrete, and they modeled the patrolling problem as a finite zero-sum Bayesian game. (Yin & Tambe, 2012) proposed a compact representation for defender mixed strategies as flows in a network. We adapt this compact representation idea to a continuous domain. In particular, in our domain, we need to model the interaction between the defender's flow and attacker's continuous strategy space. Our proposed sub-interval analysis used spatio-temporal reasoning to efficiently reduce the problem into a finite LP.

There is an extensive literature on equilibrium refinement; however most existing work on the computation of equilibrium refinement focuses on finite games. For simultaneous-move finite games, solution concepts such as perfect equilibrium and proper equilibrium were proposed as refinements of Nash equilibrium (Fudenberg & Tirole, 1991). (Miltersen & Sørensen, 2007) proposed an efficient algorithm for computing proper equilibria in finite zero-sum games. For finite security games, (An, Tambe, Ordóñez, Shieh, & Kiekintveld, 2011) proposed a refinement of

Stackelberg equilibrium and techniques for computing such refinements. The resulting defender strategy is robust against possibilities of constrained capabilities of the attacker. These existing approaches rely on the finiteness of action sets and is thus not applicable to our setting. (Simon & Stinchcombe, 1995) proposed definitions of perfect equilibrium and proper equilibrium for infinite games with continuous strategy sets. However, they did not propose any computational procedure for the resulting solution concepts. Exact computation of equilibrium refinements of continuous games remains a challenging open problem.

There is a rising interest in applying game theory to green security domains, e.g., protecting fisheries from over-fishing (Brown, Haskell, & Tambe, 2014; Haskell, Kar, Fang, Tambe, Cheung, & Denicola, 2014a) and protecting wildlife from poaching (Yang et al., 2014a). However, previous work in green security domains (Yang, Ford, Tambe, & Lemieux, 2014b; Haskell, Kar, Fang, Tambe, Cheung, & Denicola, 2014b) models the problem as a game with multiple rounds and each round is an SSG (Yin, Korzhyk, Kiekintveld, Conitzer, & Tambe, 2010) where the defender commits to a mixed strategy and the attackers respond to it. However, these efforts share the standard Stackelberg assumption that the defender's mixed strategy is fully observed by the attacker via extensive surveillance before each attack. This assumption can be unrealistic in green security domains due to the frequent and repeated attacks. Due to this assumption, previous efforts do not engage in any planning and instead rely only on designing strategies for the current round.

The bounded rationality of attackers has been studied extensively in the context of Stackelberg security games. Instead of always choosing to attack the target with the highest expected utility, the attackers may choose sub-optimal targets. It has been shown that considering human behavioral models (McKelvey & Palfrey, 1995) when designing defender strategies in security

games can significantly improve the performance of defender in human subject experiments (Pita et al., 2010; Yang et al., 2012; Nguyen, Yang, Azaria, Kraus, & Tambe, 2013a). In addition, the bounded rationality of attackers has also been studied in green security problems, and new human behavior models are proposed to incorporate heterogeneity among a population of attackers (Yang et al., 2014a). However, previous work does not consider the attacker's lagged understanding of the defender strategy and bounded memory. By embedding these factors, we complement previous work that focuses on modeling human bounded rationality and bounded memory (Rubinstein, 1997; Cowan, 2005).

Previous work on learning in repeated SSGs (Marecki, Tesauro, & Segal, 2012; Letchford, Conitzer, & Munagala, 2009; Blum, Haghtalab, & Procaccia, 2014) has mainly focused on learning the payoffs of attackers assuming perfectly rational attackers. In contrast, we not only generalize the Stackelberg assumption to fit green security domains but also provide algorithms to learn the parameters in the attackers' bounded rationality model. While the work by (Yang et al., 2014a) do exploit the available attack data, they use Maximum Likelihood Estimation (MLE) to learn the parameters of the SUQR model for individual attackers which may lead to skewed results.

3.2 Continuous Strategy Space in Games

Games with continuous strategy spaces have been well-studied in game theory. Much of the economics literature has focused on games whose equilibria can be solved analytically (and thus the question of computation does not arise), for example, the classical theory of auctions (see e.g., Krishna, 2009). Recent computational approaches for the analysis and design of auctions

have focused on discretized versions of the auction games (e.g., Thompson & Leyton-Brown, 2009; Daskalakis & Weinberg, 2012). There has been research on efficiently solving two-player continuous games with specific types of utility functions, such as zero-sum games with convex-concave utility functions (Owen, 1995) and separable continuous games with polynomial utility functions (Stein, Ozdaglar, & Parrilo, 2008). However, the approaches to handle the continuous strategy space cannot be directly applied to security games with complex spatio-temporal settings.

Hybrid games and timed games in control problems have also been studied with an investigation into the continuous timeline (Henzinger, Horowitz, & Majumdar, 1999; De Alfaro, Henzinger, & Majumdar, 2001; Platzer, 2015). To address the infinite number of possible actions over the continuous timeline, region construction is used. In timed games, the options are partitioned into equivalent classes of states based on bisimilarity, similarity, and trace equivalence, forming a finite number of regions in the action space. The region construction in timed games and the compact representation in my work share the spirit of exploiting the equivalent class of strategies. Also, the region construction and the partitioning of the attacker's strategy space in my work share the high-level intuition of reducing the number of actions to be considered from infinite to finitely many through exploiting the problem structure. However, the underlying techniques cannot apply directly. In my work, I partition the continuous timeline into a finite set of zones and consider only one best time point in each zone by exploiting the dominance relationship within each zone.

3.3 Planning and Learning in Repeated Games

Planning and learning in repeated games against opponents with bounded memory has been studied (Sabourian, 1998; Powers & Shoham, 2005; Chakraborty, Agmon, & Stone, 2013; de Cote &

Jennings, 2010; Banerjee & Peng, 2005). However, most of the work considers the case where each player chooses *one* action from his finite action set in each round of the game. In addition, there is no delay in observing the other player's action. Therefore, on-line learning based approaches can be used to find a good strategy against the opponent. Instead, we focus on the problem motivated by real-world green security challenges where the players can choose a *mixed strategy* and implement it for multiple episodes in each round. Previous approaches fail to apply in such settings, which is a better fit for the green security problems.

Chapter 4

Reasoning in Continuous Time

My thesis has addressed reasoning in continuous time when the defender has mobile resources to protect moving targets. One major example of the practical domains motivating this work is the problem of protecting ferries that carry passengers in many waterside cities. Packed with hundreds of passengers, these may present attractive targets for an attacker. For example, the attacker may ram a suicide boat packed with explosives into the ferry as happened with attacks on French supertanker Limburg and USS Cole (Greenberg, Chalk, & Willis, 2006). In this case, the intention of the attacker can only be detected once he gets very close to the ferry. Small, fast and well-armed patrol boats (patrollers) can provide protection to the ferries, by detecting the attacker and stopping him with the armed weapons. However, there are often limited numbers of patrol boats, i.e., they cannot protect the ferries at all times at all locations.

Most previous work on game-theoretic models for security has assumed either stationary targets such as airport terminals (Pita et al., 2008b), or targets that are stationary relative to the defender and the attacker, e.g., trains (Yin & Tambe, 2012) and planes (Tsai et al., 2009b), where the players can only move along with the targets to protect or attack them). This stationary nature leads to discrete game models with finite numbers of pure strategies. In contrast, the attacker

in this problem can attack these targets at any point in time during their movement, leading to a continuous set of strategies. The defender can deploy a set of mobile defender resources (called patrollers for short) to protect these targets. The patrollers' movement is constrained by a speed limit and a patroller can provide protection to targets within a known protection radius. The values/utilities of the targets may vary depending on their locations and time. Along the line with previous work (Tambe, 2011; Yin & Tambe, 2012; Kiekintveld et al., 2009a), the attackers in this problem are assumed to be perfectly rational and will choose to attack a target at a time that is most favorable to him (i.e., with the highest expected utility). The defender's objective is to schedule the patrollers to minimize attacker's maximal expected utility (zero-sum game).

The first contribution of this chapter is a novel game model for MRMT called MRMT_{sg}. MRMT_{sg} is an attacker-defender Stackelberg game model with a continuous set of strategies for the attacker. In contrast, while the defender's strategy space is also continuous, we discretize it in MRMT_{sg} for three reasons. Firstly, if we let the defender's strategy space to be continuous, the space of mixed strategies for the defender would then have infinite dimensions, which makes exact computation infeasible. Secondly, in practice, the patrollers are not able to have such fine-grained control over their vehicles, which makes the actual defender's strategy space effectively a discrete one. Finally, the discretized defender strategy space is a subset of the original continuous defender strategy space, so the optimal solution calculated under our formulation is a feasible solution in the original game and gives a lower-bound guarantee for the defender in terms of expected utility for the original continuous game. On the other hand, discretizing the attacker's strategy space can be highly problematic as we will illustrate later in this chapter. In particular, if we deploy a randomized schedule for the defender under the assumption that the attacker could

only attack at certain discretized time points, the actual attacker could attack at some other time point, leading to a possibly much worse outcome for the defender.

Our second contribution is CASS (Solver for Continuous Attacker Strategies), an efficient linear program to exactly solve MRMT_{sg}. Despite discretization, the defender strategy space still has an exponential number of pure strategies. We overcome this shortcoming by compactly representing the defender's mixed strategies as marginal probability variables. On the attacker side, CASS exactly and efficiently models the attacker's continuous strategy space using *sub-interval analysis*, which is based on the observation that given the defender's mixed strategy, the attacker's expected utility is a piecewise-linear function. Along the way to presenting CASS, we present DASS (Solver for Discretized Attacker Strategies), which finds minimax solutions for MRMT_{sg} games while constraining the attacker to attack at discretized time points. For clarity of exposition, we first derive DASS and CASS for the case where the targets move on a one-dimensional line segment. We later show that DASS and CASS can be extended to the case where targets move in a two-dimensional space.

Our third contribution is focused on equilibrium refinement. Our game has multiple equilibria, and the defender strategy found by CASS can be suboptimal with respect to uncertainties in the attacker's model, e.g. if the attacker can only attack during certain time intervals. We present two heuristic equilibrium refinement approaches for this game. The first, *route-adjust*, iteratively computes a defender strategy that dominates earlier strategies. The second, *flow-adjust*, is a linear-programming-based approach. Our experiments show that flow-adjust is computationally faster than route-adjust but route-adjust is more effective in selecting robust equilibrium strategies.

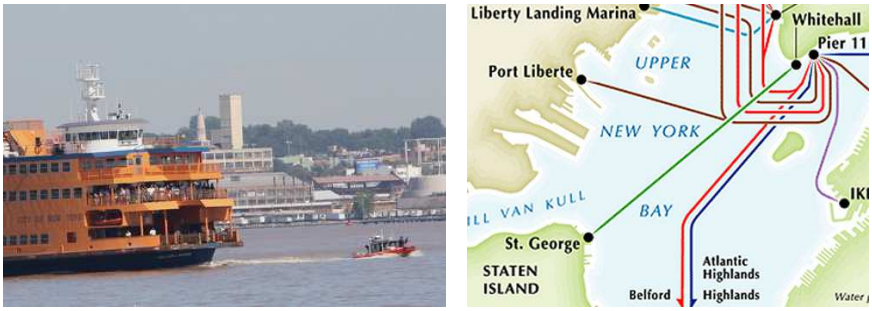
Additionally, I provide several sampling methods for generating practical patrol routes given the defender strategy in a compact representation. Finally, I present detailed experimental analyses of our algorithm in the ferry protection domain. CASS has been deployed by the US Coast Guard since April 2013.

The rest of the chapter is organized as follows: Section 4.1 provides our problem statement. Section 4.2 presents the MRMT_{sg} model and an initial formulation of the DASS and CASS for a one-dimensional setting. Section 4.3 discusses equilibrium refinement, followed by Section 4.4 which gives the generalized formulation of DASS and CASS for two-dimensional settings. Section 4.5 describes how to sample a patrol route and Section 4.6 provides experimental results in the ferry protection domain. Section 4.7provides concluding remarks.

4.1 Problem Statement

One major example of the practical domains motivating this chapter is the problem of protecting ferries that carry passengers in many waterside cities. Packed with hundreds of passengers, these may present attractive targets for an attacker. For example, the attacker may ram a suicide boat packed with explosives into the ferry as happened with attacks on French supertanker Limburg and USS Cole (Greenberg et al., 2006). In this case, the intention of the attacker can only be detected once he gets very close to the ferry. Small, fast and well-armed patrol boats (patrollers) can provide protection to the ferries (Figure 4.1(a)), by detecting the attacker and stopping him with the armed weapons. However, there are often limited numbers of patrol boats, i.e., they cannot protect the ferries at all times at all locations. We first focus on the case where ferries and patrol boats move in a one-dimensional line segment (this is a realistic setting and also simplifies

exposition); we will discuss the two-dimensional case in Section 4.4. Table 6.1 provides a table listing all the notations used in the work.



(a) (b)

Figure 4.1: (a) Protecting ferries with patrol boats; (b) Part of the map of New York Harbor Commuter Ferry Routes. The straight line linking St. George Terminal and Whitehall Terminal indicates a public ferry route run by New York City Department of Transportation.

4.1.1 Domain Description

In this problem, there are L moving targets, $F_1, F_2, ..., F_L$. We assume that these targets move along a one-dimensional domain, specifically, a straight line segment linking two terminal points which we will name A and B. This is sufficient to capture real-world domains such as ferries moving back-and-forth in a straight line between two terminals as they do in many ports around the world; an example is the green line shown in Figure 4.1(b). We will provide an illustration of our geometric formulation of the problem in Figure 4.1.1. The targets have fixed daily schedules. The schedule of each target can be described as a continuous function $S_q : T \rightarrow D$ where $q = 1, ..., L$ is the index of the target, $T = [0, 1]$ is a continuous time interval (e.g., representing the duration of a typical daily patrol shift) and $D = [0, 1]$ is the continuous space of possible locations (normalized) with 0 corresponding to terminal A and 1 to terminal B. Thus $S_q(t)$ denotes the position of the target F_q at a specified time t. We assume S_q is piecewise linear.

The defender has W mobile patrollers that can move along D to protect the targets, denoted as $P_1, P_2, ..., P_W$. Although capable of moving faster than the targets, they have a maximum speed of v_m. While the defender attempts to protect the targets, the attacker will choose a certain time and a certain target to attack. (In the rest of the chapter, we denote the defender as "she" and the attacker as "he"). The probability of attack success depends on the positions of the patrollers at that time. Specifically, each patroller can detect and try to intercept anything within the *protection radius* r_e but cannot detect the attacker prior to that radius. Thus, a patroller protects all targets within her protective circle of radius r_e (centered at her current position), as shown in Figure 4.1.1.

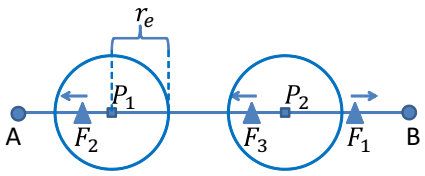


Figure 4.2: An example with three targets (triangles) and two patrollers (squares). The protective circles of the patrollers are shown with protection radius r_e. A patroller protects all targets in her protective circle. Patroller P_1 is protecting F_2 and P_2 is protecting F_3.

Symmetrically, a target is protected by all patrollers whose protective circles can cover it. If the attacker attacks a protected target, then the probability of successful attack is a decreasing function of the number of patrollers that are protecting the target. Formally, we use a set of coefficients $\{C_G\}$ to describe the strength of the protection.

Definition 1. *Let $G \in \{1, ..., W\}$ be the total number of patrollers protecting a target F_q, i.e., there are G patrollers such that F_q is within radius r_e of each of the G patrollers. Then $C_G \in [0, 1]$ specifies the probability that the patrollers can successfully stop the attacker. We require that $C_{G_1} \leq C_{G_2}$ if $G_1 \leq G_2$, i.e., more patrollers offer better protection.*

As with previous work in security games (Tambe, 2011; Yin & Tambe, 2012; Kiekintveld et al., 2009a), we model the game as a Stackelberg game, where the defender commits to a randomized strategy first, and then the attacker can respond to such a strategy. The patrol schedules in these domains were previously created by hand; and hence suffer the drawbacks of hand-drawn patrols, including lack of randomness (in particular, informed randomness) and reliance on simple patrol patterns (Tambe, 2011), which we remedy in this chapter.

4.1.2 Defender Strategy

A pure strategy of the defender is to designate a movement schedule for each patroller. Analogous to the target's schedule, a patroller's schedule can be written as a continuous function $R_u : T \rightarrow D$ where $u = 1, ..., W$ is the index the patroller. R_u must be compatible with the patroller's velocity range. A mixed defender strategy is a randomization over the pure strategies, denoted as f.

4.1.3 Attacker Strategy

The attacker conducts surveillance of the defender's mixed strategy and the targets' schedules; he may then execute a pure strategy response to attack a certain target at a certain time. The attacker's pure strategy can be denoted as $\langle q, t \rangle$ where q is the index of the target to attack and t is the time to attack.

4.1.4 Utility Function

We assume the game is zero-sum. If the attacker performs a successful attack on target F_q at location x at time t, he gets a positive reward $U_q(x, t)$ and the defender gets $-U_q(x, t)$, otherwise

both players get utility zero. The positive reward $U_q(x,t)$ is a known function which accounts for many factors in practice. For example, an attacker may be more effective in his attack when the target is stationary (such as at a terminal point) than when the target is in motion. As the target's position is decided by the schedule, the utility function can be written as $U_q(t) \equiv U_q(S_q(t),t)$. We assume that for each target F_q, $U_q(t)$ can be represented as a piecewise linear function of t.

4.1.5 Equilibrium

Since our game is zero-sum, the Strong Stackelberg Equilibrium can be calculated by finding the minimax/maximin strategy (Fudenberg & Tirole, 1991; Korzhyk et al., 2010a). That is, we can find the optimal defender strategy by finding a strategy that minimizes the maximum of attacker's expected utility.

Definition 2. *For single patroller case, the attacker expected utility of attacking target F_q at time t given defender mixed strategy f is*

$$\mathit{AttEU}_f(F_q,t) = (1 - C_1\omega_f(F_q,t))U_q(t) \tag{4.1}$$

$U_q(t)$ is the reward for a successful attack, $\omega_f(F_q,t)$ is the probability that the patroller is protecting target F_q at time t and C_1 is the protection coefficient of single patroller. We drop the subscript if f is obvious from the context. As C_1 and $U_q(t)$ are constants for a given attacker's pure strategy $\langle q,t\rangle$, AttEU(F_q,t) is purely decided by $\omega(F_q,t)$. The definition with multiple patrollers will be given in Section 4.2.5. We further denote the attacker's maximum expected utility as

$$\text{AttEU}_f^m = \max_{q,t} \text{AttEU}_f(F_q,t) \tag{4.2}$$

So the optimal defender strategy is a strategy f such that the AttEU_f^m is minimized, formally

$$f \in \arg\min_{f'} \text{AttEU}_{f'}^m \tag{4.3}$$

4.1.6 Assumptions

In our problem, the following assumptions are made based on discussions with domain experts. Here we provide our justifications for these assumptions. While appropriate for the current domain of application, relaxing these assumptions for future applications remains an issue for future work, and we provide an initial discussion at the end of this chapter.

- *The attacker's plan is decided off-line, i.e., the attacker does not take into account the patroller's current partial route (partial pure strategy) in executing an attack:* This assumption is similar to the assumption made in other applications of security games and justified elsewhere (An, Kempe, Kiekintveld, Shieh, Singh, Tambe, & Vorobeychik, 2012; Pita, Jain, Ordonez, Portway, Tambe, Western, Paruchuri, & Kraus, 2009; Tambe, 2011). One key consideration is that given that attackers have limited resources as well, for them to generate and execute complex conditional plans that change based on "on-line" observations of defender's pure strategy is both difficult and risky.

- *A single attacker is assumed instead of multiple attackers:* This assumption arises because performing even a single attack is already costly for the attacker. Thus, having coordinating attackers at the same time will be even harder and therefore significantly less likely for the attacker.

- *The game is assumed to be zero-sum:* In this case, the objectives of the defender and attacker are in direct conflict: preventing an attack with higher potential damage is a bigger success to the defender in our game.

- *The schedules for the targets are deterministic:* For the domains we focus on, potential delays in the targets' schedules are usually within several minutes if any, and the targets will try to catch up with the fixed schedules as soon as possible. Therefore, even when delays occur, the deterministic schedule for a target can be viewed as a good approximation of the actual schedule.

4.2 Models

In this section, we introduce our MRMT_{sg} model that uses a discretized strategy space for the defender and a continuous strategy space for the attacker. For clarity of exposition, we then introduce the DASS approach to compute a minimax solution for discretized attacker strategy space (Section 4.2.2), followed by CASS for the attacker's continuous strategy space (Section 4.2.3). We first assume a single patroller in Sections 4.2.1 through 4.2.3 and then generalize to multiple patrollers in Section 4.2.5.

4.2.1 Representing Defender's Strategies

In this subsection, we introduce the discretized defender strategy space and the compact representation used to represent the defender's mixed strategy. We show that the compact representation is equivalent to the intuitive full representation, followed by several properties of the compact representation.

Since the defender's strategy space is discretized, we assume that each patroller only makes changes at a finite set of time points $T = \{t_1, t_2, ..., t_M\}$, evenly spaced across the original continuous time interval. $t_1 = 0$ is the starting time and $t_M = 1$ is the normalized ending time. We denote by δ_t the distance between two adjacent time points: $\delta_t = t_{k+1} - t_k = \frac{1}{M-1}$. We set δ_t to be small enough such that for each target F_q, the schedule $S_q(t)$ and the utility function $U_q(t)$ are linear in each interval $[t_k, t_{k+1}]$ for $k = 1, \ldots, M-1$, i.e., the target is moving with uniform speed and the utility of a successful attack on it changes linearly during each of these intervals. Thus, if t_0 is a breakpoint of $S_q(t)$ or $U_q(t)$ for any q, it can be represented as $t_0 = \delta_t K_0$ where K_0 is an integer.

In addition to discretization in time, we also discretize the line segment AB that the targets move along into a set of points $D = \{d_1, d_2, ..., d_N\}$ and restrict each patroller to be located at one of the discretized points d_i at any discretized time point t_k. Note that D is not necessarily evenly distributed and the targets' locations are not restricted at any t_k. During each time interval $[t_k, t_{k+1}]$, each patroller moves with constant speed from her location d_i at time t_k to her location d_j at time t_{k+1}. Only movements compatible with the speed limit v_m are possible. The points $d_1, d_2, ..., d_N$ are ordered by their distance to terminal A, and d_1 refers to A and d_N refers to B. Since the time interval is discretized into M points, a patroller's route R_u can be represented as a vector $R_u = (d_{r_u(1)}, d_{r_u(2)}, ..., d_{r_u(M)})$. $r_u(k)$ indicates the index of the discretized distance point where the patroller is located at time t_k.

We discretized the defender's strategy space not only for computational reasons. It is not even clear whether an equilibrium exists in the original game with continuous strategy space for both

players. The discretization is made also because of the practical constraint of patrollers. In addition, as we show later in this chapter, we can provide a bi-criteria polynomial time approximation scheme for the optimal defender strategy in the original game using the discretized games.

For expository purpose, we first focus on the case with a single defender resource and then generalize to a larger number of resources later. For a single defender resource, the defender's mixed strategy in *full representation* assigns a probability to each of the patrol routes that can be executed. Since at each time step a patroller can choose to go to at most N different locations, there are at most N^M possible patrol routes in total and this number is achievable if there is no speed limit (or v_m is large enough). The exponentially growing number of routes will make any analysis based on full representation intractable.

Therefore, we use the *compact representation* of the defender's mixed strategy.

Definition 3. *The* compact representation *for a single defender resource is a compact way to represent the defender's mixed strategy using* flow distribution variables $\{f(i,j,k)\}$. *$f(i,j,k)$ is the probability of the patroller moving from d_i at time t_k to d_j at time t_{k+1}.*

The complexity of the compact representation is $O(MN^2)$, which is much more efficient compared to the full representation.

Proposition 1. *Any strategy in full representation can be mapped into a compact representation.*

Proof sketch: If there are H possible patrol routes $R_1, R_2, ..., R_H$, a mixed defender strategy can be represented in full representation as a probability vector $(p(R_1), ...p(R_H))$ where $p(R_u)$ is the probability of taking route R_u. Taking route R_u means the patroller moves from $d_{r_u(k)}$ to $d_{r_u(k+1)}$ during time $[t_k, t_{k+1}]$, so the edge $E_{R_u(k),R_u(k+1),k}$ is taken when route R_u is chosen.

Then the total probability of taking edge $E_{i,j,k}$ is the sum of probabilities of all the routes R_u where $R_u(k) = i$ and $R_u(k+1) = j$. Therefore, given any strategy in full presentation specified by the probability vector $(p(R_1), ...p(R_H))$, we can construct a compact representation consisting of a set of flow distribution variables $\{f(i,j,k)\}$ where

$$f(i,j,k) = \sum_{R_u : R_u(k)=i \text{ and } R_u(k+1)=j} p(R_u). \tag{4.4}$$

□

Figure 4.3 shows a simple example illustrating the compact representation. Numbers on the edges indicate the value of $f(i,j,k)$. We use $E_{i,j,k}$ to denote the directed edge linking nodes (t_k, d_i) and (t_{k+1}, d_j). For example, $f(2,1,1)$, the probability of the patroller moving from d_2 to d_1 during time t_1 to t_2, is shown on the edge $E_{2,1,1}$ from node (t_1, d_2) to node (t_2, d_1). While a similar compact representation was used earlier by (Yin & Tambe, 2012), we use it in a continuous setting.

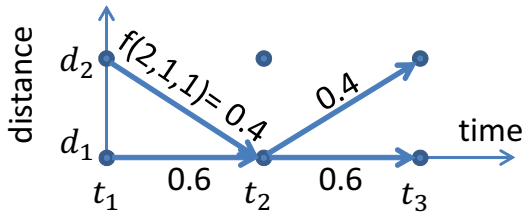


Figure 4.3: Compact representation: x-axis shows time intervals; y-axis the discretized distance-points in the one-dimensional movement space.

Note that different mixed strategies in full representation can be mapped to the same compact representation. Table 4.2 shows two different mixed defender strategies in full representations that can be mapped to the same mixed strategy in compact representation as shown in Figure 4.3.

The probability of a route is labeled on all edges in the route in full representation. Adding up the numbers of a particular edge $E_{i,j,k}$ in all routes of a full representation together, we can get $f(i,j,k)$ for the compact representation.

Theorem 1. *Compact representation does not lead to any loss in solution quality.*

Proof sketch: The complete proof of the theorem relies on the calculations in Section 4.2.2 and 4.2.3. Here we provide a sketch. Recall our goal is to find an optimal defender strategy f that minimizes the maximum attacker expected utility AttEU_f^m. As we will show in the next subsections, $\omega(F_q, t)$ can be calculated from the compact representation $\{f(i,j,k)\}$. If two defender strategies under the full representation are mapped to the same compact representation $\{f(i,j,k)\}$, they will have the same ω function and then the same AttEU function according to Equation 4.1. Thus the value of AttEU_f^m is the same for the two defender strategies. So an optimal mixed defender strategy in compact representation is still optimal for the corresponding defender strategies in full representation.

We exploit the following properties of the compact representation.

Property 1. *For any time interval $[t_k, t_{k+1}]$, the sum of all flow distribution variables equals to 1: $\sum_{i=1}^{N}\sum_{j=1}^{N} f(i,j,k) = 1$.*

Property 2. *The sum of flows that go into a particular node equals the sum of flows that go out of the node. Denote the sum for node (t_k, d_i) by $p(i,k)$, then $p(i,k) = \sum_{j=1}^{N} f(j,i,k-1) = \sum_{j=1}^{N} f(i,j,k)$. Each $p(i,k)$ is equal to the marginal probability that the patroller is at location d_i at time t_k.*

Property 3. *Combining Property 1 and 2, $\sum_{i=1}^{N} p(i,k) = 1$.*

4.2.2 DASS: Discretized Attacker Strategies

In this subsection, we introduce DASS, a mathematical program that efficiently finds minimax solutions for MRMT_{sg}-based games under the assumption that the attacker will attack at one of the discretized time points t_k. In this problem, we need to minimize v where v is the maximum of attacker's expected utility. Here, v is the maximum of AttEU(F_q, t) for any target F_q at any discretized time point t_k.

From Equation (4.1), we know that AttEU(F_q, t) is decided by $\omega(F_q, t)$, the probability that the patroller is protecting target F_q at time t. Given the position of the target $S_q(t)$, we define the protection range $\beta_q(t) = [\max\{S_q(t) - r_e, d_1\}, \min\{S_q(t) + r_e, d_N\}]$. If the patroller is located within the range $\beta_q(t)$, the distance between the target and the patroller is no more than r_e and thus the patroller is protecting F_q at time t. So $\omega(F_q, t)$ is the probability that the patroller is located within range $\beta_q(t)$ at time t. For the discretized time points t_k, the patroller can only be located at a discretized distance point d_i, so we define the following.

Definition 4. *$I(i, q, k)$ is a function of two values. $I(i, q, k) = 1$ if $d_i \in \beta_q(t_k)$, and otherwise $I(i, q, k) = 0$.*

In other words, $I(i, q, k) = 1$ means that a patroller located at d_i at time t_k can protect target F_q. Note that the value of $I(i, q, k)$ can be calculated directly from the input parameters (d_i, $S_q(t)$ and r_e) and stored in a look-up table. In particular, $I(i, q, k)$ is not a variable in the formulations that follow. It simply encodes the relationship between d_i and the location of target F_q at t_k. The probability that the patroller is at d_i at time t_k is $p(i, k)$. So we have

$$\omega(F_q, t_k) = \sum\nolimits_{i:I(i,q,k)=1} p(i, k), \tag{4.5}$$

$$\text{AttEU}(F_q, t_k) = \left(1 - C_1 \sum\nolimits_{i:I(i,q,k)=1} p(i,k)\right) U_q(t_k). \tag{4.6}$$

Equation (4.6) follows from Equations (4.1) and (4.5), expressing attacker's expected utility for discretized time points. Finally, we must address speed restrictions on the patroller. We set all flows corresponding to actions that are not achievable to zero,[1] that is, $f(i,j,k) = 0$ if $|d_j - d_i| > v_m \delta_t$. Thus, DASS can be formulated as a linear program. This linear program solves for any number of targets but only one defender resource.

$$\min_{f(i,j,k),p(i,k)} z \tag{4.7}$$

$$f(i,j,k) \in [0,1], \qquad \forall i,j,k \tag{4.8}$$

$$f(i,j,k) = 0, \qquad \forall i,j,k \text{ such that } |d_j - d_i| > v_m \delta_t \tag{4.9}$$

$$p(i,k) = \sum_{j=1}^{N} f(j,i,k-1), \qquad \forall i, \forall k > 1 \tag{4.10}$$

$$p(i,k) = \sum_{j=1}^{N} f(i,j,k), \qquad \forall i, \forall k < M \tag{4.11}$$

$$\sum_{i=1}^{N} p(i,k) = 1, \qquad \forall k \tag{4.12}$$

$$z \geq \text{AttEU}(F_q, t_k), \qquad \forall q, \forall k \tag{4.13}$$

Constraint 4.8 describes the probability range. Constraint 4.9 describes the speed limit. Constraints 4.10–4.11 describes Property 2. Constraint 4.12 is exactly Property 3. Property 1 can be

[1] Besides the speed limit, we can also model other practical restrictions of the domain by placing constraints on $f(i,j,k)$.

derived from Property 2 and 3, so it is not listed as a constraint. Constraint (4.13) shows the attacker chooses the strategy that gives him the maximal expected utility among all possible attacks at discretized time points; where AttEU$(\cdot)$ is described by Equation (4.6).

4.2.3 CASS: Continuous Attacker Strategies

In this subsection, we generalize the problem to one with continuous attacker strategy set and provides a linear-programming-based solution CASS. CASS efficiently finds optimal mixed defender strategy under the assumption that the attacker can attack at any time in the continuous time interval $T = [0, 1]$. With this assumption, DASS's solution quality guarantee may fail: if the attacker chooses to attack between t_k and t_{k+1}, he may get a higher expected reward than attacking at t_k or t_{k+1}. Consider the following example, with the defender's compact strategy between t_k and t_{k+1} shown in Figure 4.4. Here the defender's strategy has only three non-zero flow variables $f(3, 4, k) = 0.3$, $f(3, 1, k) = 0.2$, and $f(1, 3, k) = 0.5$, indicated by the set of three edges $E^+ = \{E_{3,4,k}, E_{3,1,k}, E_{1,3,k}\}$. A target F_q moves from d_3 to d_2 at constant speed during $[t_k, t_{k+1}]$. Its schedule is depicted by the straight line segment S_q. The dark lines L_q^1 and L_q^2 are parallel to S_q with distance r_e. The area between them indicates the protection range $\beta_q(t)$ for any time $t \in (t_k, t_{k+1})$. Consider the time points at which an edge from E^+ intersects one of L_q^1, L_q^2, and label them as $\theta_{qk}^r, r = 1 \dots 4$ in Figure 4.4). Intuitively, these are all the time points at which a defender patrol could potentially enter or leave the protection range of the target. To simplify the notation, we denote t_k as θ_{qk}^0 and t_{k+1} as θ_{qk}^5. For example, a patroller moving from d_3 to d_4 (or equivalently, taking the edge $E_{3,4,k}$) protects the target from θ_{qk}^0 to θ_{qk}^1 because $E_{3,4,k}$ is between L_1^1 and L_1^2 in $[\theta_{qk}^0, \theta_{qk}^1]$, during which the distance to the target is less or equal than protection radius r_e. Consider the sub-intervals between each θ_{qk}^r and θ_{qk}^{r+1}, for $r = 0 \dots 4$.

Since within each of these five sub-intervals, no patroller enters or leaves the protection range, the probability that the target is being protected is a constant in each sub-interval, as shown in Figure 4.5(a).

