Combating Adversaries under Uncertainties in Real-world Security Problems: Advanced Game-theoretic Behavioral Models and Robust Algorithms

by

Thanh H. Nguyen

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)

May 2016

Acknowledgement

First and foremost, I would like to thank my advisor, Prof. Milind Tambe, for all of his support and encouragement. I struggled a lot during the first two years of my PhD life due to my lack of English literacy and common sense of living in US. I barely talked to other students or spoke up at the group meetings. I doubted my ability to do research and even thought of giving up on pursuing a PhD. But Milind did not give up on me. He always stood by my side, encouraging and supporting me to get through this hard time in my life. He taught me not only to do scientific study but also to communicate with other people and to convey my ideas. He kept reminding me to think about the big picture of the research and to push all ideas to the limit. Milind is the best advisor ever and a role model who is so knowledgeable and dedicated and kind.

I also would like to thank other members of my thesis committee for providing valuable feedback on my research and suggesting new and interesting research topics to pursue: David Kempe, Jonathan Gratch, William Halfond, Richard John, and Ariel Proccacia.

I'm fortunate to collaborate with great researchers: Noa Agmon, Bo An, Amos Azaria, Colin Beal, Craig Boutilier, Reuben Clements, Rob Critchlow, Tracy Cui, Francesco Delle Fave, Margaret Driciru, Fei Fang, Benjamin Ford, Shahrzad Gholami, Nika Haghtalab, Albert Jiang, Richard John, Lucas Joppa, Debarun Kar, Christopher Kiekintveld, Sarit Kraus, Aravind Lakshminarayanan, Wai Lam, Andrew Lemieux, Rajiv Maheswaran, Rob Pickles, James Pita, Andrew Plumptre, Ariel Procaccia, Aggrey Rwetsiba, Amandeep Singh, Arunesh Sinha, Nicole Sintov, Anjon Sunny, Milind Tambe, Arjun Tambe, Jason Tsai, Richard Van Deventer, Fred Wanyama, Amulya Yadav, Rong Yang.

I want to thank the entire TEAMCORE community. Special thanks to Albert Jiang for your tremendous help in doing research, Manish Jain for always being willing to talk to me about research and to help me revise my papers, Matthew Brown and Jun Young Kwak and Amulya Yadav for being lovely officemates always listening to my silly stories, traveling around with

me, and taking me to nice restaurants, Eric Shieh and his lovely wife, Linda Shieh for being so kind and treating me so well, Fei Fang for sharing the small working corner with me and for being a nice and cute roomate during conferences, Sara Mc Carthy for helping me review papers several times and for being a cool lifting trainer, and Arunesh Sinha, Debarun Kar, Benjamin Ford, Yasaman Dehghani Abbasi, Elizabeth Orrico, Bryan Wilder, Aaron Schlenker, Shahrzad Gholami, Leandro Marcolino, Yundi Qian, Chao Zhang, Haifeng Xu, Rong Yang, Jason Tsai, James Pita, and Zhengyu Yin for being such great labmates.

Finally, I want to thank my family for their unconditional love and support. Thank you for always standing by my side and encouraging me to pursue an academic career.

Contents

List Of Figures

List Of Tables

Abstract

Security is a global concern. Real-world security problems range from domains such as the protection of ports, airports, and transportation from terrorists to protecting forests, wildlife, and fisheries from smugglers, poachers, and illegal fishermen. A key challenge in solving these security problems is that security resources are limited; not all targets can be protected all the time. Therefore, security resources must be deployed intelligently, taking into account the responses of adversaries and potential uncertainties over their types, priorities, and knowledge. Stackelberg Security Games (SSG) have drawn a significant amount of interest from security agencies by capturing the strategic interaction between security agencies and human adversaries. SSG-based decision aids are in widespread use (both nationally and internationally) for the protection of assets such as major ports in the US, airport terminals, and wildlife and fisheries.