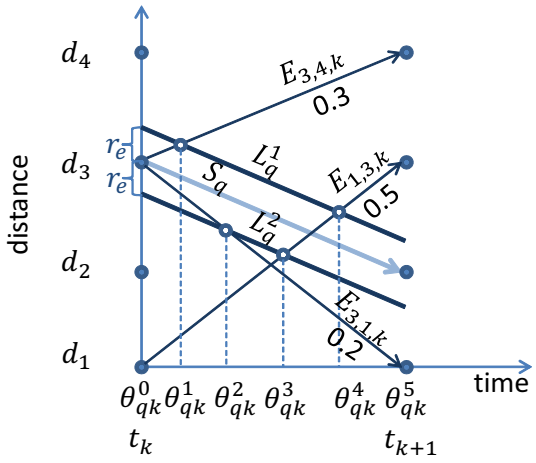


Figure 4.4: An example to show how a target moving from d_3 to d_2 during $[t_k, t_{k+1}]$ is protected. In a sub-interval $[\theta_{qk}^r, \theta_{qk}^{r+1}]$, a patroller either always protects the target or never protects the target. Equivalently, the target is either always within the protective circle of a patroller or always outside the circle.

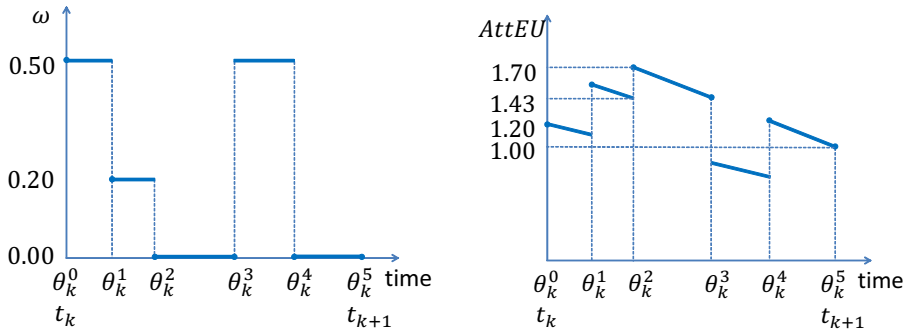
Suppose $U_q(t)$ decreases linearly from 2 to 1 during $[t_k, t_{k+1}]$ and $C_1 = 0.8$. We can then calculate the attacker's expected utility function AttEU(F_q, t) for (t_k, t_{k+1}), as plotted in Figure 4.5(b). AttEU(F_q, t) is linear in each sub-interval but the function is discontinuous at the intersection points $\theta_{qk}^1, \ldots, \theta_{qk}^4$ because of the patroller leaving or entering the protection range of the target. We denote the limit of $AttEU$ when t approaches θ_{qk}^r from the left as:

$$\lim_{t \to \theta_{qk}^{r-}} \text{AttEU}(F_q, t) = \text{AttEU}(F_q, \theta_{qk}^{r-})$$

Similarly, the right limit is denoted as:

$$\lim_{t \to \theta_{qk}^{r+}} \text{AttEU}(F_q, t) = \text{AttEU}(F_q, \theta_{qk}^{r+})$$

If F_q is the only target, an attacker can choose to attack at a time immediately after θ_{qk}^2, getting an expected utility that is arbitrarily close to 1.70. According to Equation (4.6), we can get $\text{AttEU}(F_q, t_k) = 1.20$ and $\text{AttEU}(F_q, t_{k+1}) = 1.00$, both lower than $\text{AttEU}(F_q, \theta_{qk}^{2+})$. Thus, the attacker can get a higher expected reward by attacking between t_k and t_{k+1}, violating DASS's quality guarantee.



(a) Probability that the target is protected is a constant in each sub-interval.

(b) The attacker's expected utility is linear in each sub-interval.

Figure 4.5: Sub-interval analysis in (t_k, t_{k+1}) for the example shown in Figure 4.4.]

However, because of discontinuities in the attacker's expected utility function, a maximum might not exist. This implies that the minimax solution concept might not be well-defined for our game. We thus define our solution concept to be minimizing the *supremum* of $\text{AttEU}(F_q, t)$.

Definition 5. *The supremum of attacker's expected utility is the smallest real number that is greater than or equal to all elements of the infinite set* $\{\textit{AttEU}(F_q, t)\}$*, denoted as* $\sup \textit{AttEU}(F_q, t)$.

The supremum is the least upper bound of the function $\text{AttEU}(F_q, t)$. So for CASS model, Equation 4.2 should be modified as

$$\text{AttEU}_f^m = \sup_{q,t} \text{AttEU}_f(F_q, t) \tag{4.14}$$

So a defender strategy f is minimax if AttEU_f^m is maximized, i.e.,

$$f \in \arg\min_{f'} \sup \text{AttEU}_{f'}(F_q, t)$$

In the above example, the supremum of attacker's expected utility in (t_k, t_{k+1}) is $\text{AttEU}(F_q, \theta_{qk}^{2+}) = 1.70$. In the rest of the chapter, we will not specify when supremum is used instead of maximum as it can be easily inferred from the context.

How can we deal with the possible attacks between the discretized points and find an optimal defender strategy? We generalize the process above (called sub-interval analysis) to all possible edges $E_{i,j,k}$. We then make use of the piecewise linearity of $\text{AttEU}(F_q, t)$ and the fact that the potential discontinuity points are fixed, which allows us to construct a linear program that solves the problem to **optimality**. We name the approach CASS (Solver for Continuous Attacker Strategies).

We first introduce the general sub-interval analysis. For any target F_q and any time interval (t_k, t_{k+1}), we calculate the time points at which edges $E_{i,j,k}$ and L_q^1, L_q^2 intersect, denoted as intersection points. We sort the intersection points in increasing order, denoted as $\theta_{qk}^r, r = 1 \ldots M_{qk}$, where M_{qk} is the total number of intersection points. Set $\theta_{qk}^0 = t_k$ and $\theta_{qk}^{M_{qk}+1} = t_{k+1}$. Thus (t_k, t_{k+1}) is divided into sub-intervals $(\theta_{qk}^r, \theta_{qk}^{r+1}), r = 0, ..., M_{qk}$.

Lemma 1. *For any given target F_q, AttEU(F_q, t) is piecewise-linear in t. Furthermore, there exists a fixed set of time points, independent of the defender's mixed strategy, such that AttEU(F_q, t) is linear between each adjacent pair of points. Specifically, these points are the intersection points θ^r_{qk} defined above.*

Proof: In each sub-interval $(\theta^r_{qk}, \theta^{r+1}_{qk})$ for a target F_q, a feasible edge $E_{i,j,k}$ is either totally above or below L^1_q, and similarly for L^2_q. Otherwise there will be a new intersection point which contradicts the definition of the sub-intervals. If edge $E_{i,j,k}$ is between L^1_q and L^2_q, the distance between a patroller taking the edge and target F_q is less than r_e, meaning the target is protected by the patroller. As edge $E_{i,j,k}$ is taken with probability $f(i,j,k)$, the total probability that the target is protected ($\omega(F_q, t)$) is the sum of $f(i,j,k)$ whose corresponding edge $E_{i,j,k}$ is between the two lines in a sub-interval. So $\omega(F_q, t)$ is constant in t in each sub-interval and thus the attacker's expected utility AttEU(F_q, t) is linear in each sub-interval according to Equation 2 as $U_q(t)$ is linear in $[t_k, t_{k+1}]$. Discontinuity can only exist at these intersection points and an upper bound on the number of these points for target F_q is MN^2. □

Define coefficient $A^r_{qk}(i,j)$ to be C_1 if edge $E_{i,j,k}$ is between L^1_q and L^2_q in $(\theta^r_{qk}, \theta^{r+1}_{qk})$, and 0 otherwise. According to Equation (4.1) and the fact that $\omega(F_q, t)$ is the sum of $f(i,j,k)$ whose corresponding coefficient $A^r_{qk}(i,j) = C_1$, we have the following equation for $t \in (\theta^r_{qk}, \theta^{r+1}_{qk})$.

$$\text{AttEU}(F_q, t) = \left(1 - \sum_{i=1}^{N}\sum_{j=1}^{N} A^r_{qk}(i,j) f(i,j,k)\right) \cdot U_q(t) \tag{4.15}$$

Piecewise linearity of AttEU(F_q, t) means the function is monotonic in each sub-interval and the supremum can be found at the intersection points. Because of linearity, the supremum of AttEU in $(\theta^r_{qk}, \theta^{r+1}_{qk})$ can only be chosen from the one-sided limits of the endpoints, AttEU(F_q, θ^{r+}_{qk}) and

AttEU($F_q, \theta_{qk}^{(r+1)-}$). Furthermore, if $U_q(t)$ is decreasing in $[t_k, t_{k+1}]$, the supremum is AttEU(F_q, θ_{qk}^{r+}) and otherwise it is AttEU($F_q, \theta_{qk}^{(r+1)-}$). In other words, all other attacker's strategies in $(\theta_{qk}^r, \theta_{qk}^{r+1})$ are dominated by attacking at time close to θ_{qk}^r or θ_{qk}^{r+1}. Thus, CASS adds new constraints to Constraints 4.8–4.13 which consider attacks to occur at $t \in (t_k, t_{k+1})$. We add one constraint for each sub-interval with respect to the possible supremum value in this sub-interval:

$$\min_{f(i,j,k),p(i,k)} z \tag{4.16}$$

$$\text{subject to constraints } (4.8 \dots 4.13)$$

$$z \geq \max\{\text{AttEU}(F_q, \theta_{qk}^{r+}), \text{AttEU}(F_q, \theta_{qk}^{(r+1)-})\} \tag{4.17}$$

$$\forall k \in \{1 \dots M\}, q \in \{1 \dots L\}, r \in \{0 \dots M_{qk}\}$$

This linear program stands at the core of CASS and we will not differentiate the name for the solver and the name for the linear program in the following. All the linear constraints included by Constraint 4.17 can be added to CASS using Algorithm 4.2.3. The input of the algorithm include targets' schedules $\{S_q\}$, the protection radius r_e, the speed limit v_m, the set of discretized time points $\{t_k\}$ and the set of discretized distance points $\{d_i\}$. Function CalInt(L_q^1, L_q^2, v_m) in Line 1 returns the list of all intersection time points between all possible edges $E_{i,j,k}$ and the parallel lines L_q^1, L_q^2, with additional points t_k as θ_{qk}^0 and t_{k+1} as $\theta_{qk}^{M_{qk}+1}$. Function CalCoef($L_q^1, L_q^2, v_m, \theta_{qk}^r, \theta_{qk}^{r+1}$) in Line 1 returns the coefficient matrix A_{qk}^r. A_{qk}^r can be easily decided by checking the status at the midpoint in time. Set $t_{mid} = (\theta_{qk}^r + \theta_{qk}^{r+1})/2$ and denote the patroller's position at t_{mid} when edge $E_{i,j,k}$ is taken as $E_{i,j,t_{mid}}$, thus $A_{qk}^r(i,j) = C_1$ if $E_{i,j,t_{mid}} \in \beta_q(t_{mid})$. Lines 1–1 add a constraint with respect to the larger value of AttEU(F_q, θ_{qk}^{r+}) and AttEU($F_q, \theta_{qk}^{(r+1)-}$) to CASS for this sub-interval $(\theta_{qk}^r, \theta_{qk}^{r+1})$. It means

when the attacker chooses to attack F_q in this sub-interval, his best choice is decided by the larger value of the two side-limits of AttEU in $(\theta_{qk}^r, \theta_{qk}^{r+1})$.

Algorithm 1 Add constraints described in Constraint 4.17

Input: S_q, r_e, v_m, $\{t_k\}$, $\{d_i\}$ **for** $k \leftarrow 1, \ldots, M-1$ **do**
  for $q \leftarrow 1, \ldots, L$ **do**
    $L_q^1 \leftarrow S_q + r_e$, $L_q^2 \leftarrow S_q - r_e$ $\theta_{qk}^0, \ldots, \theta_{qk}^{M_{qk}+1} \leftarrow$ CalInt(L_q^1, L_q^2, v_m) **for** $r \leftarrow 0, \ldots, M_{qk}$ **do**
      $A_{qk}^r \leftarrow$ CalCoef$(L_q^1, L_q^2, v_m, \theta_{qk}^r, \theta_{qk}^{r+1})$ **if** $U_q(t)$ *is decreasing in* $[t_k, t_{k+1}]$ **then**
        add constraint $z \geq$ AttEU(F_q, θ_{qk}^{r+})
      end
      else
        add constraint $z \geq$ AttEU$(F_q, \theta_{qk}^{(r+1)-})$
      end
    end
  end
end

Theorem 2. *CASS computes (in polynomial time) the exact solution (minimax) of the game with discretized defender strategies and continuous attacker strategies.*

Proof: According to Lemma 1, AttEU(F_q, t) is piecewise linear and discontinuity can only occur at the intersection points θ_{qk}^r. These intersection points divide the time space into sub-intervals. Because of piecewise linearity, the supremum of AttEU(F_q, t) equals to the limit of an endpoint of at least one sub-interval. For any defender's strategy f that is feasible, a feasible z of the linear program 4.16-4.17 is no less than any of the limit values at the intersection points according to Constraint 4.17 and values at the discretized time points t_k according to Constraint 4.13, and thus v can be any upper bound of AttEU(F_q, t) for f. As z is minimized in the objective function, z is no greater than the supremum of AttEU(F_q, t) given any defender strategy f, and further z will be the minimum of the set of supremum corresponding to all defender strategies. Thus we get the optimal defender strategy f.

The total number of variables in the linear program is $O(MN^2)$. The number of constraints represented in Algorithm 4.2.3 is $O(MN^2L)$ as the number of intersection points is at most $2(M-1)N^2$ for each target. The number of constraints represented in Constraints 4.8–4.13 is $O(MN^2)$. Thus, the linear program computes the solution in polynomial time. □

Corollary 1. *The solution of CASS provides a feasible defender strategy of the original continuous game and gives exact expected value of that strategy.*

4.2.4 Optimal Defender Strategy in the Original Game

We have solved the problem with discretized defender strategy space. In this subsection, we discuss how much the defender loses by using the discretization. Indeed, we can provide a bi-criteria polynomial time approximation scheme for the optimal defender strategy in the original game using the discretized games. Let $\text{DefEU}_{\delta_d,\delta_t}(r_e, v_m)$ denote the defender's optimal expected utility with a discretization granularity defined by δ_d and δ_t (i.e., $N = \frac{1}{\delta_d}$). Let $\text{DefEU}_{opt}(r_e, v_m)$ denote the defender's optimal expected utility without using discretization. The defender's capability is depicted by the protection radius r_e and speed limit v_m. Then $\text{DefEU}_{opt}(r_e, v_m)$ is bounded by the following:

$$\text{DefEU}_{\delta_d,\delta_t}(r_e, v_m) \leq \text{DefEU}_{opt}(r_e, v_m) \leq \text{DefEU}_{\delta'_d,\delta'_t}(r_e + \epsilon, v_m + \Delta) \tag{4.18}$$

$$\forall \delta_d, \delta_t, \epsilon, \Delta, \delta'_t \leq \frac{2\epsilon}{v_m}, \delta'_d \leq \delta'_t \Delta \tag{4.19}$$

4.2.5 Generalized Model with Multiple Defender Resources

In this subsection, we generalize DASS and CASS to solve the problem with multiple defender resources. When there are multiple patrollers, the patrollers will coordinate with each other. Recall the protection coefficient C_G in Definition 1, a target is better protected when more patrollers are close to it (within radius r_e). So the protection provided to a target is determined when all patrollers' locations are known. Thus it is not sufficient to calculate the probability that an individual edge is taken as in the single patroller case. Under the presence of multiple patrollers, we need a more complex representation to explicitly describe the defender strategy. To illustrate generalization to the multiple defender resources case, we first take two patrollers as an example. If there are two patrollers, the patrol strategy can be represented using flow distribution variables $\{f(i_1, j_1, i_2, j_2, k)\}$. Here the flow distribution variables are defined on the Cartesian product of two duplicated sets of all feasible edges $\{E_{i,j,k}\}$. $f(i_1, j_1, i_2, j_2, k)$ is the joint probability of the first patroller moving from d_{i_1} to d_{j_1} and the second patroller moving from d_{i_2} to d_{i_2} during time t_k to t_{k+1}, i.e., taking edge $E_{i_1,j_1,k}$ and $E_{i_2,j_2,k}$ respectively. The corresponding marginal distribution variable $p(i_1, i_2, k)$ represents for the probability that the first patroller is at d_{i_1} and the second at d_{i_2} at time t_k. Protection coefficients C_1 and C_2 are used when one or two patrollers are protecting the target respectively.

So the attacker's expected utility can be written as

$$\text{AttEU}(F_q, t) = (1 - (C_1 \cdot \omega_1(F_q, t) + C_2 \cdot \omega_2(F_q, t))) \cdot U_q(t)$$

$\omega_1(F_q, t)$ is the probability that only one patroller is protecting the target F_q at time t and $\omega_2(F_q, t)$ is the probability that both patrollers are protecting the target. For attacks that happen at discretized points t_k, we can make use of $I(i, q, k)$ in Definition 4 and $I(i_1, q, k) + I(i_2, q, k)$ is the total number of patrollers protecting the ferry at time t_k.

$$\omega_1(F_q, t_k) = \sum\nolimits_{i_1, i_2 : I(i_1,q,k)+I(i_2,q,k)=1} p(i_1, i_2, k)$$
$$\omega_2(F_q, t_k) = \sum\nolimits_{i_1, i_2 : I(i_1,q,k)+I(i_2,q,k)=2} p(i_1, i_2, k)$$

Constraints for attacks occurring in (t_k, t_{k+1}) can be calculated with an algorithm that looks the same as Algorithm 4.2.3. The main difference is in the coefficient matrix A^r_{qk} and the expression of AttEU. We set the values in the coefficient matrix $A^r_{qk}(i_1, j_1, i_2, j_2)$ as C_2 if both edges $E_{i_1,j_1,k}$ and $E_{i_2,j_2,k}$ are between L^1_q and L^2_q, and C_1 if only one of the edges protects the target. The attacker's expected utility function in $(\theta^r_{qk}, \theta^{r+1}_{qk})$ is

$$\text{AttEU}(F_q, t) = (1 - \sum_{i_1, j_1, i_2, j_2} A^r_{qk}(i_1, j_1, i_2, j_2) f(i_1, j_1, i_2, j_2, k)) \cdot U_q(t)$$

For a general case of W defender resources, we can use $\{f(i_1, j_1, ..., i_W, j_W, k)\}$ to represent the patrol strategy.

Definition 6. *The* compact representation *for multiple defender resources is a compact way to represent the defender's mixed strategy using* flow distribution variables $\{f(i_1, j_1, ..., i_W, j_W, k)\}$. $\{f(i_1, j_1, ..., i_W, j_W, k)\}$ *is the joint probability that patroller moving from d_{i_u} at time t_k to d_{j_u} at time t_{k+1} for $u = 1 \dots W$.*

Given the generalized compact representation, we get the following equations for calculating the attacker's expected utility function and the protection probability:

$$\text{AttEU}(F_q, t) = \left(1 - \sum_{Q=1}^{W} C_Q \cdot \omega_Q(F_q, t)\right) \cdot U_q(t)$$

$$\omega_Q(F_q, t_k) = \sum\nolimits_{i_1,\ldots,i_W:\ \sum_{u=1}^{W} I(i_u,q,k)=Q} p(i_1, \ldots, i_W, k)$$

Q is the number of patrollers protecting the target. We can modify Algorithm 4.2.3 to apply for the multiple defender resource case. Set $A^r_{qk}(i_1, j_1, ..., i_W, j_W)$ as C_Q if Q of the edges $\{E_{i_u,j_u,k}\}$ are between L^1_q and L^2_q.

We conclude the linear program for generalized CASS for multiple patrollers as follows.

$$\min_{f(i_1,j_1,\ldots,i_W,j_W,k),p(i_1,\ldots,i_W,k)} z \tag{4.20}$$

$$f(i_1, j_1, \ldots, i_W, j_W, k) = 0, \forall i_1, \ldots, i_W, j_1, \ldots, j_W \text{ such that } \exists u, |d_{j_u} - d_{i_u}| > v_m \delta_t \tag{4.21}$$

$$p(i_1, \ldots, i_W, k) = \sum_{j_1=1}^{n} \ldots \sum_{j_W=1}^{n} f(j_1, i_1, \ldots, j_W, i_W, k-1), \forall i_1, \ldots, i_W, \forall k > 1 \tag{4.22}$$

$$p(i_1, \ldots, i_W, k) = \sum_{j_1=1}^{n} \ldots \sum_{j_W=1}^{n} f(i_1, j_1, \ldots, i_W, j_W, k), \forall i_1, \ldots, i_W, \forall k < M \tag{4.23}$$

$$\sum_{i_1=1}^{n} \ldots \sum_{i_W=1}^{n} p(i_1, \ldots, i_W, k) = 1, \forall k \tag{4.24}$$

$$z \geq \text{AttEU}(F_q, t_k), \forall q, \forall k \tag{4.25}$$

$$z \geq \max\{\text{AttEU}(F_q, \theta_{qk}^{r+}), \text{AttEU}(F_q, \theta_{qk}^{(r+1)-})\}, \forall k, \forall q, \forall r \tag{4.26}$$

The number of variables in the linear program is $O(MN^{2W})$ and the number of constraints is $O(MN^W)$. It is reasonable to examine potentially more efficient alternatives. We summarize the results of such an examination below concluding that using the current linear program would appear to currently offer our best tradeoff in terms of solution quality and time at least for the current domains of application; although as discussed below, significant future work might reveal alternatives approaches for other future domains.

The first question to examine is that of the computational complexity of the problem at hand: generating optimal patrolling strategies for multiple patrollers on a graph. Unfortunately, despite the significant attention paid to the topic, currently, the complexity remains unknown. More specifically, the question of computational complexity of generating patrols for multiple defenders on graphs of different types has received significant attention (Letchford, 2013; Korzhyk et al., 2010a). These studies illustrate that in several cases the problem is NP-hard, in some cases the problem is known to be polynomial time, but despite significant effort, the problem complexity in many cases remains unknown (Letchford & Conitzer, 2013). Unfortunately, our graph turns out to be different from the cases considered in their work. Indeed, the DASS model can be explained as a game with homogeneous defender resources patrolling on a graph, similar to the cases that have already been considered. However, prior results cannot explain the complexity of our game as the structure of our graph does not fit any of the prior graphs.

Given that computational complexity results are not directly available, we may examine approaches to provide efficient approximations. Here we provide an overview of two such approaches (providing experimental results in Section 4.6.1.6). Our first approach attempts to provide a more compact representation in the hope of providing a speedup. To that end, we apply an intuitive approach that uses individual strategy profile for each patroller and then calculates a best possible mixed strategy combination. Unfortunately, this approach is inefficient in run-time even for the DASS model and may result in a suboptimal solution. Thus, although more compact, this approach fails to achieve our goal; we explain this approach next.

Assume each patroller independently follows her mixed strategy. Denote the individual mixed strategy for patroller u as $f_u(i_u, j_u, t_k)$, and the probability that a target is protected by Q players

can be represented as a polynomial expression of $\{f_u(i_u, j_u, t_k)\}$ of order Q. Then our optimization problem is converted to minimizing objective function z with non-linear constraints. Assume we have two patrollers, and for a potential attack at target q at time t_k, we denote the probability that patroller u is protecting the target as ϖ_u. ϖ_u is linear in f_u and the attacker's expected utility for this attack can be represented as

$$\text{AttEU}(F_q, t_k) = (1 - C_1((1 - \varpi_1)\varpi_2 + (1 - \varpi_2)\varpi_1) - C_2\varpi_1\varpi_2)U_q(t_k)$$

So a constraint $z \geq \text{AttEU}(F_q, t_k)$ is quadratic in f, due to the fact that the joint probability is represented by the product of the individual probability of each patroller. These constraints are not ensured to have a convex feasible region, and there are no known polynomial algorithms for solving this kind of non-convex optimization problems. We attempt to solve the problem by converting it into a mathematical program with a non-convex objective function and linear constraints, i.e., instead of minimizing z with constraints $z \geq \text{AttEU}(F_q, t_k)$, we incorporate the constraints into the objective function as

$$z = \max_{q,k}\{\text{AttEU}(F_q, t_k)\} \tag{4.27}$$

The results in Section 4.6.1.6 show that when we solve this mathematical program in MATLAB using function **fmincon** with interior-point method for the DASS model, the algorithm fails to get to a feasible solution efficiently and even when enough time is given, the solution can still be suboptimal as it may get stuck at a local minimum. To conclude, although this approach is

more compact and helps in saving memory, it is inefficient at run-time and may result in a loss in solution quality.

Our second approach takes a further step to reduce the run-time complexity, making it a polynomial approximation algorithm, but it can lead to a high degradation in solution quality. In this approach, we iteratively compute the optimal defender strategy for a newly added resource unit given the existing strategies for the previous defender resources. Namely, we first calculate $f_1(i_1, j_1, t_k)$ as if only one patroller is available and then calculate $f_2(i_2, j_2, t_k)$ given the value of $f_1(i_1, j_1, t_k)$. In this way, we need to solve W linear programs with complexity $O(MN^2)$ so this approach is much faster compared to the former one. Unfortunately, this approach fails to capture the coordination between the patrollers effectively and thus may result in a high degradation in solution quality. For example, suppose there are only two targets of constant utility U, one target stays at terminal A and the other one stays at terminal B. Further, suppose the protection coefficient is always 1 when a target is protected by one or more patrollers. When two patrollers are available, the optimal solution would be each protect one of the targets all the way, so both targets are protected with probability 1 and the expected utility function for the attacker is 0. If the defender strategy is calculated for each patroller sequentially as discussed above, the solution would be to protect each target with probability 0.5 for both players, making the attacker's expected utility $0.25\%U$. In other words, we reach a suboptimal solution, wasting resources when both patrollers end up protecting the same target with probability 0.25. In this case, we can already see that there is a 0.25 probability that a target is unprotected when clearly an optimal solution existed that protected all targets with probability 1. Thus, even with just two patrollers, this solution leads to a potentially significant loss in expected utility; therefore, this solution clearly appears to be inadequate for our purposes.

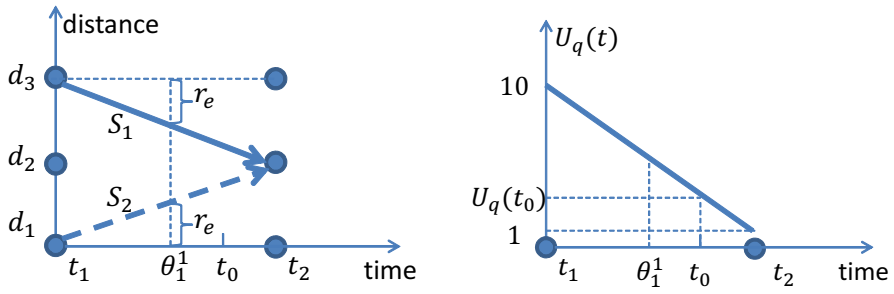
Given the above discussion, it would appear that a fast approximation may lead to significant losses in solution quality or may not be efficient enough. Fortunately for current application domains, such as the current deployment of CASS for protecting ferries (e.g., the Staten Island Ferry in New York), the number of defender resources are limited. The lack of resources is the main reason that optimization using security games becomes critical. As a result, our current approach of CASS is adequate for current domains such as ferry protection. Further research about scale-up is an issue for future work.

4.3 Equilibrium Refinement

A game often has multiple equilibria. Since our game is zero-sum, all equilibria achieve the same objective value. However, if an attacker deviates from his best response, some equilibrium strategies for the defender may provide better results than others.

Consider the following example game. There are two targets moving during $[t_1, t_2]$ (no further discretization): one moves from d_3 to d_2 and the other moves from d_1 to d_2 (See Figure 4.6(a)). Suppose $d_3 - d_2 = d_2 - d_1 = \delta_d$ and $r_e = 0.5\delta_d$. There is only one patroller available and the protection coefficient $C_1 = 1$. Both targets' utility functions decrease from 10 to 1 in $[t_1, t_2]$ (See Figure 4.6(b)). In one equilibrium, $f_{3,2,1} = f_{1,2,1} = 0.5$, i.e., the patroller randomly chooses one target and follows it all the way. In another equilibrium, $f_{3,3,1} = f_{1,1,1} = 0.5$, i.e., the patroller either stays at d_1 or at d_3. In either equilibrium, the attacker's best response is to attack at t_1, with a maximum expected utility of 5. However, if an attacker is physically constrained (e.g., due to launch point locations) to only attack no earlier than t_0 and $t_0 > \theta_1^1$ (where θ_1^1 is the only intersection time point and $\theta_1^1 = (t_1 + t_2)/2$), against both defender strategies he will choose to

attack either of the targets at t_0. The attacker's expected utility is $U_q(t_0)/2$ in the first equilibrium because there is 50% probability that the patroller is following that target. However in the second equilibrium, he is assured to succeed and get a utility of $U_q(t_0)$ because the distance between the chosen target and d_1 (or d_3) is larger than r_e at t_0, i.e., the chosen target is unprotected at t_0. In this case, the defender strategy in the first equilibrium is preferable to the one in the second; indeed, the first defender strategy dominates the second one, by which we mean the first is equally good or better than the second no matter what strategy the attacker chooses. We provide a formal definition of dominance in Section 4.3.1.



(a) Two targets moves with schedules S_1 and S_2. (b) Utility function is the same for both targets and is decreasing linearly over time.

Figure 4.6: An example to show one equilibrium outperforms another when the attacker is constrained to attack in $[t_0, t_2]$ if $t_0 > \theta_1^1$.

Our goal is to improve the defender strategy so that it is more robust against constrained attackers while keeping the defender's expected utility against unconstrained attackers the same. This task of selecting one from the multiple equilibria of a game is an instance of the *equilibrium refinement* problem, which has received extensive study in game theory (van Damme, 1987; Fudenberg & Tirole, 1991; Miltersen & Sørensen, 2007). For finite security games, (An et al., 2011) proposed techniques that provide refinement over Stackelberg equilibrium. However, there has been little prior research on the computation of equilibrium refinements for continuous games.

In this section, we introduce two equilibrium refinement approaches: "route-adjust" (Section 4.3.1) and "flow-adjust" (Section 4.3.2). Both approaches can be applied to improve any feasible defender strategy, and when they are applied to an optimal defender strategy in an existing equilibrium, we will get new equilibria with more robust optimal defender strategies.

For expository simplicity, we still use the single-resource case as an example, but both methods apply to the multiple-resources case. The results shown in evaluation section experimentally illustrates these two refinement methods can significantly improve the performance.

4.3.1 Route Adjust

Given that f is the defender strategy of one equilibrium of the game, if we can find a defender strategy f' such that for any attacker strategy (q, t), the defender's expected utility under f' is equal to or higher than the one under f, and the one under f' is strictly higher than the one under f for at least one specific attacker strategy, we say that f' dominates f. Intuitively, the defender should choose f' instead of f as f' is at least as good as f for any attacker strategy and can achieve better performance for some attacker strategies. So an equilibrium with strategy f' is more robust to unknown deviations on the attacker side. We give the formal definition of dominance as follows.

Definition 7. *Defender strategy f **dominates** f' if $\forall q, t$, DefEU$_f(F_q, t) \geq$ DefEU$_{f'}(F_q, t)$, and $\exists q, t$, DefEU$_f(F_q, t) >$ DefEU$_{f'}(F_q, t)$; or equivalently in this zero-sum game, $\forall q, t$, AttEU$_f(F_q, t) \leq$ AttEU$_{f'}(F_q, t)$, and $\exists q, t$, AttEU$_f(F_q, t) <$ AttEU$_{f'}(F_q, t)$.*

Corollary 2. *Defender strategy f **dominates** f' if $\forall q, t$, $\omega(F_q, t) \geq \omega'(F_q, t)$ and $\exists q, t$, $\omega(F_q, t) > \omega'(F_q, t)$.*

Definition 7 simply restates the commonly used weak dominance definition in game theory for this specific game. Corollary 2 follows from Equation (4.1).

In this section, we introduce the route-adjust approach which gives a procedure for finding a defender strategy f^1 that dominates the given defender strategy f^0. Route-adjust provides final routes using these steps: (i) decompose flow distribution f^0 into component routes; (ii) for each route, greedily find a route which provides better protection to targets; (iii) combine the resulting routes into a new flow distribution, f^1, which dominates f^0 if f^1 is different from f_0. The detailed process is listed in Algorithm 2. We illustrate this approach using a simple dominated strategy shown in Figure 4.3.

To accomplish step (i), we decompose the flow distribution by iteratively finding a route that contains the edge with minimum probability. As shown in Figure 4.7, we first randomly choose a route that contains edge $E_{1,2,2}$, as $f(1,2,2) = 0.4$ is the minimum among all flow variables. We choose $R_2 = (d_1, d_1, d_2)$, and set $p(R_2) = f(1,2,2) = 0.4$. Then for each edge of the route R_2 we subtract 0.4 from the original flow, resulting in a residual flow. We continue to extract routes from the residual flow until there is no route left. Denote by Z the number of non-zero edges in the flow distribution graph, then Z is decreased by at least 1 after each iteration. So the algorithm will terminate in at most Z steps and at most Z routes are found. The result of step (i) is a sparse description of a defender mixed strategy in full representation. As we will discuss in Section 4.5, this decomposition constitutes one method of executing a compact strategy.

For step (ii), we adjust each of the routes greedily. To that end, we first introduce the dominance relation of edges and routes, using the intersection points θ^r_{qk} and the coefficient matrix $A^r_{qk}(i,j)$ defined in Section 4.2.3.

Algorithm 2 Route-Adjust

Input: a mixed defender strategy f
Output: an updated mixed defender strategy f'

(i) Decompose f into multiple routes by iteratively finding a route that contains the edge with minimum probability:

(a) Initialize the remaining flow distribution $\tilde{f} = f$ and route set $S = \emptyset$. Initialize probability distribution over routes $p(R_u) = 0, \forall u$.

(b) **while** $\max \tilde{f}(i, j, k) > 0$ **do**

i. Set $(i_0, j_0, k_0) = \arg\min_{i,j,k:\tilde{f}(i,j,k)>0} \tilde{f}(i, j, k)$.

ii. Set $f_{min} = \tilde{f}(i_0, j_0, k_0)$.

iii. Find an arbitrary route R_{u_0} such that $r_{u_0}(k_0-1) = i_0$ and $r_{u_0}(k_0) = j_0$ (i.e., edge E_{i_0,j_0,k_0} is in the route) and $\tilde{f}(r_{u_0}(k), r_{u_0}(k+1), k) > 0, \forall k$ (i.e., all edges in the route has non-zero remaining flow).

iv. Add R_{u_0} to S and set $p(R_{u_0}) = f_{min}$.

v. Set $\tilde{f}(i, j, k) = \tilde{f}(i, j, k) - f_{min}$ if $r_{u_0}(k-1) = i$ and $r_{u_0}(k) = j$.

end

(ii) Adjust each route in S greedily to get a new set of routes S' and the corresponding new probability distribution p':

(a) Initialize the new set $S' = \emptyset$ and new probability distribution $p'(R_u) = 0$, $\forall u$.