My research focuses on addressing uncertainties in SSGs — one recognized area of weakness. My thesis provides innovative techniques and significant advances in addressing these uncertainties in SSGs. First, in many security problems, human adversaries are known to be boundedly rational, and often choose targets with non-highest expected value to attack. I introduce novel behavioral models of adversaries which significantly advance the state-of-the-art in capturing the adversaries' decision making. More specifically, my new model for predicting poachers' behavior in wildlife protection is the first game-theoretic model which takes into account key domain challenges including imperfect poaching data and complex temporal dependencies in poachers' behavior. The superiority of my new models over the existing ones is demonstrated via extensive experiments based on the biggest real-world poaching dataset, collected in a national park in Uganda over 12 years. Second, my research also focuses on developing new robust algorithms which address uncertainties in real-world security problems. I present the first unified maximin-based robust algorithm — a single algorithm — to handle all different types of uncertainties explored in SSGs. Furthermore, I propose a less conservative decision criterion; minimax regret,

for generating new, candidate defensive strategies that handle uncertainties in SSGs. In fact, minimax regret and maximin can be used in different security situations which may demand different robust criteria. I then present novel robust algorithms to compute minimax regret for addressing payoff uncertainty.

A contribution of particular significance is that my work is deployed in the real world; I have deployed my robust algorithms and behavioral models in the PAWS system, which is currently being used by NGOs (Panthera and Rimba) in a conservation area in Malaysia.

Chapter 1

Introduction

Security is a critical concern around the world. Real-world security problems include protecting ports, airports, and other critical national infrastructure from terrorists, protecting wildlife, fishery, and forests from poachers, illegal fisherman and smugglers, as well as problems of drug interdiction, prevention of urban crimes, and cyber-physical security (Figure 1.1). A common and important challenge that arises in all of these security problems is that security agencies have limited security resources and cannot completely protect all the targets at all times. Therefore, it is important for security agencies to effectively allocate these limited resources to protect the targets, taking into account the importance of the targets, the responses of the adversaries, and potential uncertainties over the types, priorities, and knowledge of the adversaries.

Defender-attacker SSGs have drawn great attention in providing a practical game-theoretic model for security problems (Brown, Carlyle, Salmerón, & Wood, 2006; Conitzer, 2012; Basilico, Gatti, & Amigoni, 2009) and have found application in a number of real-world domains (Tsai, Kiekintveld, Ordonez, Tambe, & Rathi, 2009; Tambe, 2011; Shieh, An, Yang, Tambe, Baldwin, DiRenzo, Maule, & Meyer, 2012). SSGs are a class of Stackelberg games where a defender acts as a leader and an adversary acts as a follower (Von Stengel & Zamir, 2004; Tambe, 2011). While the defender attempts to allocate her limited resources to protect a set of important targets, the adversary plans to attack one such target. SSGs are commonly used in real-world security domains because they captures the fact that the defender first commits to a mixed strategy assuming that the adversary can observe that strategy; then, the adversary takes his action.

(a) Ferry protection (b) Port protection (c) Wildlife protection

Figure 1.1: Real-world security domains

1.1 Problem Addressed

Standard SSGs require perfect knowledge about the game and unrealistic assumptions such as: (i) a perfectly rational adversary, who always maximizes his expected value; (ii) prior knowledge of the adversary's payoff values that quantitatively express the adversary's preference over different targets; and (iii) that the defender's strategy is always executed perfectly and is always fully observed by the adversary. However, these assumptions are not ideal for solving real-world security problems. For example, adversary payoff values can be extremely difficult to assess and are generally characterized by significant uncertainty. As a result, defensive strategies based on these limiting assumptions may not be robust to uncertainties existing in real-world security settings. Thus, it is critical to take into account a variety of uncertainties in security games in order to obtain effective patrolling strategies for the defender. There are in fact several research challenges w.r.t addressing uncertainties in SSGs that we need to resolve.

First, in real-world security problems, such as airport and flight security or wildlife protection, the defender (security agencies) must conduct patrols to protect important targets (e.g., terminal, flight, wildlife) against the adversary (e.g., terrorists, poachers). In order to be able to do this effectively, it is important for the defender to be able to anticipate which targets the adversary is going to attack. In many security problems, human attackers are known to be boundedly rational, and often choose targets with non-highest expected value to attack. While existing behavioral models rely on expected utility to infer the adversary's behavior (McFadden, 1972; McKelvey & Palfrey, 1995), these models may not be well-suited to security domains such as wildlife protection in which the adversary's decision making involves multiple complex domain

features. This challenge raises an important question of how to incorporate all important complex domain features into reasoning about the adversary's behavior.

Furthermore, while existing behavioral models in SSGs have been applied to predicting poachers' behavior in wildlife protection (Yang, Ford, Tambe, & Lemieux, 2014; Fang, Stone, & Tambe, 2015; Fang, Nguyen, Pickles, Lam, Clements, An, Singh, Tambe, & Lemieux, 2016; Kar, Fang, Fave, Sintov, & Tambe, 2015), there remain several open research challenges in wildlife protection which need to be resolved. Notably, existing behavioral models in SSGs were developed with standard SSGs in mind, and are more appropriate for predicting adversaries' behavior in infrastructure security games than in the wildlife setting. In fact, understanding the limits of these models when applied to wildlife protection is an important step which could help in improving the prediction accuracy of poachers' behavior. In particular, previous behavioral models make several limiting assumptions, including (a) all poaching signs (e.g., snares) are perfectly observable by the rangers; (b) poachers' activities in one time period are independent of their activities in previous or future time periods; (c) the number of poachers is known. To understand the limiting nature of these assumptions, consider the issue of observability. The rangers' capability of making observations over a large geographical area is limited. In other words, there may still be poaching activities happening in areas where rangers did not find any poaching sign. Furthermore, it is critical to incorporate aspects which may affect the poachers' behavior such as the time dependency of the poachers' activities. Lastly, the rangers are unaware of the total number of poachers in the park (prior information required for existing behavioral models). Therefore, it is important to build new behavioral models of poachers which overcome these limitations.

In addition to research on behavioral modeling to address the adversary's bounded rationality, there is another major line of research which focuses on developing robust algorithms to compute an optimal strategy for the defender under uncertainties. Unfortunately, all previous work in robust optimization in SSGs compartmentalizes the uncertainties. For example, while some research has focused exclusively on uncertainty over the defender's assessment of the adversary's payoffs (Kiekintveld, Islam, & Kreinovich, 2013), other work has focused exclusively on uncertainty over the defender's execution of the provided strategy and the adversary's surveillance of this strategy (Yin, Jain, Tambe, & Ordonez, 2011), and yet other exclusively on the uncertainty given the adversary's bounded rationality (Jiang, Nguyen, Tambe, & Procaccia, 2013; Pita, Jain,

Ordonez, Tambe, Kraus, & Magori-Cohen, 2009). The lack of a unified framework implies that existing algorithms suffer losses in solution quality when handling uncertainties in real-world security situations – where multiple types of uncertainties may exist simultaneously. In addition, insights for improving performance are not leveraged across these compartments; again leading to losses in solution quality or efficiency. These limitations of existing robust algorithms lead to an important need to develop a unified robust framework that can handle multiple types of uncertainties.

Moreover, previous work in security games only focuses on robust techniques such as the maximin method (Kiekintveld et al., 2013; Yin et al., 2011; Jiang et al., 2013); we lack an alternative less-conservative robust criterion for addressing uncertainties in SSGs. In fact, different robust criteria may be appropriate for different security uncertainty settings. Therefore, developing new robust algorithms based on different robust criteria is neccessary, allowing security policy makers to flexibly select a robust solution which is well-suited for their security domains. Last but not least, while security agencies can use available resources to elicit information w.r.t uncertain elements (e.g., payoffs) to reduce uncertainties, how to efficiently exploit these elicitation resources given that the resources are limited remains an open important research question.

1.2 Main Contributions

My research focuses on providing innovative techniques and significant advances for addressing the challenges of uncertainties in real-world security problems, including 1) uncertainty in the adversary's payoff; 2) uncertainty related to the defender's strategy; and 3) uncertainty in the adversary's rationality. My key research contributions include:

- new behavioral models of adversaries (e.g., terrorists and poachers) in real-world security domains built using both real-world and laboratory data.
- new robust planning algorithms for security agencies based on maximin and minimax regret methods developed for a variety of domain uncertainty settings.