(b) **while** $S \neq \emptyset$ **do**

i. Pick a route R_u from S.

ii. Adjust R_u to get new route $R_{u'}$: for a given R_u and a specified k^*, set $r_{u'}(k) = r_u(k)$ if $k \neq k^*$. Set $r_{u'}(k^*) = i_0$ such that: 1) $E(u_1, k^* - 1)$ and $E(u_1, k^*)$ meet the speed constraint; 2) $R_{u'}$ dominates R_u with the choice of i_0; 3) $R_{u'}$ is not dominated by a route with any other choice of i_0. If no such i_0 exists, set $r_{u'}(k^*) = r_u(k^*)$

iii. Add R_u to S' and set $p'(R_{u'}) = p(R_u)$.

iv. Remove R_u from S.

end

(iii) Reconstruct a new compact representation f' from S' and p' according to Equation 4.4.

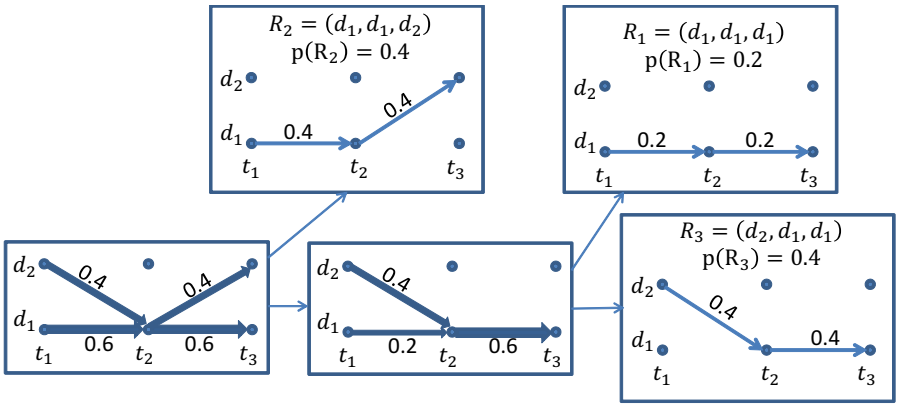


Figure 4.7: Step (i): decomposition. Every time a route containing the minimal flow variable is subtracted and a residual graph is left for further decomposition. The original flow distribution is thus decomposed into three routes R_2, R_1, and R_3 with probability 0.4, 0.2 and 0.4 respectively.

Definition 8. *Edge* $E_{i,j,k}$ ***dominates*** *edge* $E_{i',j',k}$ *in* $[t_k, t_{k+1}]$ *if* $A^r_{qk}(i,j) \geq A^r_{qk}(i',j')$, $\forall q = 1..L$, $\forall r = 0..M_{qk}$, *and* $\exists q, r$ *such that* $A^r_{qk}(i,j) > A^r_{qk}(i',j')$.

The dominance relation of edges is based on the comparison of protection provided to the targets in each sub-interval. In the following dominance relation of routes, we denote the edge $E_{r_u(k),r_u(k+1),k}$ as $E(u,k)$ to simplify the notation, .

Definition 9. *Route* $R_u = (d_{r_u(1)}, \ldots, d_{r_u(M)})$ ***dominates*** $R_{u'} = (d_{r_{u'}(1)}, \ldots, d_{r_{u'}(M)})$ *if* $\forall k = 1 \ldots M-1$, $E(u,k) = E(u',k)$ *or* $E(u,k)$ *dominates* $E(u',k)$ *and* $\exists k$ *such that* $E(u,k)$ *dominates* $E(u',k)$.

Route R_u dominates $R_{u'}$ if each edge of R_u is either the same as or dominates the corresponding edge in $R_{u'}$ and at least one edge in R_u dominates the corresponding edge in $R_{u'}$.

Denote the original route to be adjusted as R_u and the new route as $R_{u'}$. A greedy way to improve the route is to replace only one node in the route. If we want to replace the node at time t_{k^*}, then we have $r_{u'}(k) = r_u(k)$, $\forall k \neq k^*$ and $d_{r_u(k^*)}$ in the original route is replaced with

$d_{r_{u'}(k^*)}$. So the patroller's route changes only in $[t_{k^*-1}, t_{k^*+1}]$. Thus, only edges $E(u, k^* - 1)$ and $E(u, k^*)$ in the original route are replaced by $E(u', k^* - 1)$ and $E(u', k^*)$ in the new route.

We are trying to find a new route $R_{u'}$ that dominates the original route to provide equal or more protection to the targets. So the selection of $r_{u'}(k^*)$ needs to meet the requirements specified in Algorithm 2. The first one describes the speed limit constraint. The second one requires the changed edges $E(u', k^* - 1)$ and $E(u', k^*)$ are either equal to or dominate the corresponding edges in the original route (and dominance relation exist for at least one edge). The third requirement attains a local maximum. If such a new node does not exist for a specified k^*, we return the original route R_u.

We can iterate this process for the new route and get a final route denoted by $R_{u'}$ after several iterations or when the state of convergence is reached. When the state of convergence is reached, the resulting route $R_{u'}$ keeps unchanged no matter which k^* is chosen for the next iteration.

For the example in Figure 4.7, assume the only target's moving schedule is $d_1 \to d_1 \to d_2$, $d_3 - d_2 = d_2 - d_1 = \delta_d$, $r_e = 0.1\delta_d$ and utility function is constant. We adjust each route for only one iteration by changing the patroller's position at time t_3, i.e., $r_u(3)$. As t_3 is the last discretized time point, only edge $E(u, 2)$ may be changed. For $R_1 = (d_1, d_1, d_1)$, we enumerate all possible patroller's positions at time t_3 and choose one according to the three constraints mentioned above. In this case, the candidates are d_1 and d_2, so the corresponding new routes are R_1 (unchanged) and $R_2 = (d_1, d_1, d_2)$ respectively. Note that edge $E_{d_1,d_2,2}$ dominates $E_{d_1,d_1,2}$ because the former one protects the target all the way in $[t_2, t_3]$ and thus R_2 dominates R_1. So d_2 is chosen as the patroller's position at t_3 and R_2 is chosen as the new route. The adjustment for all routes with non-zero probability after decomposition is shown in Table 4.3.

The new routes we get after step (ii) are same as the original routes or dominate the original routes. That is, whenever a route R_u is chosen according to the defender mixed strategy resulting from step (i), it is always equally good or better to choose the corresponding new route $R_{u'}$ instead, because $R_{u'}$ provides equal or more protection to the targets than R_u. Suppose there are H possible routes in the defender strategy after step (i), denoted as $R_1, ..., R_H$. After adjusting the routes, we get a new defender strategy $(p'(R_1), p'(R_2), ..., p'(R_H))$ in full representation (See Table 4.4). Some routes are taken with higher probability (e.g. $p'(R_2) = 0.2 + 0.4 = 0.6$) and some are with lower probability (e.g. $p'(R_3) = 0$) compared to the original strategy. For step (iii), we reconstruct a new compact representation according to Equation 4.4. This is accomplished via a process that is the inverse of decomposition and is the same as how we map a strategy in full representation into a compact representation. For the example above, the result is shown in Table 4.4.

Theorem 3. *After steps (i)–(iii), we get a new defender strategy f^1 that dominates the original one f^0 if f^1 is different from f^0.*

Proof: We continue to use the notation that the decomposition in step (i) yields the routes $R_1, ..., R_H$. For each flow distribution variable in the original distribution $f^0(i, j, k)$, it is decomposed into H sub-flows $\{f_u^0(i, j, k)\}$ according to the route decomposition. $f_u^0(i, j, k) = p(R_u)$ if $i = r_u(k), j = r_u(k + 1)$ and $f_u^0(i, j, k) = 0$ otherwise. Thus we have the following equation.

$$f^0(i, j, k) = \sum\nolimits_{u=1}^{H} f_u^0(i, j, k) \tag{4.28}$$

$$= \sum\nolimits_{u:r_u(k)=i, r_u(k+1)=j} f_u^0(i, j, k) \tag{4.29}$$

After adjust each route separately, each non-zero sub-flow $f_u^0(i,j,k)$ on edge $E(u,k)$ is moved to edge $E(u',k)$ as route R_u is adjusted to $R_{u'}$. Reconstructing the flow distribution f^1 can also be regarded as adding up all the sub-flows after adjustment together on each edge. That means, f^1 is composed of a set of sub-flows after adjustment, denoted as $\{f_u^1(i',j',k)\}$. The subscript u represents for the index of the **original** route to indicate it is moved from edge $E(u,k)$. So $f_u^1(i',j',k) = f_u^0(r_u(k), r_u(k+1), k)$, if $i' = R_{u'}(k)$ and $j' = R_{u'}(k+1)$; otherwise $f_u^1(i',j',k) = 0$. Similarly to Equation 4.29, we have the following equation for f^1.

$$f^1(i',j',k) = \sum\nolimits_{u=1}^{H} f_u^1(i',j',k) \tag{4.30}$$

$$= \sum\nolimits_{u':r_{u'}(k)=i', r_{u'}(k+1)=j'} f_u^1(i',j',k) \tag{4.31}$$

Based on how the adjustment is made, $R_{u'}$ is same as or dominates R_u and thus $E(u',k)$ is same as or dominates $E(u,k)$. So if edge $E(u,k)$ protects target F_q at time t, the corresponding edge $E(u',k)$ after adjustment also protects target F_q at time t.

Recall from Section 4.2.3 that $\omega(F_q,t)$ is the sum of $f(i,j,k)$ whose corresponding edge $E_{i,j,k}$ can protect the target F_q at time t. We denote by $\omega^0(F_q,t)$ and $\omega^1(F_q,t)$ the probabilities of protection corresponding to f^0 and f^1 respectively. According to Equation 4.29, $\omega^0(F_q,t)$ can be viewed as the sum of all the non-zero sub-flows $f_u^0(i,j,k)$ where the corresponding $E(u,k)$ protects the target F_q at time t. If $f_u^0(i,j,k)$ is a term in the summation to calculate $\omega^0(F_q,t)$, it means $E(u,k)$ protects F_q at t and thus the corresponding $E(u',k)$ protects F_q at t, so the corresponding sub-flow $f_u^1(r_{u'}(k), r_{u'}(k+1), k)$ in f^1 is also a term in the summation to calculate $\omega^1(F_q,t)$. It leads to the conclusion $\omega^0(F_q,t) \leq \omega^1(F_q,t)$. Note that if $\forall q,t$, $\omega^0(F_q,t) = \omega^1(F_q,t)$, then all routes kept unchanged in step (ii) as otherwise it contradicts with the fact that

the new route dominates the original route. According to Corollary 2, we have f^1 dominates f^0 if it is different from f^0. □

In the example in Figure 4.7, $f^0(1,1,2)$ is decomposed into two non-zero terms $f_1^0(1,1,2) = 0.2$ and $f_3^0(1,1,2) = 0.4$ along with routes R_1 and R_3 (See Figure 4.7). After adjustment, we get the corresponding subflows $f_1^1(1,2,2) = 0.2$, $f_3^1(1,2,2) = 0.4$. Recall that the target's schedule is $d_1 \rightarrow d_1 \rightarrow d_2$. The flow distribution after adjustment (See Table 4.6) gives more protection to the target in $[t_2, t_3]$. Since the flow is equal from t_1 to t_2 (and therefore the protection is the same), overall the new strategy dominates the old strategy.

Therefore, if we apply route-adjust to the optimal defender strategy calculated by CASS we get a more robust equilibrium. While step (iii) allows us to prove Theorem 3, notice that at the end of step (ii), we have a probability distribution over a set of routes from which we can *sample actual patrol routes*. For two or more defender resources, a generalized version of Definition 8 can be used to define the dominance relation on the edge tuple $(E_{i_1,j_1,k}, ..., E_{i_W,j_W,k})$ with coefficient matrix for multiple patrollers $A^r_{qk}(i_1, j_1, ..., i_W, j_W)$.

There are other ways to adjust each route. Instead of adjusting only one node in the route, we can adjust more consecutive nodes at a time, for example, we can adjust both $r_{u'}(k^*)$ and $r_{u'}(k^* + 1)$ by checking edges $E(u', k^* - 1)$, $E(u', k^*)$ and $E(u', k^* + 1)$. However, we need to tradeoff the performance and the efficiency of the algorithm. This tradeoff will be further discussed in Section 4.6.

4.3.2 Flow Adjust

Whereas route-adjust tries to select an equilibrium that is robust against attackers playing sub-optimal strategies, the second approach, *flow-adjust*, attempts to select a new equilibrium that

is robust to rational attackers that are constrained to attack during any time interval $[t_k, t_{k+1}]$. As we will discuss below, flow-adjust focuses on a weaker form of dominance, which implies that a larger set of strategies are now dominated (and thus could potentially be eliminated) compared to the standard notion of dominance used by route-adjust; however flow-adjust does not guarantee the elimination of all such dominated strategies. We denote by DefEU_f^k the defender expected utility when an attacker is constrained to attack during time interval $[t_k, t_{k+1}]$ when the attacker provides his best response given the defender strategy f. Formally, $\text{DefEU}_f^k = \min_{q\in\{1...L\},t\in[t_k,t_{k+1}]}\{\text{DefEU}_f(F_q,t)\}$. We give the following definition of "local dominance".

Definition 10. *Defender strategy f* ***locally dominates*** *f' if $DefEU_f^k \geq DefEU_{f'}^k, \forall k$.*[2]

Corollary 3. *Defender strategy f* ***locally dominates*** *f' if*

$$\min_{q\in\{1...L\},t\in[t_k,t_{k+1}]}\{DefEU_f(F_q,t)\} \geq \min_{q\in\{1...L\},t\in[t_k,t_{k+1}]}\{DefEU_{f'}(F_q,t)\}, \forall k,$$

or equivalently in this zero-sum game,

$$\max_{q\in\{1...L\},t\in[t_k,t_{k+1}]}\{AttEU_f(F_q,t)\} \leq \max_{q\in\{1...L\},t\in[t_k,t_{k+1}]}\{AttEU_{f'}(F_q,t)\}, \forall k.$$

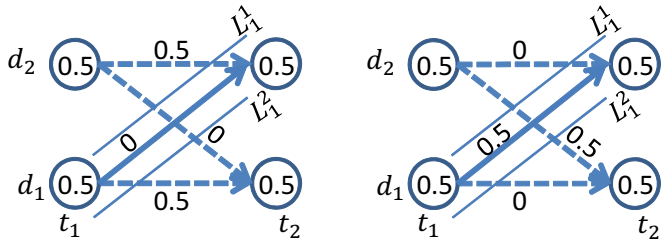
Corollary 3 follows from the fact that the attacker plays a best response given the defender strategy, and it means that f locally dominates f' if the maximum of attacker expected utilities in each time interval $[t_k, t_{k+1}]$ given f is no greater than that of f'.

[2]We don't require that there exists at least one k such that $\text{DefEU}_f^k > \text{DefEU}_{f'}^k$.

Compared to Definition 7, which gives the standard condition for dominance, local dominance is a weaker condition; that is if f dominates f' then f locally dominates f'. However, the converse is not necessarily true. Intuitively, whereas in Definition 7 the attacker can play any (possibly suboptimal) strategy, here the attacker's possible deviations from best response are more restricted. As a result, the set of locally dominated strategies includes the set of dominated strategies. From Definition 10, if f locally dominates f', and the attacker is rational (i.e., still playing a best response) but constrained to attack during some time interval $[t_k, t_{k+1}]$, then f is preferable to f' for the defender. A further corollary is that even if the rational attacker is constrained to attack in the union of some of these intervals, f is still preferable to f' if f locally dominates f'. One intuition for the local dominance concept is the following: suppose we suspect the attacker will be restricted to an (unknown) subset of time, due to some logistical constraints. Such logistical constraints would likely make the restricted time subset to be contiguous or a union of a small number of contiguous sets. Since such sets are well-approximated by unions of intervals $[t_k, t_k + 1]$, local dominance can serve as an approximate notion of dominance with respect to such attackers.

Flow-adjust looks for a defender strategy f^1 that locally dominates the original defender strategy f^0. To achieve this, we simply adjust the flow distribution variables $f(i, j, k)$ while keeping the marginal probabilities $p(i, k)$ the same. Figure 4.8 shows an example game with two discretized intervals $[t_1, t_2]$ and $[t_2, t_3]$ (only the first interval is shown). Suppose the maximal attacker expected utility is $5U_0$ in this equilibrium and is attained in the second interval $[t_2, t_3]$. If the attacker's utility for success is a constant U_0 in the first interval $[t_1, t_2]$, then the defender strategy in $[t_1, t_2]$ could be arbitrarily chosen because the attacker's expected utility in $[t_1, t_2]$ in worst case is smaller than that of the attacker's best response in $[t_2, t_3]$. However, if a attacker is

constrained to attack in $[t_1, t_2]$ only, the defender strategy in the first interval will make a difference. In this example, there is only one target moving from d_1 to d_2 during $[t_1, t_2]$. The schedule of the ferry is shown as dark lines and the parallel lines L_1^1 and L_1^2 with respect to protection radius $r_e = 0.2(d_2 - d_1)$ are shown as dashed lines. The marginal distribution probabilities $p(i, k)$ are all 0.5 and protection coefficient $C_1 = 1$. In f^0, the defender's strategy is taking edges $E_{1,1,1}$ and $E_{2,2,1}$ with probability 0.5 and the attacker's maximum expected utility is U_0, which can be achieved around time $(t_1 + t_2)/2$ when neither of the two edges $E_{1,1,1}$ and $E_{2,2,1}$ are within the target's protection range. If we adjust the flows to edge $E_{1,2,1}$ and $E_{2,1,1}$, as shown in Figure 4.8(b), the attacker's maximum expected utility in $[t_1, t_2]$ is reduced to $0.5U_0$ as edge $E_{1,2,1}$ is within the target's protection range all the way. So a rational attacker who is constrained to attack between $[t_1, t_2]$ will get a lower expected utility given defender strategy f^1 than given f^0, and thus the equilibrium with f^1 is more robust to this kind of deviation on the attacker side.



(a) f^0: the patroller is taking edges $E_{1,1,1}$ and $E_{2,2,1}$ with probability 0.5.

(b) f^1: the patroller is taking edges $E_{1,2,1}$ and $E_{2,1,1}$ with probability 0.5.

Figure 4.8: An example of flow adjust. An rational attacker who is constrained to attack in $[t_1, t_2]$ will choose to attack around time $(t_1 + t_2)/2$ to get utility U_0 given f^0 and attack around t_1 or t_2 to get utility $0.5U_0$ given f^1.

So in flow-adjust, we construct $M - 1$ new linear programs, one for each time interval $[t_{k^*}, t_{k^*+1}]$, $k^* = 1 \dots M - 1$ to find a new set of flow distribution probabilities $f(i, j, k^*)$

to achieve the lowest local maximum in $[t_{k^*}, t_{k^*+1}]$ with unchanged $p(i, k^*)$ and $p(i, k^* + 1)$. The linear program for an interval $[t^*_k, t_{k^*+1}]$ is shown below.

$$\min_{f(i,j,k^*)} v$$

$$f(i, j, k^*) = 0, \text{ if } |d_j - d_i| > v_m * \delta_t$$

$$p(i, k^* + 1) = \sum_{j=1}^{n} f(j, i, k^*), \forall i \in \{1 \dots n\}$$

$$p(i, k^*) = \sum_{j=1}^{n} f(i, j, k^*), \forall i \in \{1 \dots n\}$$

$$v \geq AttEU(F_q, t_k), \forall q \in \{1 \dots L\}, k \in \{k^*, k^* + 1\}$$

$$v \geq \max\{AttEU(F_q, \theta^{r+}_{qk^*}), AttEU(F_q, \theta^{(r+1)-}_{qk^*})\}$$

$$\forall q \in \{1 \dots L\}, r \in \{0 \dots M_{qk^*}\}$$

While the above linear program appears similar to the linear program of CASS, they have significant differences. Unlike CASS, the marginal probabilities $p(i, k^*)$ here are known constants and are provided as input and as mentioned above, there is a separate program for each $[t_{k^*}, t_{k^*+1}]$. Thus, we get $f(i, j, k^*)$ such that the local maximum in $[t_{k^*}, t_{k^*+1}]$ is minimized. Denote the minimum as $v^1_{k^*}$. From the original flow distribution f^0, we get $AttEU_{f^0}(F_q, t)$ and we denote the original local maximum value in $[t_{k^*}, t_{k^*+1}]$ as $v^0_{k^*}$. As the subset $\{f^0(i, j, k^*)\}$ of the original flow distribution f^0 is a feasible solution of the linear program above, we have $v^1_{k^*} \leq v^0_{k^*}$, noting that the equality happens for the interval from which the attacker's best response is chosen.

Note that any change made to $f(i, j, k)$ in an interval $[t^*_k, t_{k^*+1}]$ will not affect the performance of f in other intervals as the marginal probabilities $p(i, k)$ are kept the same, i.e., changing $f(i, j, k^*)$ based on the linear program above is independent from any change to

$f(i,j,k), k \neq k^*$. So we can solve the $M-1$ linear programs independently. After calculating $f(i,j,k^*)$ for all $k^* = 1..M-1$, we can get the new defender strategy f^1 by combining the solutions $f(i,j,k^*)$ of the different linear programs together. As $v^1_{k^*} \leq v^0_{k^*}$, we have

$$\max_{q\in\{1...L\},t\in[t_{k^*},t_{k^*+1}]} AttEU_{f^0}(F_q,t) \leq \max_{q\in\{1...L\},t\in[t_{k^*},t_{k^*+1}]} AttEU_{f^1}(F_q,t)$$

for all $k^* = 1..M-1$, i.e., f^1 locally dominates f^0.

On the other hand, while we have restricted the strategies to have the same $p(i,k)$, there may exist another strategy f^2 with a different set of $p(i,k)$ that locally dominates f^1. Finding locally dominating strategies with different $p(i,k)$ from the original is a topic of future research.

Although the two refinement approaches we provide do not necessarily lead to a non-dominated strategy under the corresponding dominance definition, these two approaches are guaranteed to find a more robust (or at least indifferent) equilibrium when faced with constrained attackers compared to the original equilibrium we obtain from CASS. Clearly, these two refinement approaches do not exhaust the space of refinement approaches — other refinement approaches are possible that may lead to other equilibria that are better than (e.g. dominate) the one found by CASS. However, it is likely that different defender strategies resulting from different equilibrium refinements are not comparable to each other in terms of dominance, i.e., with some constrained attackers, one equilibrium might turn out to be better and with other constrained attackers, another equilibrium might be better. Their computational costs may differ as well. Thus, understanding this space of refinement approaches regarding their computational cost and output quality, and determining which approach should be adopted under which circumstances is an important challenge for future work.

4.4 Extension To Two-Dimensional Space

Both DASS and CASS presented in Section 4.2 are based on the assumption that both the targets and the patrollers move along a straight line. However, a more complex model is needed in some practical domains. For example, Figure 4.9 shows a part of the route map of Washington State Ferries, where there are several ferry trajectories. If a number of patroller boats are tasked to protect all the ferries in this area, it is not necessarily optimal to simply assign a ferry trajectory to each of the patroller boats and calculate the patrolling strategies separately according to CASS described in Section 4.2. As the ferry trajectories are close to each other, a patrolling strategy that can take into account all the ferries in this area will be much more efficient, e.g., a patroller can protect a ferry moving from Seattle to Bremerton first, and then change direction halfway and protect another ferry moving from Bainbridge Island back to Seattle.

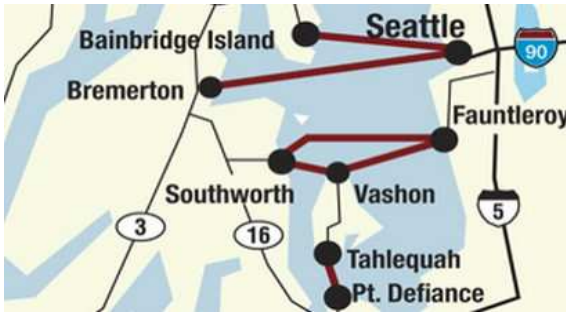


Figure 4.9: Part of route map of Washington State Ferries

In this section, we extend the previous model to a more complex case, where the targets and patrollers move in a two-dimensional space and provide the corresponding linear-program-based solution. Again we use a single defender resource as an example and generalize to multiple defenders at the end of this section.

4.4.1 Defender Strategy for 2-D

As in the one-dimensional case, we need to discretize the time and space for the defender to calculate the defender's optimal strategy. The time interval T is discretized into a set of time points $T = \{t_k\}$. Let $G = (V, E)$ represents the graph where the set of vertices V corresponds to the locations that the patrollers may be at, at the discretized time points in T, and E is the set of feasible edges that the patrollers can take. An edge $e \in E$ satisfies the maximum speed limit of patroller and possibly other practical constraints (e.g., a small island may block some edges).

4.4.2 DASS for 2-D

When the attack only occurs at the discretized time points, the linear program of DASS and described in Section 4.2 can be applied to the two-dimensional settings when the distance in Constraint 4.9 is substituted with Euclidean distance in 2-D space of nodes V_i and V_j.

$$\min_{f(i,j,k),p(i,k)} v \tag{4.32}$$

$$f(i,j,k) \in [0,1], \forall i,j,k \tag{4.33}$$

$$f(i,j,k) = 0, \forall i,j,k \text{ such that } ||V_j - V_i|| > v_m \delta_t \tag{4.34}$$

$$p(i,k) = \sum_{j=1}^{N} f(j,i,k-1), \forall i, \forall k > 1 \tag{4.35}$$

$$p(i,k) = \sum_{j=1}^{N} f(i,j,k), \forall i, \forall k < M \tag{4.36}$$

$$\sum_{i=1}^{N} p(i,k) = 1, \forall k \tag{4.37}$$

$$v \geq \text{AttEU}(F_q, t_k), \forall q, \forall k \tag{4.38}$$

Note that $f(i,j,k)$ now represents the probability that a patroller is moving from node V_i to V_j during $[t_k, t_{k+1}]$. Recall in Figure 4.1.1, a patroller protects all targets within her protective circle of radius r_e. However, in the one-dimensional space, we only care about the straight line AB, so we used $\beta_q(t) = [\max\{S_q(t) - r_e, d_1\}, \min\{S_q(t) + r_e, d_N\}]$ as the protection range of target F_q at time t, which is in essence a line segment. In contrast, here the whole circle needs to be considered as the protection range in the two-dimensional space and the extended protection range can be written as $\beta_q(t) = \{V = (x,y) : ||V - S_q(t)|| \leq r_e\}$. This change affects the value of $I(i,q,k)$ and thus the value of $AttEU(F_q, t_k)$ in Constraint 4.38.

4.4.3 CASS for 2-D

When the attacking time t can be chosen from the continuous time interval T, we need to analyze the problem in a similar way as in Section 4.2.3. The protection radius is r_e, which means only patrollers located within the circle whose origin is $S_q(t)$ and radius is r_e can protect target F_q. As we assume that the target will not change its speed and direction during time $[t_k, t_{k+1}]$, the circle will also move along a line in the 2-D space. If we track the circle in a 3-D space where the x and y axes indicate the position in 2-D and the z axis is the time, we get an oblique cylinder, which is similar to a cylinder except that the top and bottom surfaces are displaced from each other (See Figure 4.10). When a patroller moves from vertex $V_i (\in V)$ to vertex V_j during time $[t_k, t_{k+1}]$, she protects the target only when she is within the oblique cylinder. In the 3-D space we described above, the patroller's movement can be represented as a straight line.

Intuitively, there will be at most two intersection points between the patroller's route in 3-D space and the surface. This can be proved by analytically calculating the exact time of these intersection points. Assume the patroller is moving from $V_1 = (x_1, y_1)$ to $V_2 = (x_2, y_2)$ and the

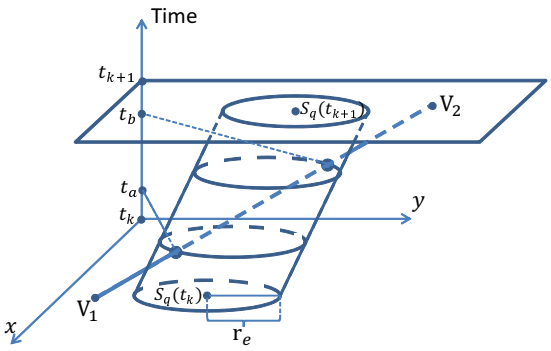


Figure 4.10: An illustration of the calculation of intersection points in the two-dimensional setting. The x and y axes indicates the position in 2-D and the z axis is the time. To simplify the illustration, z axis starts from time t_k. In this example, there are two intersection points occurring at time points t_a and t_b.

target is moving from $S_q(t_k) = (\hat{x_1}, \hat{y_1})$ to $S_q(t_{k+1}) = (\hat{x_2}, \hat{y_2})$ during $[t_k, t_{k+1}]$ (an illustration is shown in Figure 4.10). To get the time of the intersection points, we solve a quadratic equation with these coordination parameters and protection radius r_e. If a root of the quadratic equation is within the interval $[t_k, t_{k+1}]$, it indicates that the patroller's route intersects with the surface at this time point. So there will be at most two intersection points. Once we find all these intersection points, the same analysis in Section 4.2.3 applies and we can again claim Lemma 1. So we conclude that we only need to consider the attacker's strategies at these intersection points. We use the same notation θ^r_{qk} as in the one-dimensional case to denote the sorted intersection points and get the following linear program for the 2-D case.

$$\min_{f(i,j,k),p(i,k)} v \tag{4.39}$$

subject to constraints$(4.33 \dots 4.38)$

$$v \geq \max\{\text{AttEU}(F_q, \theta^{r+}_{qk}), \text{AttEU}(F_q, \theta^{(r+1)-}_{qk})\} \tag{4.40}$$

$$\forall k \in \{1 \dots M\}, q \in \{1 \dots L\}, r \in \{0 \dots M_{qk}\}$$

Algorithm 4.2.3 can still be used to add constraints to the linear program of CASS for the 2-D case. The main difference compared to CASS in the 1-D case is that since Euclidean distance in 2-D is used in Constraint 4.34 we need to use the extended definition of $\beta_q(t)$ in 2-D when deciding the entries in the coefficient matrix $A^r_{qk}(i,j)$.

The detailed calculation for finding the intersection points is shown below. We calculate the time where the patroller's route intersects with the protection range for a target when the patroller is moving from $V_1 = (x_1, y_1)$ to $V_2 = (x_2, y_2)$ and the target is moving from $S_q(t_k) = (\hat{x_1}, \hat{y_1})$ to $S_q(t_{k+1}) = (\hat{x_2}, \hat{y_2})$ during $[t_k, t_{k+1}]$. The patroller's position at a given time $t \in [t_k, t_{k+1}]$ is denoted as (x, y) and the target's position is denoted as $(\hat{x}, \hat{y})$. Then we have

$$x = \frac{t - t_k}{t_{k+1} - t_k}(x_2 - x_1) + x_1, \qquad y = \frac{t - t_k}{t_{k+1} - t_k}(y_2 - y_1) + y_1 \tag{4.41}$$

$$\hat{x} = \frac{t - t_k}{t_{k+1} - t_k}(\hat{x_2} - \hat{x_1}) + x_1, \qquad \hat{y} = \frac{t - t_k}{t_{k+1} - t_k}(\hat{y_2} - \hat{y_1}) + \hat{y_1} \tag{4.42}$$

At an intersection point, the distance from the patroller's position to the target's position equals to the protection radius r_e, so we are looking for a time t such that

$$(x - \hat{x})^2 + (y - \hat{y})^2 = r_e^2 \tag{4.43}$$

By substituting the variables in Equation 4.43 with Equations 4.41–4.42, and denoting

$$A_1 = \frac{(x_2 - x_1) - (\hat{x_2} - \hat{x_1})}{t_{k+1} - t_k}, \qquad B_1 = x_1 - \hat{x_1},$$

$$A_2 = \frac{(y_2 - y_1) - (\hat{y_2} - \hat{y_1})}{t_{k+1} - t_k}, \qquad B_2 = y_1 - \hat{y_1},$$

Equation 4.43 can be simplified to

$$(A_1 t - A_1 t_k + B_1)^2 + (A_2 t - A_2 t_k + B_2)^2 = r_e^2. \tag{4.44}$$

Denote $C_1 = B_1 - A_1 t_k$ and $C_2 = B_2 - A_2 t_k$, and we can easily get the two roots of this quadratic equation, which are

$$t_{a,b} = \frac{-2(A_1 C_1 + A_2 C_2) \pm 2\sqrt{(A_1 C_1 + A_2 C_2)^2 - (A_1^2 + A_2^2)(C_1^2 + C_2^2 - r_e^2)}}{2(A_1^2 + A_2^2)}. \tag{4.45}$$

t_a or t_b is the time of a valid intersection point if and only if it is within the time interval under consideration ($[t_k, t_{k+1}]$).

For multiple defender resources, again the linear program described in Section 4.2.5 is applicable when the extended definition of $\beta_q(t)$ is used to calculate $AttEU$ and Constraint 4.21 is substituted with the following constraint:

$$f(i_1, j_1, \ldots, i_W, j_W, k) = 0, \forall i_1, \ldots, i_W, j_1, \ldots, j_W \text{ such that } \exists u, \|V_{j_u} - V_{i_u}\| > v_m \delta_t.$$

4.5 Route Sampling

We have discussed how to generate an optimal defender strategy in the compact representation; however, the defender strategy will be executed as taking a complete route. So we need to sample a complete route from the compact representation. In this section, we give two methods of sampling and show the corresponding defender strategy in the full representation when these methods are applied.

The first method is to convert the strategy in the compact representation into a Markov strategy. A Markov strategy in our setting is a defender strategy such that the patroller's movement from t_k to t_{k+1} depends only on the location of the patroller at t_k. We denote by $\alpha(i,j,k)$ the conditional probability of moving from d_i to d_j during time t_k to t_{k+1} given that the patroller is located at d_i at time t_k. In other words $\alpha(i,j,k)$ represents the chance of taking edge $E_{i,j,k}$ given that the patroller is already located at node (t_k, d_i). Thus, given a compact defender strategy specified by $f(i,j,k)$ and $p(i,k)$, we have

$$\alpha(i,j,k) = f(i,j,k)/p(i,k), \text{ if } p(i,k) > 0. \tag{4.46}$$

$\alpha(i,j,k)$ can be an arbitrary number if $p(i,k) = 0$. We can get a sampled route by first determining where to start patrolling according to $p(i,1)$; then for each t_k, randomly choose where to go from t_k to t_{k+1} according to the conditional probability distribution $\alpha(i,j,k)$. The distribution from this sampling procedure matches the given marginal variables as each edge $E_{i,j,k}$ is sampled with probability $p(i,k)\alpha(i,j,k) = f(i,j,k)$. This sampling method actually leads to a full representation where route $R_u = (d_{r_u(1)}, d_{r_u(2)}, ..., d_{r_u(M)})$ is sampled with probability $p(r_u(1),1)\prod_{k=1}^{M-1}\alpha(r_u(k), r_u(k+1), k)$, the product of the probability of the initial distribution and the probability of taking each step. This method is intuitively straightforward and the patrol route can be decided online during the patrol, i.e., the position of the patroller at t_{k+1} is decided when the patroller reaches its position at t_k, which makes the defender strategy more unpredictable. The downside of the method is that the number of routes chosen with non-zero probability can be as high as N^M. For 2-D case, the patroller is located at node V_i at time t_k. The

sampling process is exactly the same when $\alpha(i, j, k)$ is used to denote the probability of moving from V_i to V_j during $[t_k, t_{k+1}]$.

The second method of sampling is based on the decomposition process mentioned in Section 4.3.1 (step (i)). As we discussed above for the first sampling method, sampling is essentially restoring a full representation from the compact representation. As shown in Table 4.2, there are multiple ways to assign probabilities to different routes and the decomposition process of "route-adjust" constructively defines one of them. So we can make use of the information we get from the process, and sample a route according to the probability assigned to each decomposed route. The number of routes chosen with non-zero probability is at most N^2M, much less than the first method and thus it becomes feasible to describe the strategy in full representation, by only providing the routes that are chosen with positive probability. Different sampling approaches may be necessitated by different application requirements. Some applications might require that the defender obtains a strategy in full representation and only be presented a small number of pure strategies. However, for other applications, a strategy that can be decided on-line, potentially with a hand-held smartphone such as in (Luber, Yin, Fave, Jiang, Tambe, & Sullivan, 2013) may be preferred. Therefore, based on the needs of the application, different sampling strategies might be selected.

4.6 Evaluation

We use different settings in the ferry protection domain and compare the performance in terms of the attacker's expected utility AttEU(F_q, t). As it is a zero-sum game, a lower value of AttEU indicates a higher value of defender's expected utility.

We will run experiments both for 1-D and 2-D setting. We will evaluate the performance of CASS and show the sampling results. We will also evaluate the improvement of the two refinement approaches for 1-D. Section 4.6.1 shows our results for the 1-D setting; Section 4.6.2 for the 2-D setting.

4.6.1 Experiments for One-Dimensional Setting

For the 1-D setting, we first evaluate the performance of the solvers and then show how much the performance can be improved by using the refinement methods. We also show sampled routes for an example setting and evaluate CASS for varying number of patrollers.

4.6.1.1 Experimental Settings

We used the following setting for the experiments in one-dimensional case. This is a complex spatio-temporal game; rather than a discrete security game as in most previous work. There are three ferries moving between terminals A and B and the total distance $AB = 1$. The simulation time is 30 *minutes*. The schedules of the ferries are shown in Figure 4.11, where the x-axis indicates the time, and the y-axis is the distance from terminal A. Ferry 1 and Ferry 3 are moving from A to B while Ferry 2 is moving from B to A. The maximum speed for patrollers is $v_m = 0.1/min$ and the protection radius is $r_e = 0.1$. Experiments in the one-dimensional case are using 2 patrollers (where $C_1 = 0.8$, and $C_2 = 1.0$), except in Section 4.6.1.5 where we report on experiments with different numbers of patrollers.

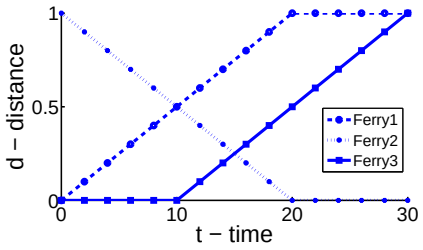
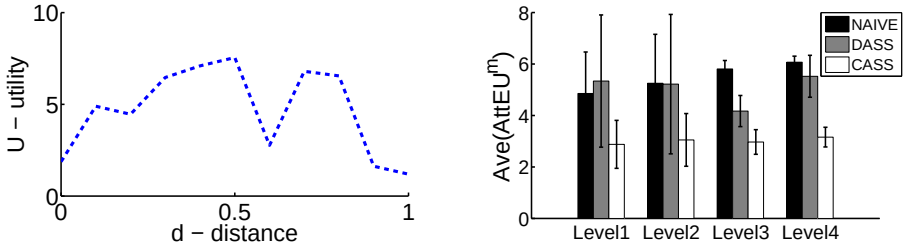


Figure 4.11: Schedules of the ferries

4.6.1.2 Performance of Solvers

We compare the strategies calculated by CASS with DASS and a baseline strategy. In the baseline strategy, the two patrollers choose a ferry with a probability of 1/3 (uniformly random) and move alongside it to offer it full protection, leaving the other two unprotected (strategy observed in practice). First, we wished to stress-test CASS by using more complex utility functions than in the realistic case that follows. Therefore, we tested under four different discretization levels (details about discretization levels are included in Table 4.5) with random utilities, and at each discretization level, we created 20 problem instances. The problem instances are different across levels. In this ferry protection domain, the utility function for each ferry usually depends on the ferry's position, so each instance has utilities uniformly randomly chosen between $[0, 10]$ at discretized distance points; an example is shown in Figure 4.12(a). The chosen discretization levels have ensured that $U_q(t)$ is linear in t in each time interval $[t_k, t_{k+1}]$ for each target F_q. In Figure 4.12(a), the x-axis indicates the distance d from terminal A, the y-axis indicates the utility of a successful attack if the ferry is located at distance d. In Figure 4.12(b), x-axis plots the four discretization levels and y-axis plots the average attacker expected utility if he plays best response over the 20 instances for baseline, DASS and CASS. CASS is shown to outperform DASS and baseline and the differences are statistically significant ($p < 0.01$). Note that different

sets of instances are generated for different discretization levels, so we cannot compare the results across levels directly. However, it is helpful to better understand the models. From the figure, we find the solution quality of DASS varies a lot and sometimes can be worse than the naive strategy (e.g., level 1). This is because DASS calculates an optimal solution that considers only the attacks at the discretized time points. In Figure 4.12(b), the solution quality is measured by $AttEU^m$, which is calculated as the maximum over the continuous attacker strategy set. The gap between the optimal objective function of DASS and the actual $AttEU^m$ given the optimal solution of DASS may vary for different strategies and different discretization levels. Another interesting observation is that the average solution quality of CASS is almost the same for all discretization levels. Despite the difference in instance sets, this result implies that the improvement of a finer discretization may be limited for CASS.

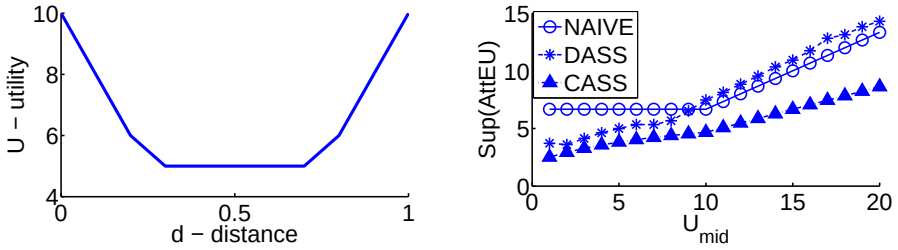


(a) Randomized attacker utility function

(b) Average solution quality of different strategies

Figure 4.12: Performance under different randomized utility function settings. The utility function in this set of experiments is a function of the distance to Terminal A. The utility function is piece-wise linear and the value at discretized distance points d_i is chosen randomly between [0,10].

Next, we turn to *more realistic* utility function in this ferry domain, which is of U-shape or inverse U-shape. Figure 4.13(a) shows a sample utility curve where the attacker gains higher utility closer to the shore. We fix the utility at the shore as 10, vary the utility in the middle (denoted as U_{mid}), which is the value on the floor of the U-shape or the top of the inverse U-shape



(a) Realistic attacker utility function with $U_{mid} = 5$

(b) Solution quality of different strategies

Figure 4.13: Performance under different realistic utility function settings. The utility function is U-shape or inverse U-shape. The utility around distance 0.5 is denoted as U_{mid}. We compare the defender strategy given by DASS and CASS with the baseline when U_{mid} is changing from 1 to 20.

and evaluate the strategies. In Figure 4.13(b), U_{mid} is shown on the x-axis, and we compare the performance of the strategies regarding attacker's expected utility when he plays the best response on the y-axis. We conclude that 1) the strategy calculated by CASS outperforms the baseline and DASS; 2) DASS may achieve worse results than the baseline.

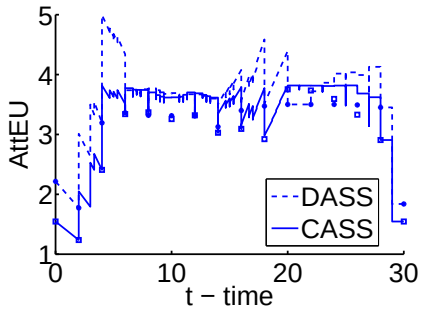


Figure 4.14: The attacker's expected utility function given the defender strategy calculated by DASS vs CASS under example setting. The expected utilities at the discretized time points are indicated by squares for CASS and dots for DASS. The maximum of $AttEU$ under CASS is 3.82, 30%less than the maximum of $AttEU$ under DASS, which is 4.99.

Among all these different experiment settings of discretization and utility function, we choose one instance and provide a more detailed analysis of it. We refer to this instance as *example setting* in the following of this section. In this example setting, discretization level 4 is used and the utility curve is as shown in Figure 4.13(a), other parameters involved are described in Section 4.6.1.1.

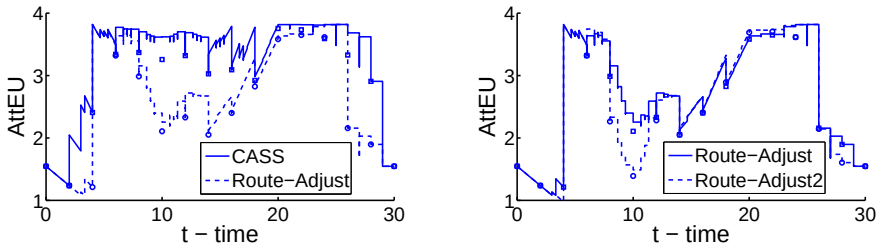
Figure 4.14 compares the attacker expected utility function when DASS and CASS is used respectively. The x-axis indicates the time t, and the y-axis indicates the attacker's expected utility if he attacks Ferry 1 at time t. For the strategy calculated by DASS, the worst performance at discretized time points is 3.50 (AttEU$(F_1, 20)$), however, the supremum of AttEU(F_1, t), $t \in [0, 30]$ can be as high as 4.99 (AttEU$(F_1, 4^+)$), which experimentally shows that taking into consideration the attacks between the discretized time points is necessary. For the strategy calculated by CASS, the supremum of AttEU(F_1, t) is reduced to 3.82.

4.6.1.3 Improvement Using Refinement Methods

We compare the refinement approaches described in Section 4.3 and analyze the tradeoff between performance improvement and runtime. Three approaches are considered for comparison: route-adjust, flow-adjust and a variation of route-adjust, denoted by route-adjust2. In step (ii) of route-adjust, we replace every node in the route one-by-one in sequence.[3] In step (ii) of route-adjust2, we replace every consecutive pair of nodes in the route in sequence.

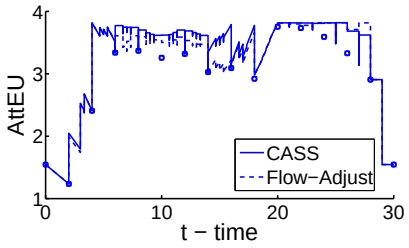
We first show results for the example setting. In Figure 4.15(a), we compare the AttEU(F_q, t) function of the defender strategy given by CASS and of the one after route-adjust for Ferry 1. It shows for an attack aiming at any target at any time, the defender strategy after route-adjust refinement is equally good or better than the one in the original equilibrium, and thus, the defender performs equally or better no matter how the attacker is constrained in time, i.e., the defender strategy after route-adjust dominates the original strategy. Figure 4.15(b) is the comparison between AttEU function of the defender strategy after route-adjust and the one after

[3] In supplementary experiments, we also tested route-adjust with more iterations, e.g., repeating the process of replacing every node in sequence five times. The extra benefit is insignificant while the runtime increases proportionally to the number of iterations. In light of this, we choose to replace each node only once in the experiments reported in this chapter.



(a) AttEU function of Ferry 1 after route-adjust (one node at a time)

(b) AttEU function of Ferry 1 after route-adjust2 (two nodes at a time)



(c) Performance of flow-adjust

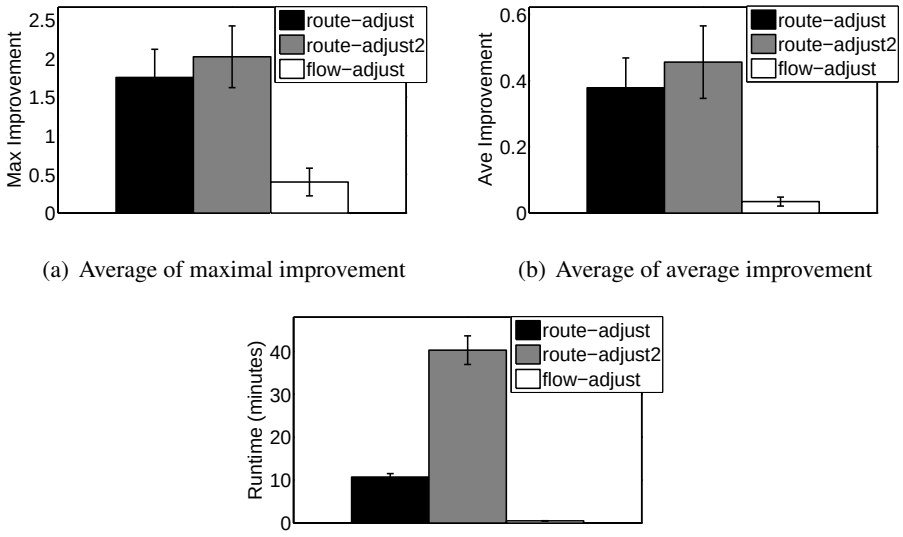
Figure 4.15: Performance of equilibrium refinement approaches.

route-adjust2 for Ferry 1. The one after route-adjust2 does not dominate the one after route-adjust but overall the former appears to perform better than the latter more frequently and by larger amounts. If we use the average value of AttEU function as a metric of performance, we will show that route-adjust2 is better than route-adjust in this example setting later in Table 4.6. Figure 4.15(c) shows the comparison between the AttEU function of the defender strategy given by CASS and that of the defender strategy after flow-adjust for Ferry 1. The strategy given by CASS is not dominated by the one after flow-adjust under Definition 7, but if we investigate the maximum of AttEU in each time interval $[t_k, t_{k+1}]$, as shown in Table 4.7, we find that the defender strategy after flow-adjust locally dominates the original strategy.

We list the worst case performance and the average performance of AttEU function over all ferries in this example setting for four defender strategies (CASS, route-adjust, route-adjust2, flow-adjust) in Table 4.6, from which we conclude that 1) the worst case performance of all

strategies of flow-adjust is the same, which means the defender achieves exactly same expected utility towards an unconstrained rational attacker; 2) the average performance of flow-adjust is slightly better than the CASS, but is outperformed by route-adjust and route-adjust2, while it takes much less time to run compared to the other two; 3) in this example setting, when we adjust two consecutive nodes at a time, the performance is better than adjusting only one node at a time, but the difference is not significant, and it is much more expensive in terms of run-time.

Figure 4.16(a) and Figure 4.16(b) shows the maximum and the average improvement of route-adjust, route-adjust2 and flow-adjust, averaged over all the 20 instances of Level 4 with randomized utilities that have been used for Figure 4.12(b); and Figure 4.16(c) shows the average runtime. The maximum improvement is the largest difference between the AttEU function given defender strategy calculated by CASS and the one after refinement. The average improvement is the average difference between the two functions. The standard deviations over all instances are shown as error bars. Figure 4.16 confirms that all the refinement approaches improve the defender strategy calculated by CASS in terms of both the maximum performance and average performance and thus provide better defender strategies given possible constrained attackers. Route-adjust2 achieves the most improvement, then route-adjust, and flow-adjust the least. Flow-adjust achieves much less improvement compared to the other two approaches. One explanation for this is that the constraints are very strong as they require all marginal probabilities to be unchanged, so it is likely that little changes are made to the original defender strategy. The difference between route-adjust2 and route-adjust is not as significant. Regarding run-time, flow-adjust is the least expensive, route-adjust the second and route-adjust2 the most. Route-adjust2 is significantly more expensive compared to the other two. So we conclude that route-adjust is a better choice considering the tradeoff between improvement and the runtime.



(a) Average of maximal improvement

(b) Average of average improvement

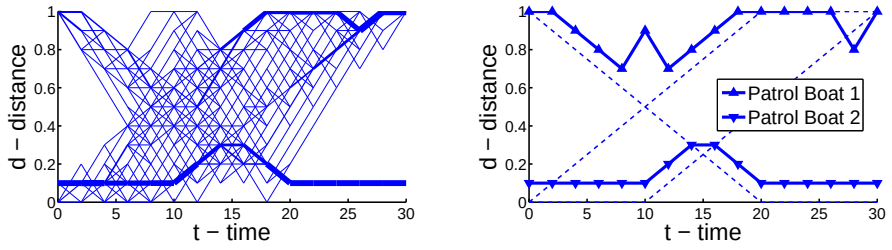
(c) Average of runtime

Figure 4.16: Comparison of refinement approaches.

4.6.1.4 Sampled Routes

We first convert the defender strategy under the example setting into a Markov strategy and sample 1000 pair of patrol routes. The defender strategy used here is the one after "route-adjust". In each sample, a pair of routes is chosen step by step for the two patrol boats according to the joint conditional probability distribution $\{\alpha(i1, j1, i2, j2, k)\}$. The routes for the two patrol routes are chosen simultaneously as they are coordinating with each other. We cannot show each pair separately for all 1000 samples. Instead, Figure 4.17(a) shows the frequency of being taken out of the 1000 samples of each edge. The x-axis indicates the time and the y-axis is the distance to terminal A. The width of the each edge indicates the frequency of being chosen by at least one patroller. Although Figure 4.17(a) does not precisely depict the samples, it provides a rough view of how the routes are taken by the patrol boats.

Figure 4.17(b) shows the pair of routes that is of highest probability when we use the decomposition method of sampling. The solid lines show the patrol boats' routes and the dashed lines show the ferries' schedules. We get 3958 different pairs of patrol routes in total in the decomposition process, and the shown pair of routes is chosen with probability 1.57%.

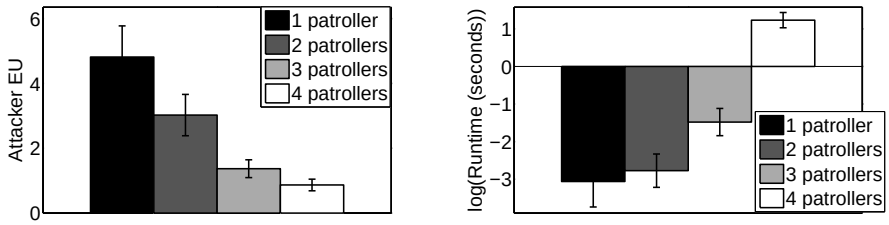


(a) (b)

Figure 4.17: Results for sampling under the example setting: (a) Frequency of each edge is chosen when the first sampling method based on Markov strategy is used. (b) Decomposed routes with highest probability superimposed on ferry schedules when the second sampling method based on decomposition is used.

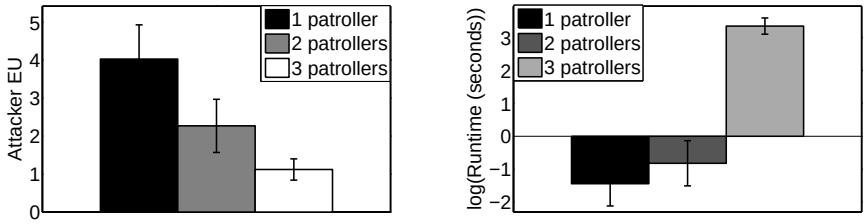
4.6.1.5 Number of Patrollers

Figure 4.18(a) shows the improvement in performance of CASS with increasing number of patrollers under discretization Level 1. The x-axis shows the number of patrollers and the y-axis indicates the average of attacker's maximal expected utility, i.e., the expected reward when he plays his best response. The results are averaged over 20 random utility settings of discretization Level 1. With fewer patrollers, the performance of the defender varies a lot depending on the randomized utility function (as indicated by standard deviation shown as the error bar). But the variance gets much smaller with more patrollers, which means the defender has sufficient resources for different instances. Figure 4.18(b) shows the run-time for CASS. The y-axis indicates the average of natural logarithm of runtime. Not surprisingly, the run-time increases when the number of patrollers increases.



(a) Solution quality of Level 1

(b) Runtime of Level 1



(c) Solution quality of Level 2

(d) Runtime of Level 2

Figure 4.18: Performance with varying number of patrollers.

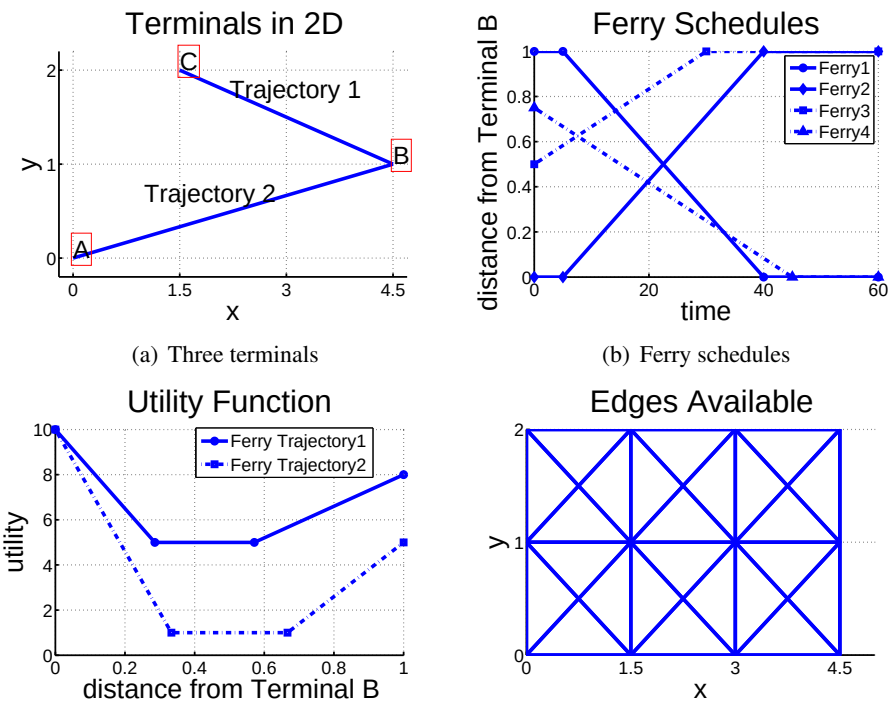
Figure 4.18(c) and 4.18(d) show the average performance and run-time of CASS with discretization Level 2, using the same set of utility settings as used in Level 1. Only results for 1 to 3 patrollers are shown. The program runs out of memory for four patrollers as there are $N^8M = 2734375$ flow distribution variables and at least $N^4M = 8757$ constraints. Note that the average solution quality of Level 2 is better than the result of Level 1 (e.g., the average attacker EU for 1 patroller is 4.81 in Level 1 and 4.13 in Level 2), which indicates a higher level of granularity can improve the solution quality. However, granularity clearly affect the ability to scale-up; which means that we need to consider the tradeoff between the solution quality and the memory used and one way to combat the scaling-up problem is to reduce the level of granularity. Nonetheless, the number of patrollers we have encountered in real-world scenarios such as at New York is of the order of 3 or 4, so CASS is capable at least for key real-world scenarios.

4.6.1.6 Approximation Approach for Multiple Defender Resources

We tested the first approximation approach for multiple defender resources described in Section 4.2.5 for the example setting. We used the **fmincon** function with interior-point method in MATLAB to minimize the non-linear objective function (Equation 4.27). Table 4.8 lists different run-time and the value of the objective function achieved given different iteration number (denoted as *MaxIter*). The function is not ensured to provide a feasible solution when the iteration number is not large enough, as shown in the first two rows. We compared the result with our LP formulation of DASS, which was implemented in MATLAB using **linprog** function. DASS can be solved within 8.032 seconds and provides an optimal solution AttEU$^m = 3.5$, this approximation approach is outperformed in both run-time efficiency and solution quality. This approach fails to provide a feasible solution efficiently and even when sufficient time is given (more than 400 times the run-time of the LP formulation), the maximum attacker expected utility is 18% larger than the optimal solution. This is mainly because the new formulation in the approximation approach is no longer linear or convex, making it difficult to find a global maximum.

4.6.2 Experiments for Two Dimensional Setting

The settings in 2-D space are more complex even with single patroller. Here we show an example setting motivated by the ferry system between Seattle, Bainbridge Island and Bremerton as shown in Figure 4.9. In this example setting, three terminals (denoted as A, B, and C) are non-collinear in the 2-D space as shown in the Figure 4.19(a). Ferry 1 and Ferry 2 are moving on the trajectory between Terminal B and C (denoted as Trajectory 1) and Ferry 3, and Ferry 4 are moving on the trajectory between Terminal B and A (denoted as Trajectory 2). The schedules of the four ferries are shown in Figure 4.19(b), where the x-axis is the time, and the y-axis is the distance from the



(a) Three terminals

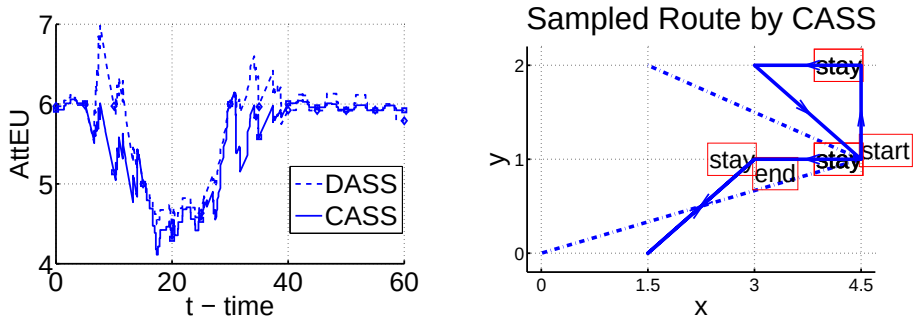
(b) Ferry schedules

(c) Utility function

(d) Available edges

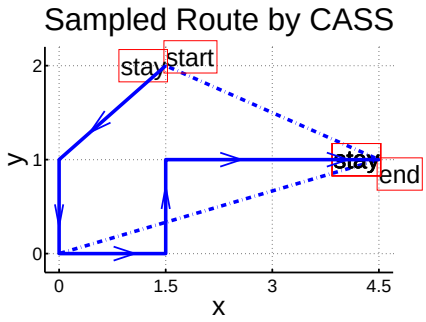
Figure 4.19: An example setting in two-dimensional space

common terminal B. Ferry 1 moves from C to B, Ferry 2 moves from B to C, Ferry 3 moves from B to A and Ferry 4 moves from A to B. Similar to the one-dimensional scenario in ferry domain, we assume the utility is decided by the ferry's position and the utility function is shown in Figure 4.19(c). The x-axis is the distance from the common terminal B and the y-axis is the utility for the two trajectories respectively. The 2-D space is discretized into a grid as shown in Figure 4.19(d) with $\delta x = 1.5$ and $\delta y = 1$ indicating the interval in the x-axis and y-axis. A patroller will be located at one of the intersection points of the grid graph at any discretized time points. The simulation time is 60 minutes and $M = 13$, i.e., $t_{k+1} - t_k = 5$ minutes. The speed limit for the patroller is $v_e = 0.38$ and all the available edges that a patroller can take during $[t_k, t_{k+1}]$ are shown in Figure 4.19(d). Only one patroller is involved. The protection radius is set to $r_e = 0.5$, and protection coefficient is $C_1 = 0.8$.



(a) Solution quality of DASSand CASS for Ferry 2

(b) Sampled route 1 superimposed on ferry trajectories



(c) Sampled route 2 superimposed on ferry trajectories

Figure 4.20: Experimental results under two-dimensional settings

Figure 4.20(a) compares the performance of DASS and CASS for Ferry 2. Ferry 2 is chosen because in both strategies, the attacker's best response is to attack Ferry 2. The x-axis is the time t, and the y-axis is the attacker expected utility of attacking Ferry 1 at time t. The maximum of AttEU of CASS is 6.1466, 12% lower compared to the result of DASS, which is 6.9817. Figure 4.20(b) and 4.20(c) show two sampled route given the strategy calculated by CASS on the 2-D map where the dashed lines represent for the ferry trajectories. The patroller starts from the node with text "start" and follows the arrowed route, and ends at the node with text "end" at the end of the patrol. She may stay at the nodes with text "stay". The patrol routes are shown in an intuitive way but can be ambiguous. The exact route should be listed as a table with time and position.

The routes are sampled based on the converted Markov strategy, and the total number of patrol routes that may be chosen with non-zero probability is 4.49×10^{10}.

4.7 Chapter Summary

This chapter makes several contributions in computing optimal strategies given moving targets and mobile patrollers. First, we introduce MRMT_{sg}, a novel Stackelberg game model that takes into consideration spatial and temporal continuity. In this model, targets move with fixed schedules and the attacker chooses his attacking time from a continuous time interval. Multiple mobile defender resources protect the targets within their protection radius, and bring in continuous space in our analysis. Second, we develop a fast solution approach, CASS, based on compact representation and sub-interval analysis. Compact representations dramatically reduce the number of variables in designing the optimal patrol strategy for the defender. Sub-interval analysis reveals the piece-wise linearity in attacker expected utility function and shows there is a finite set of dominating strategies for the attacker. Third, we propose two approaches for equilibrium refinement for CASS's solutions: route-adjust and flow-adjust. Route-adjust decomposes the patrol routes, greedily improves the routes and composes the new routes together to get the new defender strategy. Flow-adjust is a fast and simple algorithm that adjusts the flow distribution to achieve optimality in each time interval while keeping the marginal probability at the discretized time points unchanged. Additionally, we provide detailed experimental analyses in the ferry protection domain. CASS has been deployed by the US Coast Guard since April 2013.

There are several important avenues for future work. These include: (i) use a decreasing function to model the protection provided to the targets instead of using a fixed protection radius;

(ii) handle practical constraints on patrol boat schedule as not all are easily implementable; (iii) efficiently handle more complex and uncertain target schedules and utility functions.

Here we provide an initial discussion about the relaxation of the assumptions that we listed in Section 4.1 and used throughout the chapter:

- If we allow for complex and uncertain target schedules, we may model the problem as a game where the targets follow stochastic schedules. Our framework may still apply but may need to be enriched (e.g., using approaches such as the use of MDPs to represent defender strategies, see (Jiang, Yin, Zhang, Tambe, & Kraus, 2013)). Coordinating multiple such defenders then become an important challenge. It may be helpful in such cases to appeal to more of the prior work on multi-agent teamwork, given the significant uncertainty in such cases leading to more need for on-line coordination (Tambe, 1997; Stone, Kaminka, Kraus, & Rosenschein, 2010; Kumar & Zilberstein, 2010; Yin & Tambe, 2011).

- If we focus on environments where multiple attackers can coordinate their attacks, then we may need to further enhance our framework. Prior results from (Korzhyk, Conitzer, & Parr, 2011) over stationary targets and discrete time would be helpful in addressing this challenge, although the case of moving targets in continuous space and time in such cases provides a very significant challenge. Combining with the previous item for future work, a complex multiple defender multiple attacker scenarios would appear to be a very significant computational challenge.

Notation	Meaning
MRMT	The problem of multiple Mobile Resources protecting Moving Targets
MRMT_{sg}	Game model with a continuous set of strategies for the attacker for MRMT.
L	Number of ferries.
F_q	Ferry with index q.
A, B	Terminal points.
T	Continuous time interval or a finite set of time points.
D	Continuous space of possible locations or a set of distance points.
$S_q(t)$	Ferry schedule. Position of the target F_q at a specified time t.
W	Number of patrollers.
P_u	Patroller with index u.
v_m	Speed limit of patroller.
r_e	Protection radius of patroller.
C_G	Probability that the attacker can be stopped with G patrollers.
$U_q(t)$	Positive reward of a successful attack on target F_q at time t for the attacker.
M	Number of discretized time points.
N	Number of discretized distance points.
t_k	Discretized time point.
d_i	Discretized distance point.
δ_t	Distance between two adjacent time points.
R_u	Patrol route for patroller P_u. Under discretization of the defender's strategy space, R_u can be described as a vector.
$r_u(k)$	The patroller is located at $d_{r_u(k)}$ at time t_k.
$f(i,j,k)$	Flow distribution variable. Probability that the patroller moves from d_i to d_j during time $[t_k, t_{k+1}]$.
$p(i,k)$	Marginal distribution variable. Probability that the patroller is located at d_i t_k.
$E_{i,j,k}$	The directed edge linking nodes (t_k, d_i) and (t_{k+1}, d_j).
$p(R_u)$	Probability of taking route R_u.
$\text{AttEU}(F_q, t)$	Attacker expected utility of attacking target F_q at time t.
$\beta_q(t)$	Protection range of target F_q at time t
$\omega(F_q, t)$	Probability that the patroller is protecting target F_q at time t.
$I(i,q,k)$	Whether a patroller located at d_i at time t_k is protecting target F_q.
L_q^1,L_q^2	Lines of $S_q(t) \pm r_e$.
θ_{qk}^r	The rth intersection point in $[t_k, t_{k+1}]$ with respect to target F_q.
$\text{AttEU}(F_q, \theta_{qk}^{r\pm})$	Left/right-side limit of $\text{AttEU}(F_q, t)$ at θ_{qk}^r.
M_{qk}	Number of intersection points in $[t_k, t_{k+1}]$ with respect to target F_q.
$A_{qk}^r(i,j)$	C_1 if patroller taking edge $E_{i,j,k}$ can protect target F_q in $[\theta_{qk}^r, \theta_{qk}^{r+1}]$; 0 otherwise.
$E(u,k)$	Short for $E_{r_u(k),r_u(k+1),k}$.