1.2.1 Modeling Adversary Decision Making

As *the first contribution*, I introduce a new behavioral model, LensQR, which predicts a stochastic distribution of the adversary's responses over the targets (Nguyen, Yang, Azaria, Kraus, & Tambe, 2013). Essentially, LensQR is built upon two existing well-known models in liturature: the Quantal Response behavioral model (McKelvey & Palfrey, 1995; McFadden, 1972) and the Lens model (Brunswik, 1952). The advantage of the new LensQR model lies in facilitating an easy way to combine all domain features in reasoning the adversary's behavior. I conducted extensive human subject experiments using both Amazon Turk workers and security experts wherein the attacking data is collected and used to learn the model. LensQR is then used to model the adversary when computing the optimal patrolling strategy for the defender. The experimental results demonstrate the superiority of LensQR over existing approaches, including Quantal Response. Furthermore, LensQR is preliminarily evaluated in the wildlife protection domain by testing its ability to predict real poacher behavior. In wildlife protection, the rangers conduct patrols within a designated area to protect wildlife from poaching, which can be represented as a SSG problem in which the rangers play as the defender and the poachers are the attacker. I use real-world wildlife/poaching data collected in Queen Elizabeth National Park (QENP) in Uganda to evaluate the prediction accuracy of LensQR. The results show that LensQR outperforms the existing bounded rationality models. Finally, LensQR has been incorporated into the PAWS system which is currently deployed in Malaysia (Fang et al., 2016).

While LensQR is shown to be the best model in capturing the poachers' behavior compared to existing behavioral models in SSG, LensQR, which mainly focuses on standard SSGs, still faces several limitations when applied for wildlife protection as mentioned in Section 1.1. As *the second contribution*, I improve the LensQR behavioral model and integrate this new version of LensQR (called LensQR-Poacher) into the CAPTURE system — a new predictive anti-poaching tool for wildlife protection (Nguyen, Sinha, Gholami, Plumptre, Joppa, Tambe, Driciru, Wanyama, Rwetsiba, Critchlow, & Beale, 2016). The new LensQR model of poachers provides significant advances over previous models from behavioral game theory and conservation biology. It accounts for: (i) the rangers' imperfect detection of poaching signs; (ii) complex temporal dependencies in the poachers' behaviors; (iii) lack of knowledge of the number of poachers. LensQR-Poacher's prediction accuracy is extensively evaluated based on a detailed analysis of

the largest dataset of real-world defender-adversary interactions. This dataset is collected by rangers in QENP over 12 years and consists of thousands of poaching signs and years of rangers' past patrols. The experimental results show that LensQR-Poacher is superior to existing models in predicting the poachers' behavior, demonstrating the advances of LensQR-Poacher over the previous state-of-the-art models. Furthermore, I present a new game-theoretic algorithm for computing the rangers' optimal patrolling assuming the poachers' behavior follows LensQR-Poacher. Specifically, I provide a new game-theoretic algorithm for single/multiple-step patrolling plans wherein the poachers' actions are recursively explored in multiple time steps. To that end, the CAPTURE tool with the LensQR-Poacher model will be tested in Uganda in 2016.

1.2.2 Robust Algorithms for Optimizing Defender Strategy

My research also focuses on developing new robust algorithms which address uncertainties in real-world security problems. The current state-of-the-art has provided only compartmentalized robust maximin-based algorithms that handle uncertainty exclusively either in the defender's strategy or in the adversary's payoff or in the adversary's rationality, leading to potential failures in real-world scenarios where a defender often faces multiple types of uncertainties. As *the third contribution*, I present the first unified maximin-based algorithm, URAC, to handle all different types of uncertainties explored in SSGs (Nguyen, Jiang, & Tambe, 2014). URAC has two key novel ideas. First, despite the existence of simultaneous, multiple, inter-dependent uncertainties, the resulting multi-dimensional uncertainty space can be converted into a uni-dimensional uncertainty space of the adversary actions only. This dimensional conversion reduces significantly the complexity of finding the worst-case scenario for the defender due to uncertainties. Second, the allocation strategy space of the defender can be clustered into a finite number of sub-spaces such that any defender strategy within each sub-space leads to the same uncertainty set of adversary actions. This space partition allows us to overcome the challenge of infinitely (and certainly infeasibly) iterating all possible defender strategies (where each strategy has a different corresponding set of adversary actions) to find the optimal one. Instead, given the space partition, the problem of computing an optimal strategy for the defender which is robust to uncertainties can be decomposed into a finite number of sub-problems, each of which finds a sub-optimal strategy within the corresponding defender strategy sub-space. Every sub-problem is represented as

a linear program which can be solved in polynomial time. Finally, URAC finds the optimal allocation strategy as the best solution among all the sub-optimal ones across all sub-problems. Based on URAC, I then introduce approximate scalable robust algorithms which explore intrinsic properties of uncertainty space to handle uncertainties in large-scale security games.