Table 4.1: Summary of notations involved in the chapter.

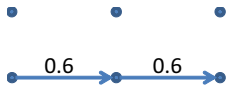
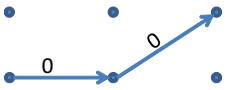
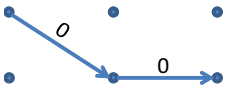
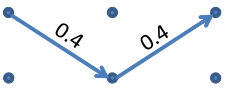
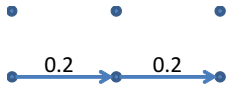
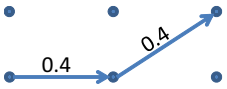
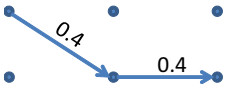
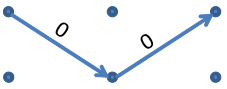
Full Representation 1			
$R_1 = (d_1, d_1, d_1)$	$R_2 = (d_1, d_1, d_2)$	$R_3 = (d_2, d_1, d_1)$	$R_4 = (d_2, d_1, d_2)$
			
Full Representation 2			
$R_1 = (d_1, d_1, d_1)$	$R_2 = (d_1, d_1, d_2)$	$R_3 = (d_2, d_1, d_1)$	$R_4 = (d_2, d_1, d_2)$
			

Table 4.2: Two full representations that can be mapped into the same compact representation shown in Figure 4.3.

R_u	$p(R_u)$ after decomposition	Adjusted Routes
$R_1 = (d_1, d_1, d_1)$	0.2	$(d_1, d_1, d_2) = R_2$
$R_2 = (d_1, d_1, d_2)$	0.4	$(d_1, d_1, d_2) = R_2$
$R_3 = (d_2, d_1, d_1)$	0.4	$(d_2, d_1, d_2) = R_4$

Table 4.3: Step (ii): Adjust each route greedily.

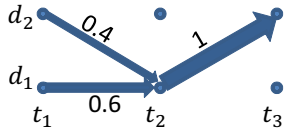
R_u	$p'(R_u)$ after adjustment		Composed Flow Distribution
$R_1 = (d_1, d_1, d_1)$	0		
$R_2 = (d_1, d_1, d_2)$	0.6		
$R_3 = (d_2, d_1, d_1)$	0		
$R_4 = (d_2, d_1, d_2)$	0.4		

Table 4.4: Step (iii): compose a new compact representation.

Level	δ_t (minutes)	M	δ_d	N
1	10	4	0.5	3
2	5	7	0.25	5
3	2.5	13	0.125	9
4	2	16	0.1	11

Table 4.5: Details about discretization levels. In the experiments mentioned in this section, the distance space is evenly discretized, parameterized by $\delta_d = d_{i+1} - d_i$.

Strategies	Worst Case Performance	Average Performance	Runtime (minutes)
CASS	3.82	3.40	-
Route-Adjust	3.82	2.88	8.96
Route-Adjust2	3.82	2.76	32.31
Flow-Adjust	3.82	3.34	0.50

Table 4.6: Comparison of different refinement approaches in terms of average performance and runtime. Only the runtime for the refinement process is calculated.

time interval $[t_k, t_{k+1}]$	maximum before	maximum after	time interval $[t_k, t_{k+1}]$	maximum before	maximum after
$[2, 4]$	3.7587	3.6675	$[16, 18]$	3.8111	3.7291
$[4, 6]$	3.8182	3.8182	$[18, 20]$	3.8182	3.8182
$[6, 8]$	3.8153	3.6164	$[20, 22]$	3.8182	3.8182
$[8, 10]$	3.8137	3.6316	$[22, 24]$	3.8182	3.8182
$[10, 12]$	3.8052	3.6316	$[24, 26]$	3.8182	3.8182
$[12, 14]$	3.8050	3.5664	$[26, 28]$	3.8182	3.8182
$[14, 16]$	3.7800	3.2100	$[28, 30]$	3.8182	3.8182

Table 4.7: The maximum of attacker's expected utility in each time interval decreases after flow-adjust is used.

$MaxIter$	$Run - time(sec)$	AttEU^m
3000	4.14	infeasible
10000	17.21	infeasible
900000	3298	4.0537

Table 4.8: Performance of approximation approach.

Chapter 5

Reasoning in Continuous Space

In addition to reasoning in continuous time, my thesis addresses the problem of reasoning in continuous space when protecting a large area. This work is motivated by the challenge in a "green security game" where policy-makers try to design patrol strategy for protecting forest areas from illegal extraction[1]. Illegal extraction of fuelwood or other natural resources from forests is a problem confronted by officials in many developing countries, with only partial success (MacKinnon, Mackinnnon, Child, & Thorsell, 1986; Dixon & Sherman, 1990; Clarke, Reed, & Shrestha, 1993; Robinson, 2008). To cite just two examples, Tanzania's Kibaha Ruvu Forest Reserves are "under constant pressure from the illegal production of charcoal to supply markets in nearby Dar es Salaam,"[2] and illegal logging is reported "decimating" the rosewood of Cambodia's Central Cardamom Protected Forest (see Fig. 5). In many cases, forest land covers a large area, which the local people may freely visit. Rather than protecting the forest by denying extractors entry to it, therefore, protective measures take the form of patrols throughout the forest, seeking to observe and hence deter illegal extraction activity (Lober, 1992; Sinclair & Arcese, 1995). With a limited budget, a patrol strategy will seek to distribute the patrols throughout the forest,

[1] The first author of the work in this chapter is Matthew P. Johnson.

[2] `http://www.tfcg.org/ruvu.html`

to minimize the resulting amount of extraction that occurs or to protect as much of the forest as possible. The extraction-preventing benefits of patrols are twofold: extraction is prevented directly, when catching would-be extractors in the act, and also indirectly, through deterrence.

Figure 5.1: "A truck loaded with illegally cut rosewood passes through Russey Chrum Village...in the Central Cardamom Protected Forest." Photo from (Boyle, 2011).

The problem setting to be addressed differs from those considered in previous works on security games, most crucially in that the forest protection setting is essentially continuous rather than discrete, both spatially and in terms of player actions. In the existing problems, there are a finite number of discrete locations to protect, whereas ideally the entire forest area would be protected from extraction.

To address this problem, I considered a Stackelberg game in which the defender publicly chooses a (mixed) patrol strategy in the form of patrol density distribution over the two-dimensional protected region, i.e. a probability distribution from which to select patrols; in response, the extractor then chooses whether or not to extract, or to what degree. Previous work in forest economics has provided an influential forest protection model (Albers, 2010) (see also (Robinson, Albers, & Williams, 2008, 2011)), in which there is a circular forest surrounded by

villages (hence potential extractors); the task is to distribute the patrols' probability density across the region of interest; the objective is to minimize the distance by which the extractors will trespass into the forest (since nearby villagers will extract as a function of this distance (HOFER, CAMPBELL, EAST, & HUISH, 2000)) and hence maximize the size of the resulting *pristine* forestland. The Stackelberg game I consider is a game-theoretic extension of this model, with additional features such as permitting spatial variation in patrol density, multiple patrol units, and convex polygon-shaped forests.

To reason about the attacker's strategy under this continuous setting, I provided a detailed benefit-cost analysis of the attacker. As the extractors go into the protected area for distance d, they incur a cost and gain a benefit *if not caught*, based on an increasing marginal cost function $c(d)$ and a decreasing marginal benefit function $b(d)$. (The instantaneous or *marginal* cost and benefit functions are the derivatives of the functions specifying the cumulative costs and benefits, respectively, of walking that far into the forest.) A given patrol strategy will reduce the extractor's expected benefit for an incursion of distance d from $b(d)$ to some value $b_p(d)$.

Based on this analysis, I proposed an efficient algorithm that calculates the optimal patrol strategy. The general idea of the optimal patrol strategy is to reduce the extractor's expected marginal benefit to a level that is equal to his expected marginal cost, i.e., $b_p(d) = c(d)$. Thus, the attacker has no incentive to going further and the area is effectively protected. I also provided a 1/2-approximation algorithm that calculates a ring patrol strategy, i.e., all patrol resources are distributed on a thin ring somewhere in the protected area.

Economists have studied the relationship generally between enforcement policy for protecting natural resources and the resulting incentives for neighbors of the protected area (Milliman, 1986; Robinson, 2008; Sanchirico & Wilen, 2001). Our point of departure in this chapter is the

influential forest protection model of (Albers, 2010) (see also (Robinson et al., 2008, 2011)), in which there is a circular forest surrounded by villages (hence potential extractors); the task is to distribute the patrols' probability density across the region of interest; the objective is to minimize the distance by which the extractors will trespass into the forest and hence (since nearby villagers will extract as a function of this distance (HOFER et al., 2000)) or maximize the size of the resulting *pristine* forestland.

We strengthen this model in several ways, permitting spatial variation in patrol density, multiple patrol units, and convex polygon-shaped forests. As has been observed (Albers, 2010), exogenous legal restrictions on patrol strategies, such as requiring *homogenous* patrols, can degrade protection performance (MacKinnon et al., 1986; Hall & Rodgers, 1992). Unlike the existing work on this model, we bring to bear algorithmic analysis on the problem. Specifically, we show that while certain such allocations can perform arbitrarily badly compared to the optimal, provably approximate or near-optimal allocations can be found efficiently.

5.1 Problem Setting

In this section we present the forest model of (Albers, 2010) and formulate a corresponding optimization problem. Villagers are distributed about the forest perimeter (see Fig. 5.2), which is initially assumed to be a circular region of radius 1, though we later extend to convex polygons. An extractor's action is to choose some distance d to walk into the forest, extracting on the return trip. We may assume, without loss of generality, that the extractor's route goes the chosen distance d towards the forest center (on a straight line), before reversing back to his starting point P on the perimeter. To see this, observe that all possible paths from P will sweep out a lens-like shape but,

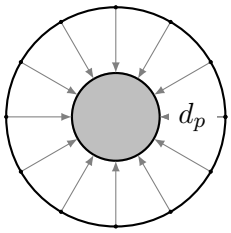


Figure 5.2: The forest, with the pristine area shaded.

since all points on the perimeter are possible starting points, the set of all trespass paths directed towards the center sweeps out the same area. Given our objective of maximizing *pristine* forest area, this holds true even if extractors are distributed around the perimeter *nonuniformly*, as long as there is a nonzero probability of villager presence at each point on the perimeter.

Due to symmetries and the fact that extractors' decisions are uncoordinated, the problem is essentially one-dimensional. Extractors incur a cost and gain a benefit *if not caught*, based on an increasing marginal cost function $c(d)$ and a decreasing marginal benefit function $b(d)$. (The instantaneous or *marginal* cost and benefit functions are the derivatives of the functions specifying the cumulative costs and benefits, respectively, of walking that far into the forest.) If caught, the extractor's benefit is 0 (the extracted resources are confiscated) but the cost is unchanged (the extractor's traveled distance does not change; there is no positive punishment beyond the confiscation itself and being prevented from engaging in further extraction while leaving the forest). Since extraction can be assumed to occur only on the return trip, and given the nature of the punishment, we may restrict our attention to detection on the return trip. Thus a given patrol strategy will reduce the extractor's expected benefit for an incursion of distance d from $b(d)$ to some value $b_p(d)$.

For a sufficiently fast-growing cost function relative to the benefit function, there will be a "natural core" of pristine forest even with no patrolling at all (Albers, 2010); that is, the optimal

trespass distance will be less than 1, since the marginal cost of extraction will eventually outweigh the marginal benefit, corresponding to the point at which the curves $b(d)$ and $c(d)$ intersect (see Fig. 5.3). The overall result of choosing a given patrol strategy, therefore, is to transform the benefit curve $b(d)$ into a lower benefit curve $b_p(d)$, thus reducing the extractor's optimal incursion distance (see Fig. 5.3). In the language of mathematical morphology (Soille, 2004), the pristine forest area P due to a given patrol strategy will be an *erosion* $P = F \ominus B$ of the forest F by a shape B, where B is a circle whose radius equals the trespass distance. The erosion is the locus of points reached by the center of B as it moves about inside of F.

Notation. $b(x), c(x), \phi(x)$ are the *marginal* benefit, cost, and capture probability functions, respectively. $B(x), C(x), \Phi(x)$ are the corresponding *cumulative* functions. d_p for $p \in \{n, o, r\}$ is the *trespass distance* under no patrols, the optimal patrol allocation, the best ring allocation, respectively. r_p is the radius of the pristine forest area under some patrol p. (Similarly, $b_p(x), B_p(x)$.) $d_n - d_p$ is the *reduction in trespass distance* under this patrol.

Definition 11. *Let $OPT(I)$ be the optimal solution value of a problem instance I, and let $ALG(I)$ be the solution value computed by a given algorithm. An algorithm for a maximization problem is a* c-approximation *(with $c < 1$) if, for every problem instance I, we have $ALG(I) \geq c \cdot OPT(I)$.*

The leader has a budget E specifying a bound on the total detection probability mass that can be distributed across the region. The task is to choose an allocation in order to minimize the extractor's resulting optimal trespass distance d_n, which is equivalent to maximizing the trespass distance reduction and *implies* maximizing the pristine radius. Note that our optimal and approximation algorithms both perform a binary search and thus incur an additive error ϵ.

5.1.1 Detection probability models

Let $\phi(x)$ be the detection probability density function chosen by the leader for the forest. An extractor is detected if he comes within some distance $\Delta << 1$ of the patrol. Under our time model, the patrol units move much less quickly than the extractors, and so patrols can be modeled as stationary from the extractor's point of view. Therefore, if e.g. $\phi(x)$ is constant (for a single patrol unit) over the region R (of size $|R|$), then the probability of detection for an extraction path of length d is proportional to ϕd, specifically $\phi d 2\Delta/|R|$, where the total area within distance Δ of the length-d walk is approximated as $d \cdot 2\Delta$. That is, probabilities are added rather than "multiplied" due to stationarity. (Here we assume the patrol unit is *not* visible to the extractor.) The model described here also covers settings in which the amount spent at a location determines the sensing range Δ there. For notational convenience, we drop Δ and $|R|$ throughout the chapter, assuming normalization as appropriate.

$\phi(x)$ influences the extractor's behavior in *two* ways. The rational extractor will trespass a distance into the forest that maximizes his total (or cumulative) net benefit, which is where his net marginal benefit $b(x) - c(x)$ equals zero. As the extractor moves about through a region with nonzero $\phi(x)$, his cost-benefit analysis is affected in two ways. First, the probability of reaching a given location x is reduced by the cumulative probability of capture up to that point, $\Phi(x)$, and so the net marginal benefit at point x is reduced from $b(x) - c(x)$ by amount $\Phi(x)b(x)$. (Recall that capture occurs on the return trip out of the forest, and so the cost $c(x)$ is paid *regardless of whether confiscation occurs*.) Second, being caught at point x is $\phi(x)$ means losing the full benefit accrued so far, which further reduces the net marginal benefit at this point by amount $\phi(x)B(x)$, where $B(x) = \int_{y=0}^{x} b(y)dy$ is the cumulative benefit.

We emphasize that the extractor's strategy (trespass distance) is chosen offline (in advance), based on the expected returns of each possible strategy. Note that the extractor acquires no new information online that can affect his decision-making: the strategy consists entirely of a distance by which to *attempt* to trespass; once caught, there is no further choice.

5.2 Patrol Allocations

Let the *patrol zone* be the region of the forest assigned nonzero patrol density. We note three patrol allocation strategies that have been proposed in the past:

- **Homogeneous:** Patrol density distributed uniformly over the entire region.
- **Boundary:** Patrol density distributed uniformly over a ring (of some negligible width) at the forest boundary.
- **Ring:** Patrol density distributed uniformly over a *some* ring (of negligible width w) concentric with the forest.

Boundary patrols can be superior to homogenous patrols since homogeneous patrols waste enforcement on the natural core (Albers, 2010). It is interesting to note that this is not always so. Suppose the homogenous-induced core radius is less than $1 - d$, w is very small, and the trip length d satisfies $w < 1/2 < d \leq 1$. With homogenous patrols, we will have $\Phi(d) = E/\pi \cdot d$. With boundary patrols, however, this probability for any $d \geq w$ will be $\frac{E}{\pi - \pi(1-w)^2} \cdot w = E/\pi \cdot \frac{w}{1-(1-w)^2}$, which approaches $\frac{E}{2\pi}$ as $w \to 0$. In this case, homogeneous patrols will outperform boundary patrols. Intuitively, this is because a patrol in the interior will "intersect" more trips

from boundary to the center than a patrol on the boundary will. Unfortunately, both boundary and homogeneous patrols can perform arbitrarily badly.

Proposition 2. *The approximation ratios of boundary and homogeneous patrols are both 0.*

(sketch). To see this, hold the budget fixed, and consider extremely large forests and cost and benefit functions yielding an empty natural core. The relationship between the cost/benefit functions and the budget be that an optimal patrol allocation will place patrols near to the forest center, halting the extractors at some distance r_o from the center but the significant dispersions of patrols due to either boundary or homogenous allocations would mean failing to stop the extractors prior to the forest center, resulting in an approximation factor of 0. □

Instead, our optimal patrol will be of the following sort:

- **Band:** The shape of the patrol zone is a band, i.e., the set difference of two circles,[3] both concentric with the forest.

The net cumulative benefit of walking distance x is $B_o(x) - C(x) = B(x) - \Phi(x)B(x) - C(x)$, where $\Phi(x)$ is the capture probability for this walk. Let $\phi(x) = d\Phi(x)/dx$ be the probability density function of the capture probability, which is proportional to patrol density. Then the probability density function corresponding to $B_o(x) - C(x)$ will be

$$\begin{aligned} d(B_o(x) - C(x))/dx &= dB(x)/dx - d\Phi(x)B(x)/dx - dC(x)/dx \\ &= (1 - \Phi(x)) \cdot b(x) - \phi(x)B(x) - c(x) \end{aligned} \tag{5.1}$$

Let band $[d_o, e)$ (with $0 \leq d_o \leq e \leq d_n$) be the patrol zone chosen by Algorithm 3.

[3] Generalizable to other forest shapes, as discussed below.

Algorithm 3 Computing the optimal allocation(b, c, E, ϵ)

$(d_1, d_2) \leftarrow (0, d_n)$
binary search: **while** $d_1 < d_2 - \epsilon/3$ *or* ϕ_2 *not set* **do**
  $d \leftarrow (d_1 + d_2)/2$
  $\phi(x) \triangleq \frac{b(x)-c(x)}{B(x)} - \frac{b(x)}{B^2(x)}(B(x) - C(x) - (B(d) - C(d)))$ $e \leftarrow x \; s.t. \; d \leq x \leq d_n$ and $\phi(x) = 0$
  $cost = \int_d^e 2\pi(1-x)\phi(x)dx$
  $\{d_2 \leftarrow d, \;\; \phi_2 \leftarrow \phi\}$ **if** $cost \leq E$ **else** $d_1 \leftarrow d$
end
return (d_2, ϕ_2)

Lemma 2. *Without loss of generality, the optimal density $\phi(x)$ at each point $x \in [d_o, e)$ can be assumed to be the smallest possible value disincentivizing further walking from x, i.e., that density yielding $b_o(x) = c(x)$. Moreover, $b_o(x) < c(x)$ and $\phi(x) = 0$ for $x > e$.*

Proof. Consider a function $\phi(\cdot)$ that successfully stops the extractor at some location d_o but which violates the stated property, at some particular level of discretization. That is, partition the interval $[d_o, d_n]$ into n equal sized subintervals, numbered $d_1, ..., d_n$. For this discretization, we write $B(d_i) = \sum_{j=1}^{i-1} b(j)$ and $\Phi(d_i) = \sum_{j=1}^{i-1} \phi(i)$ (omitting the coefficients). Let d_i be the first such subinterval for which $b_o(d_i) < c(x)$, and let d_i^+ be shorthand for $d_i + 1$. In this case (see Eq. 5.1) we have $(1 - \Phi(d_i))b(d_i) - \phi(d_i)B(d_i) - c(d_i) < 0$. We correct this by subtracting a value δ from $\phi(d_i)$ to bring about equality, and adding δ to $\phi(d_i^+)$.

The marginal net benefit of step d_i is then 0 (by construction), and that of step d_i^+ is only lower than it was before, so there is no immediate payoff to walking from d_i to d_i^+ or $d_i + 2$. Clearly $\Phi(d_i + 2)$ is unchanged. Finally, we verify that the expected total net benefit of walking

to position $d_i + 2$ is unchanged. This benefit is affected by the changes to both $\phi(d_i)$ and $\phi(d_i^+)$. First, $\delta B(d_i)$ is added to $b_o(d_i)$ by subtracting δ from $\phi(d_i)$; second, $b_o(d_i^+)$ becomes

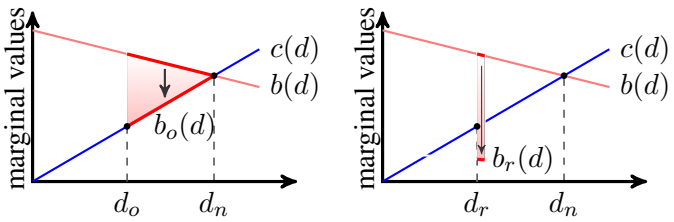
$$\begin{aligned}
& b_o(d_i^+)(1 - (\Phi(d_i^+) - \delta)) - (\phi(d_i^+) + \delta) \cdot B_o(d_i^+) \\
= \; & b_o(d_i^+)(1 - \Phi(d_i^+)) + b_o(d_i^+)\delta - \phi(d_i^+)B_o(d_i^+) - \delta B_o(d_i^+) \\
= \; & \big(b(d_i^+)(1 - \Phi(d_i^+)) - \phi(d_i^+)B_o(d_i^+)\big) + \big(b_o(d_i^+)\delta - \delta B_o(d_i^+)\big) \\
= \; & b_o(d_i^+) - \delta B(d_i)
\end{aligned}$$

Thus, since these two changes cancel out, and there was no incentive for walking from d_i *past* $d_i + 2$ prior to the modification, this remains true, and so the extractor will walk no farther than he did before the modification. We repeat this modification iteratively for all earliest adjacent violations (d_i, d_i^+), and for discretization precisions n. Since outer rings of circular (or, more generally, convex) forests have greater circumference, each such operation of moving patrol density forward only lowers the total cost of the patrol. $b_o(x) < c(x)$ and $\phi(x) = 0$ for $x > e$ follows from $\phi(\cdot)$ being a band that stops the extractor at position d_o. □

Lemma 3. *Without loss of generality, we may assume d_o kisses the outer edge of the patrol region.*

Proof. Clearly d_o will not be *prior to* the start of the patrol region. If d_o lay after the beginning of the patrol region, then, by Lemma 2, the solution would have its cost only lowered by shifting the earlier patrol density past d_o. □

Under the varying patrol density regime, the optimal patrol allocation can be computed (numerically). We remark that under the resulting patrol allocation, patrol density will decline monotonically with distance into the forest. Intuitively, the reason for this is that as distance into the



(a) Optimal patrol allocation. (b) Ring patrol.

Figure 5.3: The shaded regions correspond to the reduction in marginal benefits *within the patrol zone*. Not shown are the (less dramatic) effects on $b(\cdot)$ *following* the patrol zone, due to the cumulative capture probability.

forest grows, there is a smaller and smaller remaining net marginal benefit ($b(x) - c(x)$ that we need to compensate for by threat of confiscation, and yet the *magnitude* of the potential confiscation ($B(x)$) grows only larger.

Theorem 4. *Algorithm 3 produces a near-optimal allocation (i.e., with arbitrarily small error).*

Proof. We assume the properties stated by Lemma 2. Let d_o indeed be the optimal trespass distance. Observe that for $x < d_o$, $b_o(x) = b(x)$; for $x > e$, $b_o(x)$ is determined only by $b(x)$ and the cumulative capture probability, i.e., $b_o(x) = (1 - \Phi(x)) \cdot b(x)$. e is the point at which $\phi(x) = 0$ and $(1 - \Phi(x)) \cdot b(x) - c(x) = 0$. Now we compute $b_o(\cdot)$. Setting Eq. 5.1 to 0 yields:

$$\phi(x) = \frac{(1 - \Phi(x)) \cdot b(x) - c(x)}{B(x)} \tag{5.2}$$

The solution to this standard-form first-order differential equation (recall that $\Phi(x) = \int_{d_o}^{x} \phi(y)dy$, and note that Φ depends on the value d_o) is:

$$\Phi(x) = e^{-\int P(x)dx} \cdot \left(\int Q(x) \cdot e^{\int P(x)dx} dx + K) \right)$$

where $P(x) = \frac{b(x)}{B(x)}$, $Q(x) = \frac{b(x)-c(x)}{B(x)}$, and K is a constant. Since $\int P(x)dx = \int \frac{b(x)}{B(x)}dx = \ln B(x)$, we have $e^{\int P(x)dx} = e^{\ln B(x)} = B(x)$. Therefore

$$\begin{aligned}\int Q(x) \cdot e^{\int P(x)dx}dx &= \int \frac{b(x)-c(x)}{B(x)} \cdot B(x)dx \\ &= \int (b(x)-c(x))dx = B(x) - C(x)\end{aligned}$$

and, based on initial condition $\Phi(d_o) = 0$,

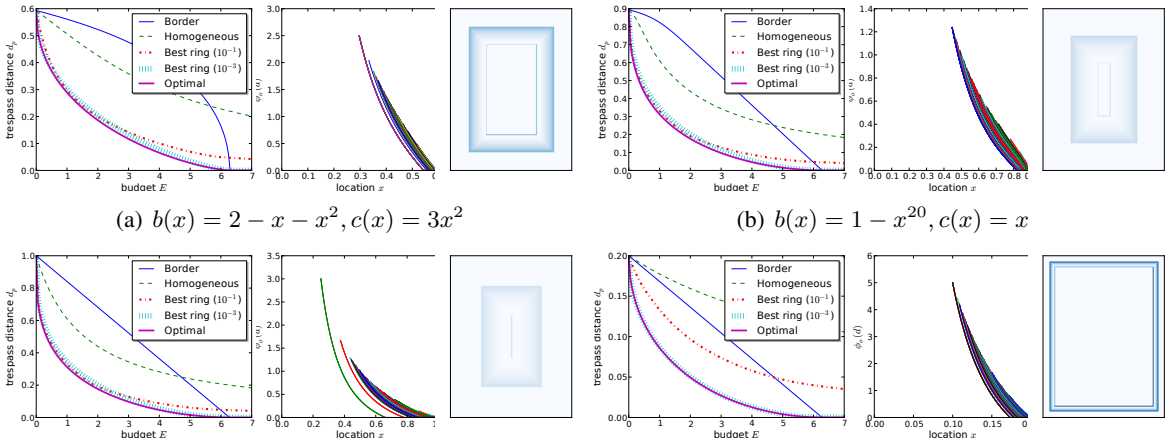
$$K = -\int Q(x) \cdot e^{\int P(x)dx}dx|_{d_o} = -(B(d_o) - C(d_o))$$

Since $\phi(x) = (\Phi(x))'$, this yields:

$$\begin{aligned}\Phi(x) &= \frac{B(x) - C(x) - (B(d_o) - C(d_o))}{B(x)} \\ \phi(x) &= \frac{b(x)-c(x)}{B(x)} - \frac{b(x)}{B^2(x)}(B(x) - C(x) - (B(d_o) - C(d_o))\end{aligned}$$

Then the optimal allocation for any given budget E will equal $\phi(x)$ for $x \in [d_o, d_n]$. The total cost of this is $E(d_o) = \int_{d_o}^{d*} 2\pi(1-x)\phi(x)dx$. If $b(x)$ and $c(x)$ are polynomial functions, then $\phi(x)$ is a rational function, and so $E(d_o)$ is solvable analytically, by the method of partial fractions. In this case, we can evaluate $E(d_o)$ in constant time (for fixed $b(x)$ and $c(x)$) in a real-number computation model. Alternatively, $E(d_o)$ can be approximated within additive error ϵ in time $O(1/\epsilon)$, using standard numerical integration methods.

We can compute the smallest d_o for which $E(d_o) \leq E$ by binary search. (e is also found by binary search, within error $\frac{\epsilon}{3}\frac{1}{2\pi\phi(0)}$, which is a constant; recall that $\phi(x)$ is a decreasing function.)



(a) $b(x) = 2 - x - x^2, c(x) = 3x^2$ (b) $b(x) = 1 - x^{20}, c(x) = x$

(c) $b(x) = 1, c(x) = x$ (d) $b(x) = 1, c(x) = 5x$

Figure 5.4: Patrol strategy effectiveness for sample $b(\cdot), c(\cdot)$ functions.

Algorithm 4 Computing the best ring patrol(b, c, E, ϵ)

$(d_1, d_2) \leftarrow (0, d_n)$

binary search: **while** $d_1 < d_2 - \epsilon$ *or* ϕ_2 *not set* **do**

  $d \leftarrow (d_1 + d_2)/2$ $\phi(d) \leftarrow E/(2\pi \cdot ((1-d) - w/2) \cdot w)$ $\Phi \leftarrow \phi \cdot w$ $e \leftarrow x$ *s.t.* $(1-\Phi)b(x) = c(x)$ $pos \leftarrow \int_d^e (1-\Phi)b(x) - c(x)dx$ $neg \leftarrow \Phi \cdot B(d)$ $\{d_2 \leftarrow d,\ \ \phi_2 \leftarrow \phi\}$ **if** $neg \geq pos$ **else** $d_1 \leftarrow d$

end

return (d_2, ϕ_2)

This yields a total running time of either $O(\log^2 1/\epsilon)$ or $O(1/\epsilon \log 1/\epsilon)$, depending on whether $E(d_o)$ is solved analytically or approximated. □

The varying-density allocation of Algorithm may be difficult or impractical to implement; moreover, each iteration of the loop requires an expensive iterative approximation parameterized by s, if $E(d_o)$ is not solvable analytically. Now we present a more efficient algorithm that produces easier-to-implement allocations. Assuming $b(\cdot)$ and $c(\cdot)$ can be integrated analytically and that their intersection can be found analytically, Algorithm 4 runs in time $O(\log 1/\epsilon)$.

Theorem 5. *Algorithm 4 produces a near-optimal ring patrol (i.e., within additive error at most ϵ).*

Proof. For a candidate trespass distance d, allocating the budget E to a width-w ring (where w is negligible) of radius d yields $\phi = E/(2\pi \cdot ((1-d) - w/2) \cdot w)$ and $\Phi = \phi \cdot w$. In order to discourage the extractor from passing point d, it must be the case that the expected cost of doing so (the potential loss due to confiscated, weighted by probability: $\Phi \cdot B(d)$) exceeds the expected benefit (the remaining net benefit, weighted by probability: $(1-\Phi)\int_d^{d_n} b(x) - c(x)dx$). We do binary search for the smallest such value d. □

Theorem 6. *Algorithm 4 provides a 1/2-approximation, both in trespass distance reduction and pristine radius.*

Proof. Let $r_n = 1 - d_n$ be the radius of the natural core. Let $r_o = 1 - d_o$ be the pristine area radius under the optimal patrol allocation $\phi_o(\cdot)$. We know that $\phi_o(\cdot)$ will be nonzero over the range $[d_o, d_n]$. Consider locations x within this range. As x grows from d_o to d_n, the marginal benefit $b(x)$ falls monotonically while $c(x)$ grows, and the cumulative benefit $B(x)$ and cumulative capture probability $\Phi(x)$ both grow monotonically. Thus by Eq. 5.2, $\phi(x)$ falls monotonically over $[d_o, d_n]$.

Now consider the radius $r_r = (r_o + r_n)/2$ and the corresponding location $d_r = 1 - r_r$, which divides the range $[d_o, d_n]$ into two halves. Because $\phi(x)$ is monotonic decreasing, $\phi(x)$ has at least as much total mass in the first half than in the second, i.e., $\int_{d_o}^{d_r} \phi(x)dx \geq \int_{d_r}^{d_n} \phi(x)dx$. Because the total cost of patrol density $\phi(x)$ at location x, rotated about the entire circle, is $2\pi\phi(x)$, "flattening" $\phi(x)$ over the range $[d_o, d_n]$ (i.e., setting it equal to $\frac{1}{d_n - d_o}\int_{d_o}^{d_n} \phi(x)dx$) will only lower the total cost. (Though doing so will sacrifice the guarantee of trespass distance d_o.) Then "compressing" this total probability mass $\int_{d_o}^{d_n} \phi(x)dx$ from the range $[d_o, d_n]$ to the point

d_r will not change the cost any further, since the mean circle circumference for radii in $[r_o, r_n]$ is $2\pi(r_o + r_n)/2$, which is the same as that for radius r_r.

We now claim that the constructed negligible-width ring patrol at d_r will deter the extractors from crossing it, by accounting for the two "halves" of $\phi_o(x)$. First, the "left" half of $\phi_o(x)$ transferred to d_r will yield a cumulative detect probability of $\Phi_o(d_r)$, just as under the optimal patrol. Second, the "right" half of $\phi_o(x)$ will inflict the same total reduction in net benefits for the action of traversing $[d_r, d_n]$ as the optimal patrol does. After passing d_r, each additional step would provide a positive net marginal benefit, until regaining the pre-d_r cumulative net benefit only at point d_n, after which all net marginal benefits are negative. Thus every stopping point after d_r will have cumulative net benefit lower than this value immediately before d_r.

We have constructed a ring patrol allocation that reduces the trespass distance by at least half the optimal such value, i.e., $(r_n - r_o)/2$, yielding pristine radius $r_r = (r_o + r_n)/2 \geq r_o/2$, and so the result follows. □

We note that the approximation ratio is tight. To see this, problem instances can be constructed satisfying the following: $c(x) = 0$ and $b(x)$ is constant (and small) over the interval $[d_o, d_n]$ (which meets an empty natural core, i.e. $d_n = 1$), and E is very small and hence $[d_o, d_n]$ is very narrow. In this case, $\Phi_i(x)$ grows very slowly over the patrol region, and $\phi_o(x)$ declines very slowly over it. In the extreme case, the weight of $\phi_o(x)$'s probability mass to the right of d_r approaches the weight to the left.

5.2.1 Algorithmic extensions

Multiple patrol units. We can extend from one to multiple patrol units, weighted equally or unequally. Given k patrol units, each given budget E_i (e.g., $1/k$) with $E = \sum E_i$, we partition

the forest into k sectors, each of angle $2\pi E_i/E$. We run one of our algorithms below, with budget E. Then we position patrol unit i at a location within sector i, chosen according to the computed $\phi(\cdot)$.

Other forest shapes. In the noncircular forest context, permitting extractors to traverse any length-bounded path from their starting points implies that the pristine area determined by a given patrol strategy will again be an erosion of the forest. Computing the erosion of an arbitrary shape is computationally intensive (Soille, 2004), but it is easily computable for convex polygons, which will approximate many realistic forests. In order to be practically implementable in such cases, the patrol should be symmetric around the forest area. Our algorithms above adapt easily to the setting of convex polygon forest shapes, where pristine areas are erosions, by integrating the cost of a patrol around the forest boundary. In both cases, we replace the circle circumferences $2\pi(1-x)$ with the cost of the corresponding polygon circumference. For large polygons with a reasonable number of sides, the resulting error due to corners will be insignificant.

5.3 Experiments

We implemented both our algorithms, as well as the baseline solutions of homogenous and boundary patrols. We tested these algorithms on certain realistic pairs of benefit and cost functions (with forest radius 1; see four examples in Fig. 5.2). We now summarize our observations on these results.

In each setting (see left subfigures), we vary the patrol budget, computing the patrol allocation function and hence the extractor's trespass distance d_p, for each. First, the optimal algorithm indeed dominates all the others. Both our algorithms perform much better overall than the two

baselines, however, up until the point at which the budget is sufficient to deter *any* entry into the forest, using boundary and best ring. Best ring will consider a ring at the boundary, so it cannot do worse than boundary, and so the two curves must intersect at zero. Prior to this best ring does outperform boundary. As observed above, neither homogeneous nor boundary consistently dominates the other.

We computed ring patrols for two ring widths, one very narrow ($1/10^4$) and one less so (0.1). Interestingly, neither ring size dominates the other. With a sufficiently large budget, the rings will lie on the boundary, but a wider ring will permit some nonnegligible trespass (part way across the ring itself). With smaller budgets the rings will lie in the interior of the forest. In this case, the narrow ring will spend the entire budget at one (expensive) density level, whereas the wider ring can (more cheaply, and hence more successfully) spend some of its budget at lower-density levels.

Next (see middle subfigures), we plot the optimal $\phi_o(\cdot)$ functions under many different budgets. As can be seen, these curves sweep out different regions of the plane, depending on the $b(\cdot), c(\cdot)$ pair.

Finally (see right subfigures), we illustrate the result of applying Algorithm 3 to a rectangular forest, with one sample budget (3.5, normalized to the dimensions of the forest). The patrol density is represented by the level of shading. The border of the natural core is also shown.

5.4 Chapter Summary

In this chapter, we have presented a Stackelberg security game setting that differs significantly from those previously considered in the AI literature, which necessitates the use of very different

techniques from those used in the past. At the same time, this work opens up an exciting new area of research for AI at the intersection of forest economics and game theory. Eventually, as with counterterrorism Stackelberg games studied in the literature, we aim to deploy our solutions in real-world settings. Potential sites for such deployments include Tanzania's aforementioned Kibaha Ruvu Forest Reserves and the mangrove forests of Mnazi Bay Ruvuma Estuary Marine Park.

Chapter 6

Reasoning with Frequent and Repeated Attacks

Another spatio-temporal aspect that is addressed in my thesis is reasoning with frequent and repeated attacks in domains such as protecting fisheries from over-fishing (Qian, Haskell, Jiang, & Tambe, 2014; Haskell et al., 2014a) and protecting rhinos and tigers from illegal poaching (Yang et al., 2014a). Poaching and illegal over-fishing are critical international problems leading to a destruction of ecosystems. For example, three out of nine tiger species have gone extinct in the past 100 years, and others are now endangered due to poaching (Secretariat, 2013). Law enforcement agencies in many countries are hence challenged with applying their limited resources to protecting endangered animals and fish stocks.

Building upon the success of applying SSGs to protect infrastructure including airports (Pita, Jain, Western, Portway, Tambe, Ordonez, Kraus, & Paruchuri, 2008), ports (Shieh, An, Yang, Tambe, Baldwin, DiRenzo, Maule, & Meyer, 2012b) and trains (Yin, Jiang, Johnson, Tambe, Kiekintveld, Leyton-Brown, Sandholm, & Sullivan, 2012a), researchers are now applying game theory to green security domains, e.g., protecting fisheries from over-fishing (Brown et al., 2014; Haskell et al., 2014b) and protecting wildlife from poaching (Yang et al., 2014b).

There are several key features in green security domains that introduce novel research challenges. First, the defender is faced with multiple adversaries who carry out repeated and frequent illegal activities (attacks), yielding a need to go beyond the one-shot SSG model. Second, in carrying out such frequent attacks, the attackers generally do not conduct extensive surveillance before performing an attack and spend less time and effort in each attack, and thus, it becomes more important to model the attackers' bounded rationality and bounded surveillance. Third, there is more attack data available in green security domains than in infrastructure security domains, which makes it possible to learn the attackers' decision-making model from data.

Previous work in green security domains (Yang et al., 2014b; Haskell et al., 2014b) models the problem as a game with multiple rounds and each round is an SSG (Yin et al., 2010) where the defender commits to a mixed strategy and the attackers respond to it. In addition, they address the bounded rationality of attackers using the SUQR model (Nguyen, Yang, Azaria, Kraus, & Tambe, 2013b). While such advances have allowed these works to be tested in the field, there are three key weaknesses in these efforts. First, the Stackelberg assumption in these works — that the defender's mixed strategy is fully observed by the attacker via extensive surveillance before each attack – can be unrealistic in green security domains as mentioned above. Indeed, the attacker may experience a delay in observing how the defender strategy may be changing over time, from round to round. Second, since the attacker may lag in observing the defender's strategy, it may be valuable for the defender to plan ahead; however these previous efforts do not engage in any planning and instead rely only on designing strategies for the current round. Third, while they do exploit the available attack data, they use Maximum Likelihood Estimation (MLE) to learn the parameters of the SUQR model for individual attackers which we show may lead to skewed results.

Figure 6.1: Snare poaching

In this chapter, we offer remedies for these limitations. First, we introduce a novel model called Green Security Games (GSGs). Generalizing the perfect Stackelberg assumption, GSGs assume that the attackers' understanding of the defender strategy may not be up-to-date and can be instead approximated as a convex combination of the defender strategies used in recent rounds. Previous models in green security domains, e.g., such as (Yang et al., 2014b; Haskell et al., 2014b) can be seen as a special case of GSGs, as they assume that the attackers always have up-to-date information, whereas GSGs allow for more generality and hence planning of defender strategies.

Second, we provide two algorithms that plan ahead — the generalization of the Stackelberg assumption introduces a need to plan ahead and take into account the effect of defender strategy on future attacker decisions. While the first algorithm plans a fixed number of steps ahead, the second one designs a short sequence of strategies for repeated execution.

Third, the chapter also provides a novel framework that incorporates learning of parameters in the attackers' bounded rationality model into the planning algorithms where, instead of using MLE as in past work, we use insights from Bayesian updating. All proposed algorithms are fully implemented and we provide detailed empirical results.

6.1 Motivation and Defining GSGs

Our motivating example assumes a perfectly rational attacker purely for simplicity of exposition. In the rest of the chapter, we consider attackers with bounded rationality.

Example 1. *Consider a ranger protecting a large area with rhinos. The area is divided into two subareas N_1 and N_2 of the same importance. The ranger chooses a subarea to guard every day and she can stop any snaring by poachers in the guarded area. The ranger has been using a uniform random strategy throughout last year. So for this January, she can choose to continue using the uniform strategy throughout the month, catching 50% of the snares. But now assume that the poachers change their strategy every two weeks based on the most recently observed ranger strategy. In this case, the ranger can catch 75% of the snares by always protecting N_1 in the first two weeks of January, and then switching to always protecting N_2: At the beginning of January, the poachers are indifferent between the two subareas due to their observation from last year. Thus, 50% of the snares will be placed in N_1 and the ranger can catch these snares in the first half of January by only protecting N_1. But after observing the change in ranger strategy, the poachers will switch to only putting the snares in N_2. The poachers' behavior change can be expected by the ranger and the ranger can catch 100% of the snares by only protecting N_2 starting from mid-January. (Of course the poachers must then be expected to adapt further).*

This example conceptually shows that the defender can benefit from planning strategy changes in green security domains. We now define GSG as an abstraction of the problem in green security domains (borrowing some terminology from Stackelberg Security Games (Yin et al., 2010)).

Definition 12. *A **GSG** is a $T(< \infty)$ round repeated game between a defender and L GSG attackers and (i) The defender has K guards to protect $N(\geq K)$ targets. (ii) Each round has multiple episodes and in every episode, each guard can protect one target and each attacker can attack one target. (iii) In round t, the defender chooses a mixed strategy at the beginning of the round, which is a probability distribution over all pure strategies, i.e., N choose K assignments*

from the guards to targets. In every episode, the guards are assigned to targets according to an assignment randomly sampled from the mixed strategy. (iv) Each target $i \in [N]$ has payoff values P_i^a, R_i^a, P_i^d, R_i^d ("P" for "Penalty", "R" for "Reward", "a" for "attacker" and "d" for "defender"). If an attacker attacks target i which is protected by a guard, the attacker gets utility P_i^a, and the defender gets R_i^d. If target i is not protected, the attacker gets utility R_i^a, and the defender gets P_i^d. $R_i^d > P_i^d$ and $R_i^a > P_i^a$. (v) The defender's utility in round t is the total expected utility calculated over all attackers.

Each round of the repeated game corresponds to a period of time, which can be a time interval (e.g., a month) after which the defender (e.g., warden) communicate with local guards to assign them a new strategy. We divide each round into multiple episodes for the players to take actions.

Consistent with previous work on green security games (Yang et al., 2014b; Haskell et al., 2014b), we divide the protected area into subareas or grid cells and treat each subarea or cell as a target. Different targets may have different importance to the defender and the attackers due to differences in resource richness and accessibility. We therefore associate each target $i \in [N]$ with payoff values. A mixed defender strategy can be represented compactly by a coverage vector $c = \langle c_i \rangle$ where $0 \leq c_i \leq 1$ is the probability that target i is covered by some guard and it satisfies $\sum_{i=1}^{N} c_i \leq K$ (Kiekintveld, Jain, Tsai, Pita, Ordonez, & Tambe, 2009b; Korzhyk, Conitzer, & Parr, 2010b). If an attacker attacks target i, the expected utility for the defender is $U_i^d(c) = c_i R_i^d + (1 - c_i) P_i^d$ given defender strategy c. We denote the mixed defender strategy in round t as c^t.

Definition 13. *A **GSG attacker** is characterized by his memory length Γ, coefficients $\alpha_0, ...\alpha_\Gamma$ and his parameter vector ω. In round t, A GSG attacker with memory length Γ responds to a*

convex combination of the defender strategy in recent $\Gamma + 1$ *rounds, i.e., he responds to* $\eta^t = \sum_{\tau=0}^{\Gamma} \alpha_\tau c^{t-\tau}$ *where* $\sum_{\tau=0}^{\Gamma} \alpha_\tau = 1$ *and* $c^t = c^0$ *if* $t \leq 0$*. In every episode of round* t*, a GSG attacker follows the SUQR model and chooses a random target to attack based on his parameter vector* ω *in the SUQR model.*

We aim to provide automated decision aid to defenders in green security domains who defend against human adversaries such as poachers who have no automated tools — hence we model the poachers as being boundedly rational and having bounded memory. We approximate a GSG attacker's belief of the defender's strategy in round t as a convex combination of the defender strategy in the current round and the last Γ rounds. This is because the attackers may not be capable of knowing the defender's exact strategy when attacking; naturally, they will consider the information they get from the past. Further, human beings have bounded memory, and the attackers may tend to rely on recent information instead of the whole history. The Stackelberg assumption in (Yang et al., 2014b; Haskell et al., 2014b) can be seen as a special case of this approximation with $\alpha_0 = 1$. In this chapter, we assume all attackers have the same memory length Γ, coefficients α_τ and these values are known to the defender. c^0 is the defender strategy used before the game starts and is known to players.

To model the bounded rationality of the human attackers such as poachers, we use the SUQR model, which has performed the best so far against human subjects in security games (Nguyen et al., 2013b). In this model, an attacker's choice is based on key properties of each target, including the coverage probability, the reward and the penalty, represented by the parameter

Notation	
T, N, K	# of rounds, targets and guards, respectively.
L, Γ	# of attackers and memory length of attackers.
c^t	Defender strategy in round t.
η^t	Attackers' belief of defender strategy in round t, which is a convex combination of c^t.
α_τ	Coefficient of $c^{t-\tau}$ when calculating η^t.
ω^l	Parameter vector of the SUQR model for attacker l. ω_1^l, ω_2^l and ω_3^l are the coefficient on c_i, R_i^a, P_i^a respectively in the SUQR model.
q_i	The probability of attacking target i.
E^t	Defender's expected utility in round t.

Table 6.1: Summary of key notations.

vector $\omega = (\omega_1, \omega_2, \omega_3)$. Given η as the attacker's belief (with η_i the belief of the coverage probability on target i), the probability that an attacker with parameter ω attacks target i is

$$q_i(\omega, \eta) = \frac{e^{\omega_1 \eta_i + \omega_2 R_i^a + \omega_3 P_i^a}}{\sum_j e^{\omega_1 \eta_j + \omega_2 R_j^a + \omega_3 P_j^a}} \tag{6.1}$$

Following the work of Yang et. al (Yang et al., 2014b), in this chapter, we assume the group of attackers may have heterogeneous weighting coefficients, i.e., each attacker $l \in [L]$ is associated with a parameter vector $\omega^l = (\omega_1^l, \omega_2^l, \omega_3^l)$.

A GSG defender strategy profile $[c]$ is defined as a sequence of defender strategies with length T, i.e., $[c] = \langle c^1, ..., c^T \rangle$. The defender's expected utility in round t is $E^t([c]) = \sum_l \sum_i q_i(\omega^l, \eta^t) U_i^d(c^t)$. The objective of the defender is to find the strategy profile with the highest average expected utility over all rounds, i.e., to maximize $E([c]) = \sum_{t=1}^T E^t([c])/T$.

Algorithm 5 Plan Ahead(ω, M)

Output: a defender strategy profile $[c]$ **for** t=1 *to* T **do**
  c^t = f-PlanAhead($c^{t-1}, \omega, \min\{T-t+1, M\}$)
end

6.2 Planning in GSGs

The defender can potentially improve her average expected utility by carefully planning changes in her strategy from round to round in a GSG. In this section, we consider the case where the attackers' parameter vectors $\omega^1, ..., \omega^L$, are known to the defender. For clarity of exposition, we will first focus on the case where $\alpha_0 = 0$ and $\Gamma = 1$. This is the special case when the attackers have one round memory and have no information about the defender strategy in the current round, i.e., the attackers respond to the defender strategy in the last round. We discuss the more general case in Section 6.4.

To maximize her average expected utility, the defender could optimize over all rounds simultaneously. However, this approach is computationally expensive when T is large: it needs to solve a non-convex optimization problem with NT variables (c_i^t) as the defender must consider attacker response, and the attacking probability has a non-convex form (see Equation 7.3). An alternative is a myopic strategy, i.e., the defender can always protect the targets with the highest expected utility in the current round. However, this myopic choice may lead to significant quality degradation as it ignores the impact of c^t in the next round.

Therefore, we propose an algorithm named PlanAhead-M (or PA-M) that looks ahead a few steps (see Algorithm 5). PA-M finds an optimal strategy for the current round as if it is the M^{th} last round of the game. If $M = 2$, the defender chooses a strategy assuming she will play a myopic strategy in the next round and end the game. When there are less than $M - 1$

$Target$	R_i^d	P_i^d
N_1	2	1
N_2	X	3

future rounds, the defender only needs to look ahead $T - t$ steps (Line 5). PA-T corresponds to the optimal solution and PA-1 is the myopic strategy. Unless otherwise specified, we choose $1 < M < T$. Function f-PlanAhead(c^{t-1}, ω, m) solves the following mathematical program (MP).

$$\max_{c^t, c^{t+1}, \ldots c^{t+m-1}} \sum_{\tau=0}^{m-1} E^{t+\tau} \tag{6.2}$$

$$s.t \quad E^\tau = \sum_l \sum_i q_i(\omega^l, \eta^\tau) U_i^d(c^\tau), \tau = t, .., t+m-1 \tag{6.3}$$

$$\eta^\tau = c^{\tau-1}, \tau = t, .., t+m-1 \tag{6.4}$$

$$\sum_i c_i^\tau \leq K, \tau = t, .., t+m-1 \tag{6.5}$$

This is a non-convex problem when $m > 0$ and can be solved approximately with local search approaches.

Although we show in the experiment section that PA-2 can provide a significant improvement over baseline approaches in most cases, there exist settings where PA-2 can *perform arbitrarily badly* when compared to the optimal solution. The intuition is that the defender might make a suboptimal choice in the current round with an expectation to get a high reward in the next round; however, when she moves to the next round, she plans for two rounds again, and as a result, she never gets a high reward until the last round.

Example 2. *Consider a guard protecting two subareas with payoff values shown on the right ($X \gg 1$). For simplicity of the example, assume the defender can only choose pure strategies. There is one poacher with a large negative coefficient on coverage probability, i.e., the poacher*

Algorithm 6 Fixed Sequence

Output: defender strategy profile $[c]$ $(a^1, ..., a^M) =$ f-FixedSequence(ω, M). **for** t=1 *to* T **do**
| $c^t = a^{(t \bmod M)+1}$
end

will always snare in the subarea that is not protected in the last round. The initial defender strategy is protecting N_1, meaning the attacker will snare in N_2 in round 1. According to PA-2, the defender will protect N_1 in round 1 and plan to protect N_2 in round 2, expecting a total utility of $3 + X$. However, in round 2, the defender chooses N_1 again as she assumes the game ends after round 3. Thus, her average expected utility is $\frac{3(T-1)+X}{T} \approx 3$. On the other hand, if the defender alternates between N_1 and N_2, she gets a total utility of $X + 2$ for two consecutive rounds, and her average utility is at least $\frac{X}{2} \gg 3$.

PA-2 fails in such cases because it over-estimates the utility in the future. To remedy this, we generalize PA-M to PA-M-γ by introducing a discount factor $0 < \gamma \leq 1$ for future rounds when $T - t < M - 1$, i.e., substituting Equation 6.2 with

$$\max_{c^t, c^{t+1}, ... c^{t+m-1}} \sum_{\tau=0}^{m-1} \gamma^\tau E^{t+\tau} \tag{6.6}$$

While PA-M-γ presents an effective way to design sequential defender strategies, we provide another algorithm called FixedSequence-M (FS-M) for GSGs (see Algorithm 6). FS-M not only has provable theoretical guarantees, but may also ease the implementation in practice. The idea of FS-M is to find a short sequence of strategies with fixed length M and require the defender to execute this sequence repeatedly. If $M = 2$, the defender will alternate between two strategies and she can exploit the attackers' delayed response. It can be easier to communicate with local guards to implement FS-M in green security domains as the guards only need to alternate between

several types of maneuvers. Function f-FixedSequence(ω, M) calculates the best fixed sequence of length M through the following MP.

$$\max_{a^1,...,a^M} \sum_{t=1}^{M} E^t \tag{6.7}$$

$$s.t \quad E^t = \sum_l \sum_i q_i(\omega^l, \eta^t) U_i^d(a^t), t = 1, ..., M \tag{6.8}$$

$$\eta^1 = a^M \tag{6.9}$$

$$\eta^t = a^{t-1}, t = 2, ..., M \tag{6.10}$$

$$\sum_i a_i^t \leq K, t = 1, .., M \tag{6.11}$$

Theorem 7 shows that the solution to this MP provides a good approximation of the optimal defender strategy profile.

Theorem 7. *In a GSG with T rounds, $\alpha_0 = 0$ and $\Gamma = 1$, for any fixed length $1 < M \leq T$, there exists a cyclic defender strategy profile $[s]$ with period M that is a $(1 - \frac{1}{M})\frac{Z-1}{Z+1}$ approximation of the optimal strategy profile in terms of the normalized utility, where $Z = \lceil \frac{T}{M} \rceil$.*

The intuition is to divide the optimal sequence into sections with length $M - 1$ and bound the defender's expected utility in each section.

Definition 14. *A cyclic defender strategy profile for a GSG is a profile consisting of a cyclic sequence of strategies, i.e., $\exists \bar{T}$, such that $\forall t > \bar{T}$, $c^t = c^{t-\bar{T}}$, $\bar{T}$ is denoted as the period of the strategy profile.*

Proof of Theorem 7: Use $U(x^1, x^2)$ to denote the defender's *normalized* expected utility in a round where defender strategy x^2 is used in this round and defender strategy x^1 is used in the

previous round. Then $0 \leq U(x^1, x^2) \leq 1$. For the optimal defender strategy profile $[c]$, denote the normalized utility as U^{opt}.

$\langle b^1, ..., b^M \rangle$ is a strategy sequence whose average normalized expected utility for the last $M-1$ rounds, i.e., $U_b = \frac{\sum_{t=2}^{M} U(b^{t-1}, b^t)}{M-1}$, is maximized. $\langle a^1, ..., a^M \rangle$ is a strategy sequence such that the average normalized expected utility of the sequence when it forms a cycle, i.e., $U_a = \frac{U(a^M, a^1) + \sum_{t=2}^{M} U(a^{t-1}, a^t)}{M}$, is maximized. Then

$$\begin{aligned} M * U_a &= U(a^M, a^1) + \sum_{t=2}^{M} U(a^{t-1}, a^t) \\ &\geq U(b^M, b^1) + \sum_{t=2}^{M} U(b^{t-1}, b^t) \\ &\geq \sum_{t=2}^{M} U(b^{t-1}, b^t) \\ &= (M-1) * U_b \end{aligned}$$

Let $Z = \lceil \frac{T}{M} \rceil$. Construct a cyclic defender strategy profile $[s]$ by repeating the strategy sequence $\langle a^1, ..., a^M \rangle$. Then

$$T * U([s]) = U(c^0, s^1) + \sum_{t=2}^{T} U(s^{t-1}, s^t) \tag{6.12}$$

$$\geq (Z-1) * M * U_a \tag{6.13}$$

$$\geq (Z-1) * (M-1) * U_b \tag{6.14}$$

Strategy profile $[s]$ contains $Z-1$ complete cycles (starting with a^2) with an average normalized utility U_a. The first inequality is derived by ignoring the first round and the last incomplete cycle when $mod(T, M) \neq 1$.

On the other hand, for the optimal defender strategy profile $[c] = [c]^{opt}$, we know that for any consecutive sequence of length M, the average normalized utility of last $M-1$ rounds can be no more than U_b. So we divide the strategy profile into $\lceil \frac{T}{M-1} \rceil$ pieces, each piece with length $M-1$ except the last piece. Then for each piece, the sum of normalized utility is no more than $U_b * (M-1)$. Otherwise, if the sum of normalized utility of the i^{th} piece is higher than $U_b * (M-1)$, then the strategy sequence $< c^{(i-1)(M-1)}, ..., c^{i(M-1)} >$ contradicts the optimality of $< b^1, ..., b^M >$. Thus,

$$\begin{aligned} T * U^{opt} &= U(c^0, c^1) + \sum_{t=2}^{T} U(c^{t-1}, c^t) && (6.15) \\ &\leq U_b * (M-1) * \lceil \frac{T}{M-1} \rceil && (6.16) \\ &\leq (T + M - 1) * U_b && (6.17) \end{aligned}$$

The last inequality is yield by conceptually completing the last piece. Combining these results, we get

$$\begin{aligned} \frac{U([s])}{U^{opt}} &\geq \frac{(Z-1) * (M-1)}{T + M - 1} \\ &\geq \frac{(Z-1) * (M-1)}{Z * M + M} \\ &= (1 - \frac{1}{M}) * \frac{Z-1}{Z+1} \end{aligned}$$

So $[s]$ is a $(1 - \frac{1}{M})\frac{Z-1}{Z+1}$ approximation of the optimal strategy profile in terms of the normalized utility. □

According to Theorem 7, when the game has many rounds ($T \gg M$), the cyclic sequence constructed by repeating $a^1, ...a^M$ is a $1 - 1/M$ approximation. While in experiments this non-convex MP is solved approximately, with a large number of random restarts, we may be able to achieve this $1 - 1/M$ approximation.

According to Theorem 7, when a GSG has many rounds ($T \gg M$), the cyclic sequence constructed by repeating $a^1, ...a^M$ is a $1 - 1/M$ approximation.

6.3 Learning and Planning in GSGs

In Section 6.2, we assume that the parameter vectors $\omega^1, ..., \omega^L$ in the attackers' bounded rationality model are known. Since the defender may not know these parameter values precisely at the beginning of the game in practice, we now aim to learn the attackers' average parameter distribution from attack data. Previous work in green security domains (Yang et al., 2014b; Haskell et al., 2014b) treats each data point, e.g., each snare or fishnet, as an independent attacker and applies MLE to select the most probable parameter vector. However, some of the assumptions made in previous work in proposing MLE may not always hold as MLE works well when a large number of data samples are used to estimate one set of parameters (Eliason, 1993). Here we show that estimating ω from a single data point using MLE can lead to highly biased results.

Example 3. *Consider a guard protecting two targets in round* 1. *The payoff structure and initial defender strategy are shown in Table 6.2 where* $X \gg 1$ *and* $0 < \delta \ll 1$. *An attacker with parameter vector* $\omega = (-1, 0, 0)$ *will choose* N_1 *or* N_2 *with the probability* ≈ 0.5, *as* $\omega_1 = -1$ *means he has a slight preference on targets with lower coverage probability (see Equation 7.3). If the attacker attacks* N_1, *applying MLE will lead to an estimation of* $\omega = (+\infty, \cdot, \cdot)$, *meaning*

the attacker will always choose the target with higher coverage probability. This is because the probability of attacking N_1 is 1 given $\omega_1 = +\infty$, which is higher than that of any other parameter value. Similarly, if the attacker attacks N_2, an extreme parameter of $(-\infty, \cdot, \cdot)$ is derived from MLE. These extreme parameters will mislead the defender in designing her strategy in the following round.

Target	R_i^d	P_i^d	R_i^a	P_i^a	c_i^0
N_1	1	-1	1	-1	$0.5 + \delta$
N_2	1	$-X$	1	-1	$0.5 - \delta$

Table 6.2: Payoff structure of Example 3.

We, therefore, leverage insights from Bayesian Updating. For each data point, we estimate a probability distribution over parameter values instead of selecting the ω vector that maximizes the likelihood of the outcome. This approach is also different from maximum a posteriori probability (MAP) because MAP still provides single value estimates, whereas Bayesian Updating uses distributions to summarize data.

Algorithm 7 describes the learning algorithm for one round of the game. Rather than learning single parameter values, one from each attack, we learn a probability distribution. The input of the algorithm includes the number of attacks χ_i found on each target $i \in [N]$, the attackers' belief of the defender strategy η, and the prior distribution $p = \langle p_1, ..., p_S \rangle$ over a discrete set of parameter values $\{\hat{\omega}\} = \{\hat{\omega}^1, ..., \hat{\omega}^S\}$, each of which is a 3-element tuple. If an attacker attacks target i, we can calculate the posterior distribution of this attacker's parameter by applying Bayes' rule based on the prior distribution p (Line 7). We then calculate the average posterior distribution $\bar{p}$ over all attackers (Line 7).

Based on Algorithm 7, we now provide a novel framework that incorporates the learning algorithm into PA-M(-γ) as shown in Algorithm 8. The input p^1 is the prior distribution of the

Algorithm 7 Learn-BU (η, χ,$\{\hat{\omega}\}$, p)

Output: $\bar{p}$ – a probability distribution over $\{\hat{\omega}\} = \{\hat{\omega}^1, ..., \hat{\omega}^S\}$. **for** i=1 *to* N **do**
  for s=1 *to* S **do**
    $\bar{p}_i(s) = \frac{p(s)q_i(\hat{\omega}^s,\eta)}{\sum_r p(r)q_i(\hat{\omega}^r,\eta)}$
  end
end
for s=1 *to* S **do**
  $\bar{p}(s) = \frac{\sum_i \chi_i \bar{p}_i(s)}{\sum_i \chi_i}$
end

Algorithm 8 BU-PA-M-$\gamma(p^1)$

Output: Defender strategy profile $\langle c^1, ...c^T \rangle$. **for** t=1 *to* T **do**
  c^t = f-PlanAhead$(c^{t-1}, \omega, \min\{T-t, M-1\})$ $\bar{p}^t$ = Learn-BU$(c^{t-1}, \chi^t, \{\hat{\omega}\}, p^t)$ $p^{t+1} = \bar{p}^t$
end

attackers' parameters before the game starts. This prior distribution is for the general population of attackers and we need to learn the distribution of the L attackers we are facing in one game. The main idea of the algorithm is to use the average posterior distribution calculated in round t (denoted as $\bar{p}^t$) as the prior distribution in round $t+1$ (denoted as p^{t+1}), i.e.,$p^{t+1} = \bar{p}^t$. Given prior p^t, Function f-PlanAhead in Line 8 is calculated through Equation 6.2 – 6.5 by substituting Equation 6.3 with $E^t = L \sum_s \sum_i p^t(s) q_i(\bar{\omega}^s, c^{t-1}) U_i^d(c^t)$. Note that there was no probability term in Equation 6.3 because there we know exactly the parameter values of the attackers. After we collect data in round t, we apply Learn-BU (Algorithm 7) again and update the prior for next round (Line 8). This is a simplification of the more rigorous process which enumerates the matchings (exponentially many) between the data points and attackers and updates the distribution of each attacker separately when the attack data is anonymous (the guard may only find the snares placed on ground without knowing the identity of the poacher).

When incorporating Algorithm 7 into FS-M, we divide the game into several stages, each containing more than M rounds, and only update the parameter distribution at the end of each

stage. As FS-M may not achieve its average expected utility if only a part of the sequence is executed, updating the parameter distribution in every round may lead to a low utility.

6.4 General Case

Generalization from $\Gamma = 1$ and $\alpha_0 = 0$ to $\Gamma > 1$ and/or $\alpha_0 \in [0, 1]$ can be achieved via generalizing η^t. PA-M(-γ) can be calculated by substituting Constraint 6.4 with $\eta^\tau = \sum_{k=0}^{M} \alpha_k c^{\tau-k}$, and FS-M can be calculated by changing Constraints 6.9-6.10 accordingly. Theorem 8 shows the theoretical bound of FS-M with $\Gamma > 1$ and the proof is similar to that of Theorem 7.

Theorem 8. *In a GSG with T rounds, for any fixed length $\Gamma < M \leq T$, there exists a cyclic defender strategy profile $[s]$ with period M that is a $(1 - \frac{\Gamma}{M})\frac{Z-1}{Z+1}$ approximation of the optimal strategy profile in terms of the normalized utility, where $Z = \lceil \frac{T-\Gamma+1}{M} \rceil$.*

Proof of Theorem 8: Use $U([x], x_0)$ to denote the defender's normalized expected reward in a round where defender strategy x_0 is used in this round, and defender strategy sequence $[x] = \langle x_{-\Gamma}, ..., x_{-1} \rangle$ is used in the previous Γ rounds. Then $0 \leq U([x], x_0) \leq 1$. For the optimal defender strategy profile $[c]$, denote the normalized utility as U^{opt}.

$\langle b^1, ..., b^M \rangle$ is a strategy sequence whose average normalized expected utility for last $M - \Gamma$ rounds, is maximized and the value is denoted as U_b. $\langle a^1, ..., a^M \rangle$ is a strategy sequence such that the average normalized expected utility of the sequence when it forms a cycle is maximized and the value is denoted as U_a. Then

$$\begin{aligned} M * U_a &\geq \sum_{t=\Gamma+1}^{M} U(b^{t-1}, b^t) \\ &= (M - \Gamma) * U_b \end{aligned}$$

Construct a defender strategy profile $[s]$ by repeating the strategy sequence $\langle a^1, ..., a^M \rangle$. Then

$$\begin{aligned} T * U([s]) &\geq (Z-1) * M * U_a && (6.18) \\ &\geq (Z-1) * (M-\Gamma) * U_b && (6.19) \end{aligned}$$

Strategy profile $[s]$ contains $\lfloor \frac{T-\Gamma}{M} \rfloor$ complete cycles (starting from the first round with a^Γ) with average normalized reward U_a. As $Z = \lceil \frac{T-\Gamma+1}{M} \rceil$, $\lfloor \frac{T-\Gamma}{M} \rfloor = Z-1$. The inequality 6.18 is derived by ignoring the first round and the last incomplete cycle if any (when $mod(T, M) \neq \Gamma$).