Furthermore, as *the fourth contribution*, I develop several different regret-based robust algorithms to handle payoff uncertainty in both non-zero-sum and zero-sum games against perfectly and boundedly rational adversaries (Nguyen, Yadav, An, Tambe, & Boutilier, 2014; Nguyen, Fave, Kar, Lakshminarayanan, Yadav, Tambe, Agmon, Plumptre, Driciru, Wanyama, & Rwetsiba, 2015). Previous work in security games only focuses on robust techniques such as the maximin method; we lack an alternative less-conservative robust criterion for addressing uncertainties in SSGs. I propose a less conservative decision criterion; Minimax regret for generating new, candidate defensive strategies that handle uncertainties in SSGs. I then present novel robust algorithms to compute minimax regret for addressing payoff uncertainty. Minimax Regret is a robust approach for handling uncertainty, finding the solution which minimizes the maximum regret (i.e., solution quality loss) with respect to a given uncertainty set. A key challenge is that there are an infinite number of possible payoffs within the range of uncertainty, leading to intractable computation. My idea is to use incremental payoff generation; I start by solving a relaxed minimax regret problem given a small set of payoff samples. Then new payoff samples are iteratively generated and added into the current set of samples until the optimal solution is obtained. Finally, I develop the first elicitation strategies (based on minimax regret) that optimize the defender's efforts in assessing payoffs, allowing reduction in uncertainty of those parameters. To that end, my regret-based algorithm is extended and integrated in the PAWS system to generate patrolling strategies used by rangers in a protected area in Malaysia (Fang et al., 2016).

1.3 Overview of Thesis

My thesis is organized as follows. Chapter 2 introduces fundamental background materials necessary for the research presented in the thesis. Chapter 3 provides an overview of the related work. Chapter 4 presents the new behavioral model which integrates the Lens utility function into the Quantal Response model to further improve the model performance in predicting the adversary's

behaviors in SSGs. Chapter 5 investigates the new behavioral model of poachers in wildlife protection, which addresses intrinsic domain challenges that existing behavioral models in SSGs failed to handle. Chapter 6 explains the unified robust maximin-based algorithms for computing an optimal patrolling strategy for the defender which is robust to multiple types of uncertainties. Chapter 7 introduces the new robust algorithms which is based on minimax regret for addressing the challenge of payoff uncertainty in generating an optimal patrolling strategy for the defender. Chapter 8 presents the new regret-based algorithms which are applied to generating patrolling strategies for rangers in wildlife protection in different security settings of uncertainties. Finally, chapter 9 concludes the thesis and presents possible future directions.

Chapter 2

Background

In this chapter, I will provide a general background of Stackelberg games, Bayesian Stackelberg games as well as the standard solution concept, Strong Stackelberg equilibrium, for solving these games. I then describe a restricted class of Stackelberg games for security, which is called Stackelberg security games. I follow up with introducing motivating domain examples of real-world security problems . Finally, I present baseline algorithms for solving Stackelberg security games.

2.1 Stackelberg Games

Stackelberg games refer to a class of leader-follower games in which the leader commits to a strategy first while the follower can observe the leader's strategy and then optimally responds by maximizing his expected utility. In this thesis, I refer to the leader as "she" while the follower as "he" for explanatory purpose. Table 2.1 shows the payoff matrix of a Stackelberg game; the row player is the leader and the column player is the follower. This example was first introduced by (Conitzer & Sandholm, 2006) to illustrate the advantage of being a leader. In this Stackelberg game, each player has two actions: (L_1, L_2) for the leader and (F_1, F_2) for the follower. For each pair of actions, each player will receive a payoff (i.e., the first number is the leader payoff and the second is the follower payoff). For example, if the leader takes action L_1 while the follower chooses action F_2, the leader receives a payoff of 4 while the follower obtains a payoff of 0.