On the other hand, for the optimal defender strategy profile $[c] = [c]^{opt}$, we know that for any consecutive sequence of length M, the average normalized reward of last $M - \Gamma$ rounds can be no more than U_b. So we divide the strategy profile into $\lceil \frac{T}{M-\Gamma} \rceil$ pieces, each piece with length $M - \Gamma$ except the last piece. Then for each piece, the sum of normalized reward is no more than $U_b * (M - \Gamma)$. Thus,

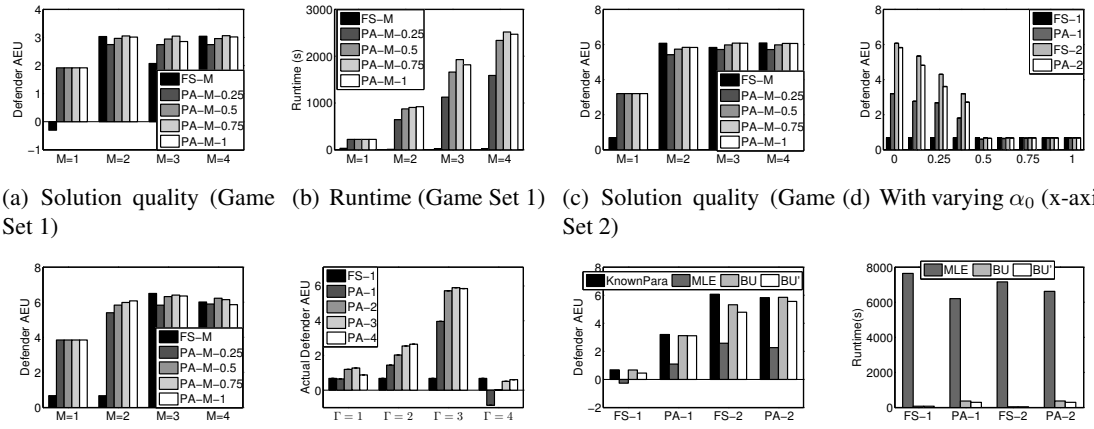
$$\begin{aligned} T * U^{opt} &\leq U_b * (M-\Gamma) * \lceil \frac{T}{M-\Gamma} \rceil && (6.20) \\ &\leq (T + M - \Gamma) * U_b && (6.21) \end{aligned}$$

The inequality 6.21 is yield by conceptually completing the last piece. Combine 6.18 - 6.21, we get

$$\begin{aligned} \frac{U([s])}{U^{opt}} &\geq \frac{(Z-1) * (M-\Gamma)}{T + M - \Gamma} \\ &\geq \frac{(Z-1) * (M-\Gamma)}{M + Z * M} \\ &= (1 - \frac{\Gamma}{M}) * \frac{Z-1}{Z+1} \end{aligned}$$

Equation is derived from the definition of Z, as $T - \Gamma \leq Z * M - 1 \leq Z * M$. So the cyclic strategy profile $[s]$ is a $(1 - \frac{\Gamma}{M})\frac{Z-1}{Z+1}$ approximation of the optimal strategy profile in terms of normalized utility. □

6.5 Experimental Results



(a) Solution quality (Game Set 1) (b) Runtime (Game Set 1) (c) Solution quality (Game Set 2) (d) With varying α_0 (x-axis)

(e) Solution quality ($\Gamma = 2$) (f) Estimated $\Gamma = 3$ (g) Solution quality for learning (h) Runtime for learning

Figure 6.2: Experimental results show improvements over algorithms from previous work.

We test all the proposed algorithms on GSGs motivated by scenarios in green security domains such as defending against poaching and illegal fishing. Each round corresponds to 30 days, and each poacher/fisherman will choose a target to place snares/fishnets every day. All algorithms are implemented in MATLAB with the $fmincon$ function used for solving MPs and tested on 2.4GHz CPU with 128 GB memory. All key differences noted are statistically significant ($p < 0.05$).

6.5.1 Planning Algorithms

We compare proposed planning algorithms PA-M(-γ) and FS-M with baseline approaches FS-1 and PA-1. FS-1 is equivalent to calculating the defender strategy with a perfect Stackelberg assumption, which is used in previous work (Yang et al., 2014b; Haskell et al., 2014b), as the defender uses the same strategy in every round and the attackers' belief coincides with the defender strategy. PA-1 is the myopic strategy which tries to maximize the defender's expected utility in the current round. We assume c_0 is the MAXIMIN strategy.

We first consider the special case ($\alpha_0 = 0$, $\Gamma = 1$) and test on 32 game instances of 5 attackers, three targets, one guard and 100 rounds with random reward and penalty chosen from $[0, 10]$ and $[-10, 0]$ respectively (denoted as Game Set 1). We run 100 restarts for each MP. Figure 6.2(a) shows that PA-M(-γ) and FS-M significantly outperform FS-1 and PA-1 in terms of the defender's average expected utility (AEU). This means using the perfect Stackelberg assumption would be detrimental to the defender if the attackers respond to last round's strategy. For PA-M, adding a discount factor γ may improve the solution. Figure 6.2(b) shows FS-M takes much less time than PA-M overall as FS-M only needs to solve one MP throughout a game while PA-M solves an MP for each round.

We also test on 32 games with 100 attackers, ten targets, four guards and 100 rounds (denoted as Game Set 2) in the special case (see Figure 6.2(c)). We set a 1-hour runtime limit for the algorithms and again, FS-M and PA-M(-γ) significantly outperform FS-1 and PA-1 in solution quality.

We then test general cases on Game Set 2. Figure 6.2(d) shows the defender's AEU with varying α_0 when $\Gamma = 1$. In the extreme case of $\alpha_0 = 1$, i.e., the attackers have perfect knowledge

of the current defender strategy, the problem reduces to a repeated Stackelberg game, and all approaches provide similar solution quality. However, when $\alpha_0 < 0.5$, FS-2 and PA-2 provide a significant improvement over FS-1 and PA-1, indicating the importance of planning.

We further test the robustness of FS-2 when there is slight deviation in α_0 with $\Gamma = 1$ (see Figure 6.3). For example, the value of 5.891 in the 2^{nd} row, 1^{st} column of the table is the defender's AEU when the actual $\alpha_0 = 0$ and the defender assumes (estimates) it to be 0.125 when calculating her strategies. Cells in the diagonal show the case when the estimation is accurate. Cells in the last row show result for baseline algorithm FS-1. FS-1 uses the Stackelberg assumption, and thus, the estimated value makes no difference. When the actual value slightly deviates from the defender's estimate (cells adjacent to the diagonal ones in the same column), the solution quality does not change much if the actual $\alpha_0 > 0.5$. When the actual $\alpha_0 < 0.5$, FS-2 outperforms FS-1 significantly even given the slight deviation.

In Figure 6.2(e), we compare algorithms assuming $\Gamma = 2$, $\alpha_1 = \alpha_2 = 0.5$ and $\alpha_0 = 0$. As expected, PA-M with $M > 1$ and FS-M with $M > 2$ significantly outperforms FS-1 and PA-1. The improvement of FS-2 over FS-1 is negligible, as any fixed sequence of length 2 can be exploited by the attackers with memory length $= 2$.

Figure 6.2(f) shows the solution quality of PA-M when the defender assumes the attackers' memory length is 3, but the actual value of Γ varies from 1 to 4. When Γ is slightly over-estimated (actual $\Gamma = 1$ or 2), PA-M still significantly outperforms the baseline algorithm FS-1 and PA-1. However, when Γ is under-estimated (actual $\Gamma = 4$), the attackers have a longer memory than the defender's estimate and thus the attackers can exploit the defender's planning. This observation suggests that it is more robust to over-estimate the attackers' memory length when there is uncertainty in Γ. We defer to future work to learn α_τ and Γ from attack data.

FS-2		Actual Value								
	alpha0	0.000	0.125	0.250	0.375	0.500	0.625	0.750	0.875	1.000
Estimated Value	0.000	6.068	5.000	2.505	0.224	-0.848	-1.431	-2.117	-2.798	-3.078
	0.125	5.891	5.344	3.747	0.967	-0.675	-1.623	-2.628	-3.205	-3.396
	0.250	5.318	5.014	4.298	2.187	-0.400	-2.060	-3.037	-3.334	-3.448
	0.375	4.180	4.056	3.830	3.200	0.081	-2.908	-3.387	-3.539	-3.625
	0.500	0.818	0.804	0.772	0.728	0.684	0.615	0.491	0.356	0.232
	0.625	0.685	0.685	0.685	0.685	0.685	0.685	0.685	0.684	0.683
	0.750	0.685	0.685	0.685	0.685	0.685	0.685	0.685	0.684	0.683
	0.875	0.677	0.672	0.671	0.673	0.676	0.680	0.683	0.683	0.680
	1.000	0.672	0.670	0.670	0.672	0.675	0.679	0.682	0.683	0.682
FS-1	[0,1]	0.685	0.685	0.685	0.685	0.685	0.685	0.685	0.684	0.683

Figure 6.3: Robustness against uncertainty in α_0 when $\Gamma = 1$

6.5.2 Learning and Planning Framework

When the parameter vectors $\{\omega^l\}$ are unknown, we compare Algorithm 7 with the baseline learning algorithm that uses MLE (denoted as **MLE**) when incorporated into planning algorithms. In each game of Game Set 2, we randomly choose $\{\omega^l\}$ for the 100 attackers from a three-dimensional normal distribution with mean $\mu = (-17.81, 0.72, 0.47)$ and covariance $\Sigma = \begin{pmatrix} 209.48 & -2.64 & -0.71 \\ -2.64 & 0.42 & 0.24 \\ -0.71 & 0.24 & 0.36 \end{pmatrix}$. We use **BU** to denote the case when an accurate prior (μ and Σ) is given to the defender. Recall that the defender plays against 100 attackers throughout a game, and **BU** aims to learn the parameter distribution of *these 100* attackers. **BU**$'$ represents the case when the prior distribution is a slightly deviated estimation (a normal distribution with random μ' and Σ' satisfying $\|\mu_i - \mu'_i\| \leq 5$ and $\|\Sigma'_{ii} - \Sigma_{ii}\| \leq 5$). **KnownPara** represents the case when the exact values of $\{\omega^l\}$ are known to the defender. We set a time limit of 30 minutes for the planning algorithms. Figure 6.2(g) – 6.2(h) show that **BU** and **BU**$'$ significantly outperform **MLE**. Indeed, the solution quality of **BU** and **BU**$'$ is close to that of **KnownPara**, indicating the effectiveness of the proposed learning algorithm. Also, **BU** and **BU**$'$ run much faster than **MLE** as **MLE** solves a convex optimization problem for each target in every round.

6.6 Chapter Summary

So far, the field had been lacking an appropriate game-theoretic model for green security domains: this chapter provides Green Security Games (GSG) to fill this gap. GSG's generalization of the Stackelberg assumption which is commonly used in previous work has led it to provide two new planning algorithms as well as a new learning framework, providing a significant advance over previous work in green security domains (Yang et al., 2014b; Haskell et al., 2014b).

Additional related work includes criminological work on poaching and illegal fishing (Lemieux, 2014; Beirne & South, 2007), but a game-theoretic approach is completely missing in this line of research.

Chapter 7

Reasoning about Spatial Constraints

In bringing a defender strategy represented by coverage probability to the real world, my thesis addresses one important aspect which is to deal with spatial constraints. Efforts have been made by law enforcement agencies in many countries to protect endangered animals; the most commonly used approach is conducting foot patrols. However, given their limited human resources, improving the efficiency of patrols to combat poaching remains a major challenge.

To address this problem, prior work introduced a novel emerging application called PAWS (Protection Assistant for Wildlife Security) (Yang et al., 2014b); PAWS is proposed as a game-theoretic decision-aid to optimize the use of human patrol resources to combat poaching. PAWS is an application in the general area of "security games" (Tambe, 2011); security-game-based decision support systems have previously been successfully deployed in the real-world in protecting critical infrastructure such as airports, flights, ports, and metro trains. PAWS was inspired by this success, and was the first of a new wave of proposed applications in the subarea now called "green security games" (Fang, Stone, & Tambe, 2015; Kar, Fang, Fave, Sintov, & Tambe, 2015). Specifically, PAWS solves a *repeated* Stackelberg security game, where the patrollers (defenders) conduct randomized patrols against poachers (attackers), while balancing the priorities

of different locations with different animal densities. Despite its promise, *the initial PAWS effort did not test the concept in the field*.

This chapter reports on PAWS's significant evolution over the last two years from a proposed decision aid to a regularly deployed application. We report on the innovations made in PAWS and lessons learned from the first tests in Uganda in Spring 2014, through PAWS's continued evolution to current regular use in Malaysia (in collaboration with two Non-Governmental Organizations: Panthera and Rimba). Indeed, the first tests revealed key shortcomings in PAWS's initial algorithms and assumptions (we will henceforth refer to the initial version of PAWS as *PAWS-Initial*, and to the version after our enhancement as PAWS). First, a major limitation, the severity of which was completely unanticipated, was that *PAWS-Initial* ignored topographic information. Yet in many conservation areas, high changes in elevation and the existence of large water bodies may result in a big difference in the effort needed for patrollers' movement. These factors also have a direct effect on poachers' movement. Second, *PAWS-Initial* assumed animal density and relevant problem features at different locations to be known. However, in practice, there are uncertainties in the payoffs of different locations, due to uncertainty over the animal movement. Not considering such uncertainty may lead to high degradation in patrol quality. Third, *PAWS-Initial* could not scale to provide detailed patrol routes in large conservation areas. Detailed routes require fine-grained discretization, which leads to a large number of feasible patrol routes. Finally, *PAWS-Initial* failed to consider patrol scheduling constraints.

In this chapter, we outline novel research advances which remedy the aforementioned limitations, making it possible to deploy PAWS on a regular basis. First, we incorporate elevation information and land features and use a novel hierarchical modeling approach to build a virtual "street map" of the conservation area. This virtual "street map" helps scale-up while providing

fine-grained guidance, and is an innovation that would be useful in many other domains requiring patrolling of large areas. Essentially, the street map connects the whole conservation area through easy-to-follow route segments such as a ridgeline, streams and river banks. The rationale for this comes from the fact that animals, poachers, and patrollers all use these features while moving. To address the second and third limitations, we build on the street map concept with a novel algorithm that uniquely synthesizes two threads of prior work in the security games literature; specifically, the new PAWS algorithm handles payoff uncertainty using the concept of minimax regret (Nguyen, Fave, Kar, Lakshminarayanan, Yadav, Tambe, Agmon, Plumptre, Driciru, Wanyama, & Rwetsiba, 2015), while simultaneously ensuring scalability – using our street maps – via the cutting plane framework (Yang, Jiang, Tambe, & OrdoÂŽnez, 2013). To address the final limitation, we incorporate in PAWS's algorithm the ability to address constraints such as patrol time limit and starting and ending at the base camp. In the final part of the chapter, we provide detailed information about the regular deployment of PAWS.

7.1 Background

Criminologists have begun to work on the problem of combating poaching, from policy design to illegal trade prevention (Lemieux, 2014). Geographic Information Systems (GIS) experts (Hamisi, 2008) and wildlife management staff (Wato, Wahungu, & Okello, 2006) have carefully studied the identification of poaching hotspots. In recent years, software tools such as SMART (SMART, 2013), MIST (Stokes, 2010) have been developed to help conservation managers record data and analyze patrols retrospectively. We work on a complementary problem of optimizing the patrol planning of limited security staff in conservation areas.

In optimizing security resource allocation, previous work on *Stackelberg Security Games (SSGs)* has led to many successfully deployed applications for the security of airports, ports and flights (Pita et al., 2008; Fang, Jiang, & Tambe, 2013). Based on the early work on SSG, recent work has focused on *green security games* (Kar et al., 2015), providing conceptual advances in integrating learning and planning (Fang et al., 2015) and the first application to wildlife security *PAWS-Initial*. *PAWS-Initial* (Yang et al., 2014b) models the interaction between the patroller (defender) and the poacher (attacker) who places snares in the conservation area (see Figure 7.1) as a basic green security game, i.e., a repeated SSG, where every few months, poaching data is analyzed, and a new SSG is set up to improve patrolling strategies. The deployed version of PAWS adopts this framework.

We provide a brief review of SSGs, using PAWS as a key example. In SSGs, the defender protects T targets from an adversary by optimally allocating a set of R resources ($R < T$) (Pita et al., 2008). In PAWS, the defender discretizes the conservation area into a grid, where each grid cell is viewed as a target for poachers, to be protected by a set of patrollers. The defender's pure strategy is an assignment of the resources to targets. The defender can choose a mixed strategy, which is a probability distribution over pure strategies. The defender strategy can be compactly represented as a coverage vector $\mathbf{c} = \langle c_i \rangle$ where c_i is the coverage probability, i.e., the probability that a defender resource is assigned to be at target i (Korzhyk, Conitzer, & Parr, 2010c). The adversary observes the defender's mixed strategy through surveillance and then attacks a target. An attack could refer to the poacher, a snare, or some other aspect facilitating poaching (e.g., poaching camp). Each target is associated with payoff values which indicate the reward and penalty for the players. If the adversary attacks target i, and i is protected by the defender, the defender gets reward $U^d_{r,i}$ and the adversary receives penalty $U^a_{p,i}$. Conversely, if

not protected, the defender gets penalty $U^d_{p,i}$ and the adversary receives reward $U^a_{r,i}$. Given a defender strategy $\mathbf{c}$, the players' expected utilities when target i is attacked are:

$$U^a_i = c_i U^a_{p,i} + (1 - c_i) U^a_{r,i} \tag{7.1}$$

$$U^d_i = c_i U^d_{r,i} + (1 - c_i) U^d_{p,i} \tag{7.2}$$

The game in PAWS is zero-sum, $U^d_{r,i} = -U^a_{p,i}$, $U^d_{p,i} = -U^a_{r,i}$. $U^a_{r,i}$ is decided by animal density – higher animal density implies higher payoffs.

In SSGs, the adversary's behavior model decides his response to the defender's mixed strategy. Past work has often assumed that the adversary is perfectly rational, choosing a single target with the highest expected utility (Pita et al., 2008). PAWS is the first deployed application that relaxes this assumption for a bounded rationality model called SUQR, which models the adversary's stochastic response to defender's strategy (Nguyen et al., 2013b). SUQR was shown to perform the best in human subject experiments when compared with other models. Formally, SUQR predicts the adversary's probability of attacking i as follows:

$$q_i = \frac{e^{w_1 c_i + w_2 U^a_{r,i} + w_3 U^a_{p,i}}}{\sum_j e^{w_1 c_j + w_2 U^a_{r,j} + w_3 U^a_{p,j}}} \tag{7.3}$$

where (w_1, w_2, w_3) are parameters indicating the importance of three key features: the coverage probability and the attacker's reward and penalty. The parameters can be learned from data.

Figure 7.1: A picture of a snare placed by poachers.

7.2 First Tests and Feedback

We first tested *PAWS-Initial* (Yang et al., 2014b) at Uganda's Queen Elizabeth National Park (QENP) for 3 days. Subsequently, with the collaboration of Panthera and Rimba, we started working in forests in Malaysia since September 2014[1]. These protected forests are home to endangered animals such as the Malayan Tiger and Asian Elephant but are threatened by poachers. One key difference of this site compared to QENP is that there is high changes in elevation, and the terrain is much more complex. The first 4-day patrol in Malaysia was conducted in November 2014. These initial tests revealed four areas of shortcomings, which restricted *PAWS-Initial* from being used regularly and widely.

The first limitation, which was surprising given that it has received no attention in previous work on security games, is the critical importance of topographic information that was ignored in *PAWS-Initial*. Topography can affect patrollers' speed in key ways. For example, lakes are inaccessible for foot patrols. Not considering such information may lead to the failure of completing

[1]For security of animals and patrollers, no latitude/longitude information is presented in this chapter.

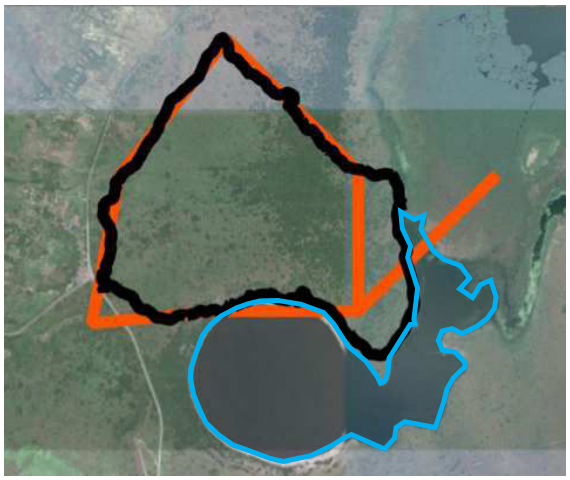
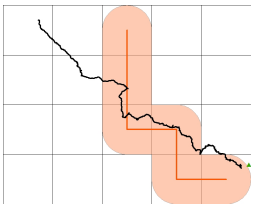


Figure 7.2: One patrol route during the test in Uganda.

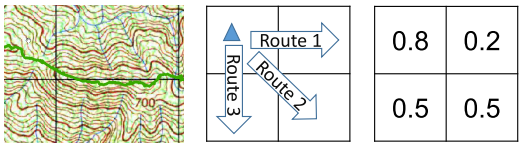


(a) Deployed route

(b) Patrollers

Figure 7.3: First 4-day patrol in Malaysia. Figure 7.3(a) shows one suggested route (orange straight lines) and the actual patrol track (black line). Figure 7.3(b) shows the patrollers walking along the stream during the patrol.

the patrol route. Figure 7.2 shows one patrol route during the test in Uganda. The suggested route (orange straight line) goes across the water body (lower right part of figure), and hence the patrollers decided to walk along the water body (black line). Also, changes in elevation require extra patrol effort, and extreme changes may stop the patrollers from following a route. For example, in Figure 7.3(a) [Malaysia], *PAWS-Initial* planned a route on a 1km by 1km grid (straight lines), and suggested that the patrollers walk to the north area (Row 1, Column 3) from the south side (Row 2, Column 3). However, such movement was extremely difficult because of the high changes in elevation. So patrollers decided to head towards the northwest area as the elevation



(a) Ridgeline (b) Feasible routes (c) Coverage

Figure 7.4: Illustrative examples.

change is more gentle. In addition, it is necessary to focus on terrain features such as ridgelines and streams (Figure 7.3(b)) when planning routes for three reasons: (i) they are important conduits for certain mammal species such as tigers; (ii) hence, poachers use these features for trapping and moving about in general; (iii) patrollers find it easier to move around here than on slopes. Figure 7.4(a) shows a prominent ridgeline.

The second limitation is that *PAWS-Initial* assumes the payoff values of the targets — e.g., $U^a_{r,i}$ – are known and fixed. In the domain of wildlife protection, there can be uncertainties due to animal movement and seasonal changes. Thus, considering payoff uncertainty is necessary for optimizing patrol strategy.

The third limitation is that *PAWS-Initial* cannot scale to provide detailed patrol routes in large conservation areas, which is necessary for successful deployment. Detailed routes require fine-grained discretization, which leads to an exponential number of routes in total.

The fourth limitation is that *PAWS-Initial* considers covering individual grid cells, but not feasible routes. In practice, the total patrolling time is limited, and the patrollers can move to nearby areas. A patrol strategy for implementation should be in the form of distribution over feasible patrol routes satisfying these constraints. Without taking these scheduling (routing) constraints into account, the optimal coverage probabilities calculated by *PAWS-Initial* may not be implementable. Figure 7.4(b) shows an example area that is discretized into four cells and the base camp is located in the upper left cell. There are three available patrol routes, each protecting two

targets. The coverage probabilities shown in Figure 7.4(c) cannot be achieved by a randomization over the three routes because the coverage of the upper left cell (Target 1) should be no less than the overall coverage of the remaining three cells since all routes start from the base camp.

7.3 PAWS Overview and Game Model

Figure 7.5 provides an overview of the deployed version of PAWS. PAWS first takes the input data and estimates the animal distribution and human activity distribution. Based on this information, an SSG based game model is built, and the patrol strategy is calculated. In wildlife protection, there is repeated interaction between patrollers and poachers. When patrollers execute the patrol strategy generated by PAWS over a period (e.g., three months), more information is collected and can become part of the input in the next round.

PAWS provides significant innovations in addressing the aforementioned limitations of *PAWS-Initial*. In building the game model, PAWS uses a novel hierarchical modeling approach to build a virtual street map, while incorporating detailed topographic information. PAWS models the poachers bounded rationality as described by the SUQR model and considers uncertainty in payoff values. In calculating the patrol strategy, PAWS uses ARROW (Nguyen et al., 2015) algorithm to deal with payoff uncertainty and adopts cutting plane approach and column generation to address the scalability issue introduced by scheduling constraints.

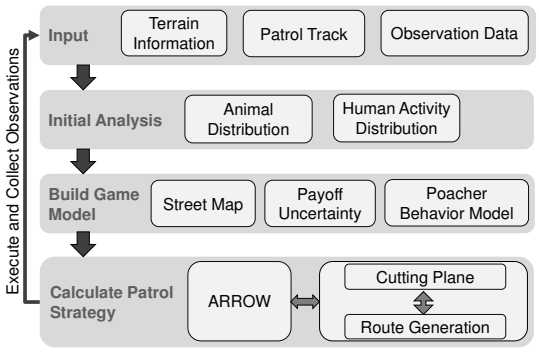


Figure 7.5: PAWS Overview

7.3.1 Input and Initial Analysis

The input information includes contour lines which describe the elevation, terrain information such as lakes and drainage, base camp locations, previous observations (animals and human activities), as well as previous patrol tracks. However, the point detections of animal and human activity presence are not likely to be spatially representative. As such, it is necessary to predict the animal and human activity distribution over the entire study area. To this end, we used: 1) JAGS (Plummer, 2003) to produce a posterior predictive density raster for tigers (as a target species) derived from a spatially explicit capture-recapture analysis conducted in a Bayesian framework; and 2) MaxEnt (Phillips, Anderson, & Schapire, 2006) to create a raster of predicted human activity distribution based on meaningful geographical covariates (e.g., distance to water, slope, elevation) in a Maximum Entropy Modelling framework.

7.3.2 Build Game Model

Based on the input information and the estimated distribution, we build a game model abstracting the strategic interaction between the patroller and the poacher as an SSG. Building a game model involves defender action modeling, adversary action modeling, and payoff modeling. We will

discuss all three parts but emphasize defender action modeling since this is one of the major challenges to bring PAWS to a regularly deployed application. Given the topographic information, modeling defender actions in PAWS is far more complex than any other previous security game domain.

7.3.2.1 Defender Action Modeling

Based on the feedback from the first tests, we aim to provide detailed guidance to the patrollers. If we use a fine-grained grid and treat every fine-grained grid cell as a target, computing the optimal patrolling strategy is exceptionally computationally challenging due to the large number of targets and the exponential number of patrol routes. Therefore, a key novelty of PAWS is to provide a hierarchical modeling solution, the first such model in security game research. This hierarchical modeling approach allows us to attain a good compromise between scaling up and providing detailed guidance. This approach would be applicable in many other domains for large open area patrolling where security games are applicable, not only other green security games applications, but others including patrolling of large warehouse areas or large open campuses via robots or UAVs.

More specifically, we leverage insights from hierarchical abstraction for heuristic search such as path planning (Botea, MÃŒller, & Schaeffer, 2004) and apply two levels of discretization to the conservation area. We first discretize the conservation area into 1km by 1km *Grid Cells* and treat every grid cell as a target. We further discretize the grid cells into 50m by 50m *Raster Pieces* and describe the topographic information such as elevation in 50m scale. The defender actions are patrol routes defined over a virtual "street map" – which is built in the terms of raster pieces while aided by the grid cells in this abstraction as described below. With this hierarchical

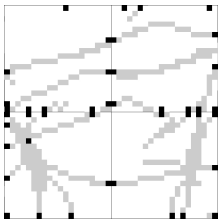


Figure 7.6: KAPs (black) for 2 by 2 grid cells.

modeling, the model keeps a small number of targets and reduces the number of patrol routes while allowing for details at the 50m scale.

The street map is a graph consisting of nodes and edges, where the set of nodes is a small subset of the raster pieces and edges are sequences of raster pieces linking the nodes. We denote nodes as Key Access Points (KAPs) and edges as route segments. The street map not only helps scalability but also allows us to focus patrolling on preferred terrain features such as ridgelines. The street map is built in three steps: (i) determine the accessibility type for each raster piece, (ii) define KAPs and (iii) find route segments to link the KAPs.

In the first step, we check the accessibility type of every raster piece. For example, raster pieces in a lake are inaccessible, whereas raster pieces on ridgelines or previous patrol tracks are easily accessible. Ridgelines and valley lines are inferred from the contour lines using existing approaches in hydrology (Tarboton, Bras, & Rodriguez-Iturbe, 2007).

The second step is to define a set of KAPs, via which patrols will be routed. We want to build the street map in such a way that each grid cell can be reached. So we first choose raster pieces which can serve as entries and exits for the grid cells as KAPs, i.e., the ones that are on the boundary of grid cells and are easily accessible. In addition, we consider existing base camps and mountain tops as KAPs as they are key points in planning the patroller's route. We choose

additional KAPs to ensure KAPs on the boundary of adjacent cells are paired. Figure 7.6 shows identified KAPs and easily accessible pieces (black and gray raster pieces respectively).

The last step is to find route segments to connect the KAPs. Instead of inefficiently finding route segments to connect each pair of KAPs on the map globally, we find route segments locally for each pair of KAPs within the same grid cell, which is sufficient to connect all the KAPs. When finding the route segment, we design a distance measure which estimates the actual patrol effort and also gives high priority to the preferred terrain features. The effort needed for three-dimensional movement can be interpreted as the equivalent distance on flat terrain. For example, for gentle slopes, equivalent "flat-terrain" distance is obtained by adding 8km for every 1km of elevation ascent according to Naismith's rule (Thompson, 2011). In PAWS, we apply Naismith's rule with Langmuir corrections (Langmuir, 1995) for gentle slopes (< 20) and apply Tobler's hiking speed function (Tobler, 1993) for steep slopes (≥ 20). Very steep slopes (> 30) are not allowed. We penalize not walking on preferred terrain features by adding extra distance. Given the distance measure, the route segment is defined as the shortest distance path linking two KAPs within the grid cell.

The defender's pure strategy is defined as a patrol route on the street map, starting from the base camp, walking along route segments and ending with base camp, with its total distance satisfying the patrol distance limit (all measured as the distance on flat terrain). The patroller confiscates the snares along the route and thus protects the grid cells. More specifically, if the patroller walks along a route segment which covers a sufficiently large portion (e.g., 50% of animal distribution) of a grid cell, the cell is considered to be protected. The defender's goal is to find an optimal mixed patrol strategy — a probability distribution over patrol routes.

7.3.2.2 Poacher Action Modeling and Payoff Modeling

The poacher's actions are defined over the grid cells to aid scalability. In this game, we assume the poacher can observe the defender's mixed strategy and then chooses one target (a grid cell) and places snares in this target. Following earlier work, the poacher in this game is assumed to be boundedly rational, and his actions can be described by the SUQR model.

Each target is associated with payoff values indicating the reward and penalty for the patrollers and the poachers. As mentioned earlier, PAWS models a zero-sum game and the reward for the attacker (and the penalty for the defender) is decided by the animal distribution. However, in this game model, we need to handle uncertainty in the players' payoff values since key domain features such as animal density which contribute to the payoffs are difficult to precisely estimate. In addition, seasonal or dynamic animal migration may lead to payoffs to become uncertain in the next season. We use intervals to represent payoff uncertainty in PAWS; the payoffs are known to lie within a certain interval whereas the exact values are unknown. Interval uncertainty is, in fact, a well-known concept to capture uncertainty in security games (Nguyen, Yadav, An, Tambe, & Boutilier, 2014; Nguyen et al., 2015). We determine the size of the payoff intervals at each grid cell based on patrollers' patrol efforts at that cell. Intuitively, if the patrollers patrol a cell more frequently, there is less uncertainty in the players' payoffs at that target and thus a smaller size of the payoff intervals.

7.4 Calculate Patrol Strategy

We build on algorithms from the rich security game literature to optimize the defender strategy. However, we find that no existing algorithm directly fits our needs as we need an algorithm that

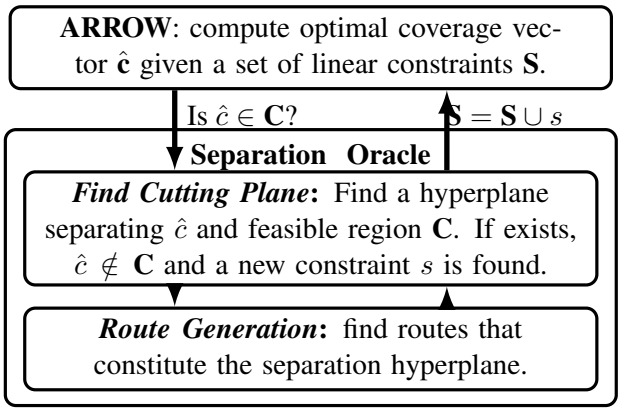


Figure 7.7: New integrated algorithm

can scale-up to the size of the domain of interest, where: (i) we must generate patrol routes over the street map over the entire conservation area region, while (ii) simultaneously addressing payoff uncertainty and (iii) bounded rationality of the adversary. While the ARROW (Nguyen et al., 2015) algorithm allows us to address (ii) and (iii) together, it cannot handle scale-up over the street map. Indeed, while the (virtual) street map is of tremendous value in scaling up as discussed earlier, scaling up given all possible routes ($\approx 10^{12}$ routes) on the street map is still a massive research challenge. We, therefore, integrate ARROW with another algorithm BLADE (Yang et al., 2013) for addressing the scalability issue, resulting in a novel algorithm that can handle all the three aforementioned challenges. The new algorithm is outlined in Figure 7.7. In the following, we explain how ARROW and BLADE are adapted and integrated.

ARROW attempts to compute a strategy that is robust to payoff uncertainty given that poachers' responses follow SUQR. The concept of minimizing maximum regret is a well-known concept in AI for decision making under uncertainty (Wang & Boutilier, 2003). ARROW uses the solution concept of *behavioral minimax regret* to provide the strategy that minimizes regret or utility loss for the patrollers in the presence of payoff uncertainty and bounded rational attackers. In small-scale domains, ARROW could be provided all the routes (the defender's pure strategies),

on the basis of which it would calculate the PAWS solution – a distribution over the routes. Unfortunately, in large scale domains like ours, enumerating all the routes is infeasible. We must, therefore, turn to an approach of incremental solution generation, which is where it interfaces with the BLADE framework.

More specifically, for scalability reasons, ARROW first generates the robust strategy for the patrollers in the form of coverage probabilities over the grid cells without consideration of any routes. Then a separation oracle in BLADE is called to check if the coverage vector is implementable. If it is implementable, the oracle returns a probability distribution over patrol routes that implements the coverage vector, which is the desired PAWS solution. If it is not implementable – see Figure 7.4(c) for an example of coverage vector that is not implementable – the oracle returns a constraint (cutting plane) that informs ARROW why it is not. For the example in Figure 7.4(b)-7.4(c), if ARROW generates a vector as shown in Figure 7.4(c), the constraint returned could be $c_1 \geq \sum_{i=2}^{4} c_i$ since all implementable coverage vector should satisfy this constraint. This constraint helps ARROW refine its solution. The process repeats until the coverage vector generated by ARROW is implementable.

As described in BLADE (Yang et al., 2013), to avoid enumerating all the routes to check whether the coverage vector is implementable, the separation oracle iteratively generate routes until it has just enough routes (usually after a small number of iterations) to match the coverage vector probabilities or get the constraint (cutting plane). At each iteration of this route generation (shown in the bottom-most box in Figure 7.7), the new route is optimized to cover targets of high value. However, we cannot directly use any existing algorithm to find the optimal route at each iteration due to the presence of our street map. But we note similarities to the well-studied

orienteering problem (Vansteenwegen, Souffriau, & Oudheusden, 2011) and exploit the insight of the S-algorithm for orienteering (Tsiligiridis, 1984).

In particular, in this bottom-most box of in Figure 7.7, to ensure each route returned is of high quality, we run a local search over a large number of routes and return the one with the highest total value. In every iteration, we start from the base KAP and choose which KAP to visit next through a weighted random selection. The next KAP to be visited can be any KAP on the map, and we assume the patroller will take the shortest path from the current KAP to the next KAP. The weight of each candidate KAP is proportional to the ratio of the additional target value that can be accrued and distance from current KAP. We set the lower bound of weight to be $\epsilon > 0$ to make sure every feasible route can be chosen with positive probability. The process continues until the patroller has to go back to the base to meet the patrol distance limit constraint. Given a large number of such routes, our algorithm returns a route close to the optimal solution.

Integrating all these algorithms, PAWS calculates the patrol strategy consisting of a set of patrol routes and the corresponding probability for taking them.

7.5 Deployment and Evaluation

PAWS patrols are now regularly deployed at a conservation area in Malaysia. This section provides details about the deployment and both subjective and objective evaluations of PAWS patrols.

PAWS patrol aims to conduct daily patrols from base camps. Before the patrol starts, PAWS generates the patrol strategy starting from the base camp selected by patrol team leader. The patrol distance limit considered by PAWS is 10km per day (equivalent flat terrain). As shown in Table 7.1, this leads to about 9000 raster pieces to be considered. Thus, it is impossible

Average # of Reachable Raster Pieces	9066.67
Average # of Reachable Grid Cells (Targets)	22.67
Average # of Reachable KAPs	194.33

Table 7.1: Problem Scale for PAWS Patrols.

Average Trip Length	4.67 Days
Average Number of Patrollers	5
Average Patrol Time Per Day	4.48 hours
Average Patrol Distance Per Day	9.29 km

Table 7.2: Basic Information of PAWS Patrols.

to consider each raster piece as a separate target or consider all possible routes over the raster pieces. With the two-level of discretization and the street map, the problem scale is reduced, with $8.57(= 194.33/22.67)$ KAPs and 80 route segments in each grid cell on average, making the problem manageable. The strategy generated by PAWS is a set of suggested routes associated with probabilities and the average number of suggested routes associated with probability $>$ 0.001 is 12.

Each PAWS patrol lasts for 4-5 days and is executed by a team of 3-7 patrollers. The patrol planner will make plans based on the strategy generated by PAWS. After reaching the base camp, patrollers execute daily patrols, guided by PAWS's patrol routes. Table 7.2 provides a summary of basic statistics about the patrols. During the patrol, the patrollers are equipped with a printed map, a handheld GPS, and data recording booklet. They detect animal and human activity signs and record them with detailed comments and photos. After the patrol, the data manager will put all the information into a database, including patrol tracks recorded by the hand-held GPS, and the observations recorded in the log book.

Figure 7.8 shows various types of signs found during the patrols. Table 7.3 summarizes all the observations. These observations show that there is a serious ongoing threat from the poachers. Column 2 shows results for all PAWS patrols. Column 3 shows results for explorative PAWS

(a) Tiger sign (Nov. 2014) (b) Human sign (lighter; Jul. 2015) (c) Human sign (old poacher camp; Aug. 2015) (d) Human sign (tree marking; Aug 2015)

Figure 7.8: Various signs recorded during PAWS patrols.

patrols, the (partial) patrol routes which go across areas where the patrollers have never been before. To better understand the numbers, we show in Column 4 the statistics about early-stage non-PAWS patrols in this conservation area, which were deployed for tiger survey. Although it is not a fair comparison as the objectives of the non-PAWS patrols and PAWS patrols are different, comparing Column 2 and 3 with Column 4 indicates that PAWS patrols are effective in finding human activity signs and animal signs. Finding the human activity signs is important to identify hotspots of poaching activity, and patrollers' presence will deter the poachers. Animals signs are not directly evaluating PAWS patrols, but they indicate that PAWS patrols prioritize areas with higher animal density. Finding these signs is aligned with the goal of PAWS – combat poaching to save animals – and thus is a proof for the effectiveness of PAWS. Comparing Column 3 with Column 2, we find the average number of observations made along the *explorative routes* is comparable to and even higher than that of all PAWS patrol routes. The observations on explorative routes are important as they lead to a better understanding of the unexplored area. These results show that PAWS can guide the patrollers towards hotspots of poaching activity and provide valuable suggestions to the patrol planners.

Along the way of PAWS deployment, we have received feedback from patrol planners and patrollers. The patrol planners mentioned that the top routes in PAWS solution (routes with highest probability) come close to an actual planner's routes, which shows PAWS can suggest feasible

Patrol Type	All PAWS Patrol	Explorative PAWS Patrol	Previous Patrol for Tiger Survey
Total Distance (km)	130.11	20.1	624.75
Average # of Human Activity Signs per km	0.86	1.09	0.57
Average # of Animal Signs per km	0.41	0.44	0.18

Table 7.3: Summary of observations.

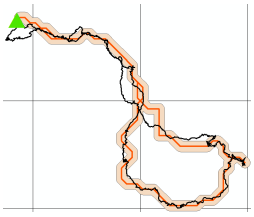


Figure 7.9: One daily PAWS Patrol route in Aug. 2015.

routes and potentially reduce the burden of planning effort. As we deploy PAWS in the future at other sites, the cumulative human planners' effort saved by using PAWS will be a considerable amount. In addition, patrollers commented that PAWS was able to guide them towards poaching hotspots. The fact that they found multiple human signs along the explorative PAWS patrol routes makes them believe that PAWS is good at finding good ridgelines that are taken by animals and humans. Patrollers and patrol planners also agree that PAWS generates detailed suggested routes which can guide the actual patrol. Patrollers commented that the suggested routes were mostly along the ridgeline, which are easier to follow, compared with the routes from the first trial by *PAWS-Initial*. Figure 7.9 shows one suggested route (orange line) and the actual patrol track (black line) during PAWS patrol in Aug. 2015 (shown on 1km by 1km grid). Due to the precision of the contour lines we get, we provide a 50m buffer zone (light orange polygon) around

the suggested route (orange line). The patrollers started from the base camp (green triangle) and headed to the southeast. The patrollers mostly followed PAWS's suggested route, indicating that the route generated by PAWS is easy to follow (contrast with *PAWS-Initial* as shown in Figure 7.3(a)). Finally, the power of randomization in PAWS solution can be expected in the long-term.

7.6 Lessons Learned

During the development and deployment process, we faced several challenges, and here we outline some lessons learned.

First, first-hand immersion in the security environment of concern is critical to understanding the context and accelerating the development process. The authors (from USC and NTU) intentionally went for patrols in the forest with the local patrolling team to familiarize themselves with the area. The first-hand experience confirmed the importance of ridgelines, as several human and animal signs are found along the way, and also confirmed that extreme changes in elevation require a considerable extra effort of the patrollers. This gave us the insight for building the street map.

Second, visualizing the solution is important for communication and technology adaptation. When we communicate with domain experts and human planners, we need to effectively convey the game-theoretic strategy generated by PAWS, which is a probability distribution over routes. We first visualize the routes with probability > 0.01 using ArcGIS so that they can be shown on the topographic map and the animal distribution map. Then for each route, we provide detailed information that can assist the human planners' decision-making. We not only provide basic statistics such as probability to be taken and total distance, but also estimate the difficulty level

for patrol, predict the probability of finding animals and human signs, and provide an elevation chart that shows how the elevation changes along the route. Such information can help planners' understanding the strategy.

Third, minimizing the need for extra equipment/effort would further ease PAWS future deployment, i.e., patrollers would prefer having a single handheld device for collecting patrol data and displaying suggested patrol routes. If PAWS routes could be embedded in the software that is already in use for collecting data in many conservation areas, e.g., SMART, it would reduce the effort required of planners. This is one direction for future development.

7.7 Chapter Summary

PAWS is a first deployed "green security game" application to optimize human patrol resources to combat poaching. We provided key research advances to enable this deployment; this has provided a practical benefit to patrol planners and patrollers. The deployment of PAWS patrols will continue at the site in Malaysia. Panthera has seen the utility of PAWS, and we are taking steps to expand PAWS to its other sites. This future expansion and maintenance of PAWS will be taken over by ARMORWAY (ARMORWAY, 2015), a "security games" company (starting in Spring 2016); ARMORWAY has significant experience in supporting security-games-based software deployments.

Chapter 8

Conclusion and Future Directions

8.1 Contributions

Whereas the first generation of ”security games” research provided algorithms for optimizing security resources in mostly static settings, my thesis advances the state-of-the-art to a new generation of security games, handling massive games with complex spatio-temporal settings and leading to real-world applications that have fundamentally altered current practices of security resource allocation. My work spans many different domains, including protecting ferry systems, forest, fisheries and wildlife. My thesis provides the first algorithms and models for advancing several key aspects of spatio-temporal challenges in security games, including actions over continuous time and space, frequent and repeated attacks as well as complex spatial constraints.

First, for games with moving targets such as ferries and refugee supply lines, players' actions are taken over continuous time, and I provide an efficient linear-programming-based solution while accurately modeling the attacker's continuous strategy. This work has been deployed by the US Coast Guard for protecting the Staten Island Ferry in New York City in past few years and fundamentally altering previously used tactics. Second, for games where actions are taken

over continuous space (for example games with forest land as the target), I provide an algorithm computing the optimal distribution of patrol effort. Third, my work addresses challenges with one key dimension of complexity – frequent and repeated attacks. Motivated by the repeated interaction of players in domains such as preventing poaching and illegal fishing, I introduce a novel game model that accounts for temporal behavior change of opponents and provide algorithms to plan effective sequential defender strategies. Furthermore, I incorporate complex terrain information and design the PAWS application to combat illegal poaching, which generates patrol plans with detailed patrol routes for local patrollers. PAWS has been deployed in a protected area in Southeast Asia, with plans for worldwide deployment.

While these challenges are brought by different domains, the proposed approaches of addressing these challenges share some high-level ideas that can shed light on future research. The spatio-temporal aspects often lead to a large and even infinite action space, and my work has shown that it is often possible to exploit the spatio-temporal structure to abstract the players' action space and strategy space.

In my work, I have used four approaches for abstraction. The first approach is to investigate the dominance relationship among the players' actions, and partitioning the action space can be helpful in finding such dominance relationship. In the problem of moving target protection, I partition the attacker's action set over continuous time and show that in each partitioned subset, there exists one time point that (weekly) dominates others. Therefore, it is sufficient to consider a finite number of attacker actions when calculating the optimal defender strategy. The second approach is to use a compact representation for the defender strategy based on equivalence relationship. Two defender strategies are equivalent if they lead to the same expected utility for the defender. If the defender strategies can be classified into several equivalent classes, and each class can be

represented using a small number of parameters, a compact representation is found and solving the problem with the compact representation can be much more efficient. The third approach is to exploit the property of the optimal defender strategy to reduce the search space. In the problem of area protection, we made the key observation that the defender should allocate the resources in a way such that the attacker gets zero benefit by going beyond the equilibrium distance. With this property, the defender's strategy can be uniquely decided once the equilibrium distance is known. Therefore, finding an optimal strategy for the defender is simplified to finding the equilibrium distance. The last approach is to use hierarchical modeling or hierarchical discretization. When a fine-grained discretization is necessary but computationally expensive, an alternative is to use multi-layers of abstraction. In my work, I have used two layers of abstraction for addressing the spatial constraints brought by topographical information. The higher layer is a coarse grid-based discretization, and in the lower layer, sub-graphs are built based on terrain features.

8.2 Future Directions

My thesis provides algorithms and models for advancing several key aspects of spatio-temporal dynamics in security games, and one extension would be to explore domains with the presence of several spatio-temporal aspects simultaneously. For example, consider games with both spatial continuity and repeated attacks. The first challenge of calculating optimal strategy in these games is to how to compactly represent the defender's strategy space. The second challenge is how to efficiently calculate the optimal patrol strategy for the defender given the complexity of the problem.

In dealing with spatio-temporal aspects in security games, my thesis has provided several approaches for exploiting the spatio-temporal structure and abstracting the players' action space or strategy space. The first three approaches mentioned in Section 8.1 can be seen as lossless abstraction since the abstraction does not lead to the degradation of solution quality. The last approach can be lossy in certain cases, but it is not yet clear in which cases a hierarchical discretization can be lossless or what kind of guarantee can be found. Furthermore, adaptive discretization may be a better way of balancing the solution quality and computation efficiency. It is important to understand the tradeoff in various problems. These questions lead to a need for further investigation on the problem of abstraction and discretization in security games.

Another direction of future work could be dealing with dynamic defender-attacker interactions in the presence of data. In domains such as wildlife conservation and urban security, repeated interactions between the defender and the attacker(s) are involved. The players can (partially) observe the other players' actions and the observation data collected from informants and surveillance may lead to a change in their behavior in the future. The basic questions remain open, for example, how to model these factors in a game-theoretic model and under what conditions a desirable equilibrium can be reached. In fact, a new solution concept may be required when the infinite time horizon of the game, the uncertainty in the observed actions and the bounded rationality of the attackers are considered.

In addition to infrastructure security domains and green security domains, a bunch of research problems are open for applying game theory to **cyber-security**. Cyber-security has become an increasingly significant problem. It is impossible to fully protect cyber assets at all times, and an important research direction is to develop game-theoretic solutions to inform decisions about what to protect and when. A key future research challenge is learning human behavior models

from data collected from human behavior logs and system logs, where timestamps could be an important attribute of attack. However, cyber-security brings a third crucial player, the human user; that was absent in my previous work. This raises fundamental new challenges in attacker-defender game settings which normally have only two players.

Bibliography

Agmon, N., Kraus, S., & Kaminka, G. A. (2008). Multi-robot perimeter patrol in adversarial settings. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2339–2345.

Albers, H. J. (2010). Spatial modeling of extraction and enforcement in developing country protected areas. *Resource and Energy Economics*, *32*, 165–179.

Alpern, S. (1992). Infiltration Games on Arbitrary Graphs. *Journal of Mathematical Analysis and Applications*, *163*, 286–288.

An, B., Kempe, D., Kiekintveld, C., Shieh, E., Singh, S. P., Tambe, M., & Vorobeychik, Y. (2012). Security games with limited surveillance. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pp. 1241–1248.

An, B., Tambe, M., Ordóñez, F., Shieh, E., & Kiekintveld, C. (2011). Refinement of strong Stackelberg equilibria in security games. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 587–593.

ARMORWAY (2015) http://armorway.com/.

Banerjee, B., & Peng, J. (2005). Efficient learning of multi-step best response. In *AAMAS*, pp. 60–66.

Basilico, N., Gatti, N., & Amigoni, F. (2009). Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS) - Volume 1*, pp. 57–64.

Beirne, P., & South, N. (Eds.). (2007). *Issues in Green Criminology*. Willan Publishing.

Blum, A., Haghtalab, N., & Procaccia, A. D. (2014). Learning optimal commitment to overcome insecurity. In *NIPS*.

Botea, A., MÃŒller, M., & Schaeffer, J. (2004). Near optimal hierarchical path-finding. *Journal of Game Development*, *1*, 7–28.

Bošanský, B., Lisý, V., Jakob, M., & Pěchouček, M. (2011). Computing time-dependent policies for patrolling games with mobile targets. In *The 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS) - Volume 3*, pp. 989–996.

Boyle, D. (2011). Logging in the wild west. *The Phnom Penh Post*.

Brown, M. (2015). *Balancing Tradeoffs in Security Games: Handling Defenders and Adversaries with Multiple Objectives*. Ph.D. thesis, University of Southern California.

Brown, M., Haskell, W. B., & Tambe, M. (2014). Addressing scalability and robustness in security games with multiple boundedly rational adversaries. In *Conference on Decision and Game Theory for Security (GameSec)*.

Chakraborty, D., Agmon, N., & Stone, P. (2013). Targeted opponent modeling of memory-bounded agents. In *Proceedings of the Adaptive Learning Agents Workshop (ALA)*.

Clarke, H. R., Reed, W. J., & Shrestha, R. M. (1993). Optimal enforcement of property rights on developing country forests subject to illegal logging. *Resource and Energy Economics*, *15*, 271–293.

Conitzer, V., & Sandholm, T. (2006). Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, EC '06, pp. 82–90.

Cowan, N. (2005). *Working Memory Capacity*. Essays in cognitive psychology. Psychology Press.

Daskalakis, C., & Weinberg, S. M. (2012). Symmetries and optimal multi-dimensional mechanism design. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, pp. 370–387.

De Alfaro, L., Henzinger, T. A., & Majumdar, R. (2001). Symbolic algorithms for infinite-state games. In *CONCUR 2001 – Concurrency Theory*, pp. 536–550. Springer.

de Cote, E. M., & Jennings, N. R. (2010). Planning against fictitious players in repeated normal form games. In *AAMAS*, pp. 1073–1080.

Dixon, J. A., & Sherman, P. B. (1990). *Economics of Protected Areas: A New Look at Benefits and Costs*. Island Press, Washington, DC.

Eliason, S. (1993). *Maximum Likelihood Estimation. Logic and Practice.*, Vol. 96 of *Quantitative Applications in the Social Sciences*. Sage Publications.

Fang, F., Jiang, A. X., & Tambe, M. (2013). Optimal patrol strategy for protecting moving targets with multiple mobile resources. In *AAMAS*.

Fang, F., Stone, P., & Tambe, M. (2015). When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Fudenberg, D., & Tirole, J. (1991). *Game Theory*. MIT Press.

Gal, S. (1980). *Search Games*. Academic Press, New York.

Gatti, N. (2008). Game theoretical insights in strategic patrolling: Model and algorithm in normal-form. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI)*, pp. 403–407.

Greenberg, M., Chalk, P., & Willis, H. (2006). *Maritime terrorism: risk and liability*. Rand Corporation monograph series. RAND Center for Terrorism Risk Management Policy.

Hall, J. B., & Rodgers, W. A. (1992). Buffers at the boundary. *Rural Development Forestry Network*, *Summer* (Paper 13a).

Halvorson, E., Conitzer, V., & Parr, R. (2009). Multi-step Multi-sensor Hider-Seeker Games. In *IJCAI*.

Hamisi, M. (2008). *Identification and mapping risk areas for zebra poaching: A case of Tarangire National Park, Tanzania*. Ph.D. thesis, Thesis, ITC.

Haskell, W., Kar, D., Fang, F., Tambe, M., Cheung, S., & Denicola, E. (2014a). Robust protection of fisheries with compass. In *Proceedings of the Twenty-Sixth Innovative Applications of Artificial Intelligence Conference, IAAI 2014, July 29 -31, 2014, Québec City, Québec, Canada.*, pp. 2978–2983.

Haskell, W. B., Kar, D., Fang, F., Tambe, M., Cheung, S., & Denicola, L. E. (2014b). Robust protection of fsheries with COmPASS. In *IAAI*.

Henzinger, T. A., Horowitz, B., & Majumdar, R. (1999). *Rectangular hybrid games*. Springer.

HOFER, H., CAMPBELL, K. L., EAST, M. L., & HUISH, S. A. (2000). Modeling the spatial distribution of the economic costs and benefits of illegal game meat hunting in the serengeti. *Natural Resource Modeling*, *13*(1), 151–177.

Jain, M. (2013). *Thwarting Adversaries with Unpredictability: Massive-scale Game-Theoretic Algorithms for Real-world Security Deployments*. Ph.D. thesis, Citeseer.

Jiang, A. X., Yin, Z., Zhang, C., Tambe, M., & Kraus, S. (2013). Game-theoretic randomization for security patrolling with dynamic execution uncertainty. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, AAMAS '13, pp. 207–214.

Kar, D., Fang, F., Fave, F. D., Sintov, N., & Tambe, M. (2015). "A Game of Thrones": When human behavior models compete in repeated Stackelberg security games. In *AAMAS 2015*.

Kiekintveld, C., Islam, T., & Kreinovich, V. (2013). Security games with interval uncertainty. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pp. 231–238.

Kiekintveld, C., Jain, M., Tsai, J., Pita, J., Ordóñez, F., & Tambe, M. (2009a). Computing optimal randomized resource allocations for massive security games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, pp. 689–696.

Kiekintveld, C., Jain, M., Tsai, J., Pita, J., Ordonez, F., & Tambe, M. (2009b). Computing optimal randomized resource allocations for massive security games. In *AAMAS*.

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

Korzhyk, D., Conitzer, V., & Parr, R. (2010b). Complexity of computing optimal stackelberg strategies in security resource allocation games. In *AAAI*, pp. 805–810.

Korzhyk, D., Conitzer, V., & Parr, R. (2010c). Complexity of computing optimal Stackelberg strategies in security resource allocation games. In *AAAI*, pp. 805–810.

Korzhyk, D., Conitzer, V., & Parr, R. (2011). Security games with multiple attacker resources. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume One*, IJCAI'11, pp. 273–279. AAAI Press.

Krause, A., Roper, A., & Golovin, D. (2011). Randomized sensing in adversarial environments. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2133–2139.

Krishna, V. (2009). *Auction theory*. Academic press.

Kumar, A., & Zilberstein, S. (2010). Anytime planning for decentralized POMDPs using expectation maximization. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pp. 294–301.

Langmuir, E. (1995). *Mountaincraft and Leadership: A Handbook for Mountaineers and Hill-walking Leaders in the British Isles*. Mountain Leader Training Board.

Lemieux, A. M. (Ed.). (2014). *Situational Prevention of Poaching*. Crime Science Series. Routledge.

Letchford, J. (2013). *Computational Aspects of Stackelberg Games*. Ph.D. thesis, Duke University.

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

Letchford, J., Conitzer, V., & Munagala, K. (2009). Learning and approximating the optimal strategy to commit to. In *Proceedings of the 2nd International Symposium on Algorithmic Game Theory*, pp. 250–262.

Letchford, J., & Vorobeychik, Y. (2012). Computing optimal security strategies for interdependent assets. In *The Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 459–468.

Lober, D. J. (1992). Using forest guards to protect a biological reserve in costa rica: one step towards linking parks to people. *Journal of Environmental Planning and Management*, *35*(1), 17.

Luber, S., Yin, Z., Fave, F. D., Jiang, A. X., Tambe, M., & Sullivan, J. P. (2013). Game-theoretic patrol strategies for transit systems: the trusts system and its mobile app (demonstration). In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)[Demonstrations Track]*, pp. 1377–1378.

MacKinnon, J., Mackinnnon, K., Child, G., & Thorsell, J. (1986). *Managing Protected Areas in the Tropics*. IUCN, Gland, Switzerland.

Marecki, J., Tesauro, G., & Segal, R. (2012). Playing repeated Stackelberg games with unknown opponents. In *AAMAS*, pp. 821–828.

McKelvey, R. D., & Palfrey, T. R. (1995). Quantal response equilibria for normal form games. *Games and Economic Behavior*, *2*, 6–38.

Milliman, S. R. (1986). Optimal fishery management in the presence of illegal activity. *Journal of Environmental Economics and Management*, *12*, 363–381.

Miltersen, P. B., & Sørensen, T. B. (2007). Computing proper equilibria of zero-sum games. In *Proceedings of the 5th International Conference on Computers and Games*, CG'06, pp. 200–211.

Nguyen, T. H., Fave, F. M. D., Kar, D., Lakshminarayanan, A. S., Yadav, A., Tambe, M., Agmon, N., Plumptre, A. J., Driciru, M., Wanyama, F., & Rwetsiba, A. (2015). Making the most of our regrets: Regret-based solutions to handle payoff uncertainty and elicitation in green security games. In *Conference on Decision and Game Theory for Security*.

Nguyen, T. H., Yadav, A., An, B., Tambe, M., & Boutilier, C. (2014). Regret-based optimization and preference elicitation for Stackelberg security games with uncertainty. In *AAAI*.

Nguyen, T. H., Yang, R., Azaria, A., Kraus, S., & Tambe, M. (2013a). Analyzing the effectiveness of adversary modeling in security games. In *Conference on Artificial Intelligence (AAAI)*.

Nguyen, T. H., Yang, R., Azaria, A., Kraus, S., & Tambe, M. (2013b). Analyzing the effectiveness of adversary modeling in security games. In *AAAI*.

Nguyen, T. H., Yang, R., Azaria, A., Kraus, S., & Tambe, M. (2013c). Analyzing the effectiveness of adversary modeling in security games.. *In AAAI*.

Owen, G. (1995). *Game Theory (3rd ed.)*. Academic Press.

Paruchuri, P., Pearce, J. P., Marecki, J., Tambe, M., Ordonez, F., & Kraus, S. (2008). Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '08, pp. 895–902, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Phillips, S. J., Anderson, R. P., & Schapire, R. E. (2006). Maximum entropy modeling of species geographic distributions. *Ecological Modelling*, *190*(3-4), 231–259.

Pita, J. (2012). *The Human Element: Addressing Human Adversaries in Security Domains*. Ph.D. thesis, University of Southern California.

Pita, J., Jain, M., Marecki, J., Ordóñez, F., Portway, C., Tambe, M., Western, C., Paruchuri, P., & Kraus, S. (2008a). Deployed ARMOR protection: the application of a game theoretic model for security at the Los Angeles International Airport. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track*, AAMAS '08, pp. 125–132.

Pita, J., Jain, M., Marecki, J., Ordóñez, F., Portway, C., Tambe, M., Western, C., Paruchuri, P., & Kraus, S. (2008b). Deployed ARMOR protection: the application of a game theoretic model for security at the Los Angeles International Airport. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track*, AAMAS '08, pp. 125–132.

Pita, J., Jain, M., Ordonez, F., Portway, C., Tambe, M., Western, C., Paruchuri, P., & Kraus, S. (2009). Using game theory for los angeles airport security.. *AI Magazine*, *30*, 43–57.

Pita, J., Jain, M., Ordonez, F., Tambe, M., & Kraus, S. (2010). Robust solutions to stackelberg games: Addressing bounded rationality and limited observations in human cognition. *Artificial Intelligence Journal, 174(15):1142-1171, 2010.*

Pita, J., Jain, M., Western, C., Portway, C., Tambe, M., Ordonez, F., Kraus, S., & Paruchuri, P. (2008). Deployed ARMOR protection: The application of a game theroetic model for security at the Los Angeles International Airport. In *AAMAS*.

Platzer, A. (2015). Differential game logic. *ACM Transactions on Computational Logic (TOCL)*, *17*(1), 1.

Plummer, M. (2003). JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling..

Powers, R., & Shoham, Y. (2005). Learning against opponents with bounded memory. In *IJCAI*, IJCAI'05, pp. 817–822, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Qian, Y., Haskell, W. B., Jiang, A. X., & Tambe, M. (2014). Online planning for optimal protector strategies in resource conservation games. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014)*.

Robinson, E. J. Z., Albers, H. J., & Williams, J. C. (2008). Spatial and temporal modelling of community non-timber forest extraction. *Journal of Environmental Economics and Management*, *56*, 234–245.

Robinson, E. J. Z., Albers, H. J., & Williams, J. C. (2011). Sizing reserves within a landscape: The roles of villagers' reactions and the ecological-socioeconomic setting. *Land Economics*, *87*, 233–249.

Robinson, E. J. (2008). India's disappearing common lands: fuzzy boundaries, encroachment, and evolving property rights. *Land Economics*, *84*(3), 409–422.

Rubinstein, A. (1997). *Modeling Bounded Rationality*, Vol. 1 of *MIT Press Books*. The MIT Press.

Sabourian, H. (1998). Repeated games with m-period bounded memory (pure strategies). *Journal of Mathematical Economics*, *30*(1), 1 – 35.

Sanchirico, J. N., & Wilen, J. E. (2001). A bioeconomic model of marine reserve creation. *Journal of Environmental Economics and Management*, *42*(November), 257–276.

Secretariat, G. T. I. (2013). Global tiger recovery program implementation plan: 2013-14. Tech. rep., The World Bank, Washington, D.C.

Shieh, E. (2015). *Not a Lone Ranger: Unleashing Defender Teamwork in Security Games*. Ph.D. thesis, University of Southern California.

Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., & Meyer, G. (2012a). PROTECT: A deployed game theoretic system to protect the ports of the United States. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pp. 13–20.

Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., & Meyer, G. (2012b). PROTECT: A deployed game theoretic system to protect the ports of the United States. In *AAMAS*.

Simon, L. K., & Stinchcombe, M. B. (1995). Equilibrium refinement for infinite normal-form games. *Econometrica*, *63*(6), 1421–1443.

Sinclair, A. R. E., & Arcese, P. (1995). *Serengeti II: Dynamics, Management, and Conservation of an Ecosystem*. University of Chicago Press, Chicago.

SMART (2013). The spatial monitoring and reporting tool (SMART). http://www.smartconservationsoftware.org/.

Soille, P. (2004). *Morphological Image Analysis: Principles and Applications*. Springer.

Stein, N. D., Ozdaglar, A., & Parrilo, P. A. (2008). Separable and low-rank continuous games. *International Journal of Game Theory*, *37*(4), 475–504.

Stokes, E. J. (2010). Improving effectiveness of protection efforts in tiger source sites: developing a framework for law enforcement monitoring using mist. *Integrative Zoology*, *5*(4), 363–377.

Stone, P., Kaminka, G. A., Kraus, S., & Rosenschein, J. S. (2010). Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pp. 1504–1509.

Tambe, M. (1997). Towards flexible teamwork. *JOURNAL OF ARTIFICIAL INTELLIGENCE RESEARCH*, *7*, 83–124.

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

Tarboton, D. G., Bras, R. L., & Rodriguez-Iturbe, I. (2007). On the extraction of channel networks from digital elevation data. *Hydrologic Processes*, *5*(1), 81–100.

Thompson, D. R. M., & Leyton-Brown, K. (2009). Computational analysis of perfect-information position auctions. In *Proceedings of the 10th ACM conference on Electronic commerce*, EC '09, pp. 51–60.

Thompson, S. (2011). *Unjustifiable Risk?: The Story of British Climbing*. Cicerone Press.

Tobler, W. (1993). Three presentations on geographical analysis and modeling. non-isotropic geographic modeling: speculations on the geometry of geography, and global spatial analysis (93-1). Tech. rep., UC Santa Barbara.

Tsai, J., Rathi, S., Kiekintveld, C., Ordonez, F., & Tambe, M. (2009a). IRIS - a tool for strategic security allocation in transportation networks. In *The Eighth International Conference on Autonomous Agents and Multiagent Systems - Industry Track*, AAMAS '09, pp. 37–44.

Tsai, J., Rathi, S., Kiekintveld, C., Ordonez, F., & Tambe, M. (2009b). IRIS - a tool for strategic security allocation in transportation networks. In *The Eighth International Conference on Autonomous Agents and Multiagent Systems - Industry Track*, AAMAS '09, pp. 37–44.

Tsiligiridis, T. (1984). Heuristic methods applied to orienteering. *The Journal of the Operational Research Society*, *35*(9), pp. 797–809.

van Damme, E. (1987). *Stability and Perfection of Nash equilibria*. Springer-Verlag.

Vansteenwegen, P., Souffriau, W., & Oudheusden, D. V. (2011). The orienteering problem: A survey. *European Journal of Operational Research*, *209*(1), 1–10.

Vaněk, O., Jakob, M., Hrstka, O., & Pěchouček, M. (2011). Using multi-agent simulation to improve the security of maritime transit. In *Proceedings of 12th International Workshop on Multi-Agent-Based Simulation (MABS)*, pp. 1–16.

Wang, T., & Boutilier, C. (2003). Incremental utility elicitation with the minimax regret decision criterion. In *IJCAI*.

Wato, Y. A., Wahungu, G. M., & Okello, M. M. (2006). Correlates of wildlife snaring patterns in tsavo west national park, Kenya. *Biological Conservation*, *132*(4), 500–509.

Yang, R. (2014). *Human Adversaries in Security Games: Integrating Models of Bounded Rationality and Fast Algorithms*. Ph.D. thesis, University of Southern California.

Yang, R., Ford, B., Tambe, M., & Lemieux, A. (2014a). Adaptive resource allocation for wildlife protection against illegal poachers. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Yang, R., Ford, B., Tambe, M., & Lemieux, A. (2014b). Adaptive resource allocation for wildlife protection against illegal poachers. In *AAMAS*.

Yang, R., Jiang, A. X., Tambe, M., & OrdoÂŽnez, F. (2013). Scaling-up security games with boundedly rational adversaries: A cutting-plane approach. In *IJCAI*.

Yang, R., Ordonez, F., & Tambe, M. (2012). Computing optimal strategy against quantal response in security games. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 847–854. International Foundation for Autonomous Agents and Multiagent Systems.

Yin, Z. (2013). *Addressing Uncertainty in Stackelberg Games for Security: Models and Algorithms*. Ph.D. thesis, University of Southern California.

Yin, Z., Jiang, A., Johnson, M., Tambe, M., Kiekintveld, C., Leyton-Brown, K., Sandholm, T., & Sullivan, J. (2012a). TRUSTS: Scheduling randomized patrols for fare inspection in transit systems. In *IAAI*.

Yin, Z., Jiang, A. X., Johnson, M. P., Kiekintveld, C., Leyton-Brown, K., Sandholm, T., Tambe, M., & Sullivan, J. P. (2012b). TRUSTS: Scheduling randomized patrols for fare inspection in transit systems. In *Proceedings of the Twenty-Fourth Conference on Innovative Applications of Artificial Intelligence (IAAI)*, pp. 2348–2355.

Yin, Z., Korzhyk, D., Kiekintveld, C., Conitzer, V., & Tambe, M. (2010). Stackelberg vs. nash in security games: Interchangeability, equivalence, and uniqueness. *In AAMAS*.

Yin, Z., & Tambe, M. (2011). Continuous time planning for multiagent teams with temporal constraints. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume One*, IJCAI'11, pp. 465–471. AAAI Press.

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