If the two players move simultaneously, the Nash equilibrium for this game is when the leader plays L_1 and the follower plays F_1. In this case, the leader and the follower receives a payoff of 2 and 1 respectively. On the other hand, if the row player moves first, she can choose action

	F_1	F_2
L_1	$2, 1$	$4, 0$
L_2	$1, 0$	$3, 1$

Table 2.1: An example of a simple Stackelberg game

L_2 to obtain a higher payoff. Specifically, since the follower now observes that the leader takes action L_2, the best action for the follower is to play action F_2. As a result, the leader receives a payoff of 3 (which is higher payoff than in the case of simultaneous moves) while the follower gets a payoff of 1. Furthermore, if the leader chooses a *mixed* strategy of playing the two actions (L_1, L_2) with equal probability of 0.5, then the follower will play F_2. In this case, the leader will obtain an expected utility of $4 \times 0.5 + 3 \times 0.5 = 3.5$.

More generally, the leader has a set of N^l actions: $L_1, L_2, \ldots, L_{N^l}$ and the follower has a set of N^f actions: $F_1, F_2, \ldots, F_{N^f}$. Each action is considered as a *pure* strategy for the two players. A *mixed* strategy for the leader, $\mathbf{x} = \{x_i\}$ for $i = 1, 2, \ldots, N^l$, is a probability distribution over the N^l actions, where x_i is the probability that the leader takes action L_i. W.r.t the follower, it is sufficient to only consider pure strategies for the follower when computing the Stackelberg equilibria (Conitzer & Sandholm, 2006). The payoff matrix of the game can be determined based on joint-pure strategies of the two players. I denote by (l_{ij}, f_{ij}) the payoffs that the leader and the follower receive respectively if the leader commits to the action L_i and the follower takes action F_j. Then, the expected utility the leader receives for playing a mixed strategy $\mathbf{x}$ when the follower chooses an action F_j is computed as $U^l(\mathbf{x}, j) = \sum_i x_i l_{ij}$. On the other hand, the follower obtains an expected utility of $U^f(\mathbf{x}, j) = \sum_i x_i f_{ij}$.

2.1.1 Bayesian Stackelberg Games

In Bayesian Stackelberg games, there are multiple types of followers, each has his own payoff matrix. This game-theoretic model allows modeling the diversity of potential adversaries in real-world security domains. Essentially, a Bayesian Stackelberg game is a leader-follwer Stackelberg game in which there is a leader and a follower whose type is randomly drawn from a certain probability distribution over a set of follower types $\{1, 2, \ldots, \Lambda\}$. Specifically, each follower type λ is associated with a probability p^λ which represents the likelihood that this type may occur. In a Bayesian Stackelberg game, the leader commits to a mixed strategy given a prior knowledge over the distribution of follower types. For each type of the follower, we can denote by $(l^\lambda_{ij^\lambda}, f^\lambda_{ij^\lambda})$

the payoffs that the leader and the follower receive respectively when the leader commits to the action L_i and the follower chooses action $F_{j^\lambda}^\lambda$. In addition, we denote by $\mathbf{j} = \{j^1, j^2, \ldots, j^\Lambda\}$ a vector of actions (or pure strategies) by all types of the followers. As an extension of Stackelberg games, in Bayesian Stackelberg games, the leader's expected utility for playing a mixed strategy $\mathbf{x}$ when the follower chooses an action $F_{j^\lambda}^\lambda$ if he is of type λ for all $\lambda = 1, 2, \ldots, \Lambda$ is computed as the expected utility over all follower types, which is formulated as: $U^l(\mathbf{x}, \mathbf{j}) = \sum_\lambda p^\lambda \sum_i x_i l_{ij^\lambda}^\lambda$. On the other hand, for each follower type, λ, his expected utility is $U^{f^\lambda}(\mathbf{x}, j^\lambda) = \sum_i x_i f_{ij^\lambda}^\lambda$.

2.1.2 Strong Stackelberg Equilibrium

One of the most commonly used solution concept in Stackelberg games is Strong Stackelberg equilibrium (SSE) which assumes that the follower is a best-response player (who maximizes his expected utility) and breaks tie in favor of the leader (Breton, Alj, & Haurie, 1988). Another type of Stackelberg equilibrium is called "weak" Stackelberg equilibrium which assumes that the follower will choose the worst strategy for the leader (Breton et al., 1988) among all the best responses. While a SSE always exists, a "weak" Stackelberg equilibrium may not. Furthermore, when ties exist, the leader can always obtain a favorable outcome by arbitrarily selecting an strategy which is close to the SSE strategy, causing the follower to strictly prefer a certain strategy which benefits the leader. In fact, since a SSE always exists, this solution concept is also commonly adopted in recent works of applying Stackelberg games for solving security problems (Paruchuri, Pearce, Marecki, Tambe, Ordonez, & Kraus, 2008; Kiekintveld, Jain, Tsai, Pita, Ordez, & Tambe, 2009). Formally, a SSE can be defined as follows:

Definition 1. *Given a Bayesian Stackelberg game with payoff matrice* $\{(l_{ij^\lambda}^\lambda, f_{ij^\lambda}^\lambda)\}$ *for all follower types* $\lambda = 1, 2, \ldots, \Lambda$ *and probability distribution* $\mathbf{p}$ *over the follower types, a pair of strategies* $(\mathbf{x}, \mathbf{j})$ *forms a SSE if and only if:*

- *The leader plays a best response:*
  $U^l(\mathbf{x}, \mathbf{j}(\mathbf{x})) \geq U^l(\mathbf{x}', \mathbf{j}(\mathbf{x}')),\ \forall \mathbf{x}'$
- *The follower players a best response:*
  $U^{f^\lambda}(\mathbf{x}, j^\lambda(\mathbf{x})) \geq U^{f^\lambda}(\mathbf{x}, j^\lambda),\ \forall j^\lambda$

- *The follower breaks tie in favor of the leader:*
  $U^{l\lambda}(\mathbf{x}, j^{\lambda}(\mathbf{x})) \geq U^{l\lambda}(\mathbf{x}, j^{\lambda})$, *for all* λ *and* j^{λ} *is a best response to* $\mathbf{x}$.

2.1.3 Stackelberg Security Games

In Stackelberg security games (SSG), there is a defender who attempts to optimally allocate her limited security resources to protect a set of targets against an adversary attempting to attack one of the targets (Tambe, 2011). In SSGs, the defender commits to a *mixed* strategy first while the attacker can observe the defender's strategy and then take an action based on that observation (Von Stengel & Zamir, 2004; Korzhyk, Conitzer, & Parr, 2010). A pure strategy of the defender is an assignment of her limited resources to a subset of targets and a mixed strategy of the defender refers to a probability distribution over all possible pure strategies. The defender's mixed strategies can be represented as a marginal coverage vector over the targets (i.e., the coverage probabilities with which the defender will protect each target) (Korzhyk et al., 2010).

Denote by $\mathbf{x}$ the defender's strategy. Specifically, x_i refers to the marginal probability that the defender protects target i for $i = 1 \dots N$ where N is the number of targets. The defender can assign R resources to targets arbitrarily, as long as at most one resource is on each target (Kiekintveld et al., 2013; Yin et al., 2011). The resulting set of feasible marginal probabilities is $\mathbf{X} = \{\mathbf{x} : \ 0 \leq x_i \leq 1, \sum_i x_i \leq R\}$. In SSGs, if the adversary attacks target i, he will receive a reward R_i^a if the defender is not protecting that target, otherwise, he will receive a penalty P_i^a. Conversely, the defender will receive a penalty P_i^d in former case and a reward R_i^d in latter case. Given that the defender chooses strategy $\mathbf{x}$ and the adversary chooses to attack target i, the expected utility of the adversary, $U_i^a(\mathbf{x})$, and the defender, $U_i^d(\mathbf{x})$, are then respectively equal to

$$U_i^a(\mathbf{x}) = x_i P_i^a + (1 - x_i) R_i^a \tag{2.1}$$

$$U_i^d(\mathbf{x}) = x_i R_i^d + (1 - x_i) P_i^d \tag{2.2}$$

Finally, denote by $\mathbf{y} \in \mathbf{Y}$ the adversary's strategy where $\mathbf{Y} = \{\mathbf{y} \in \mathbf{R}^N : y_i \geq 0, \sum_i y_i = 1\}$ is the feasible region of the adversary's strategy, i.e., y_i is the probability that the adversary attacks target i. The expected utility of the defender can be computed as $\sum_i y_i U_i^d(\mathbf{x})$.

2.2 Motivating Domain: Los Angeles International Airport

This section provides an overview of the terminal protection problem at the Los Angeles International Airport (LAX) which is a motivating security domain for my human subject experiments (as explained later). LAX is the largest destination airport in the United States that serves around a hundred million passengers every year. Any terrorist attack to the airport could cause a large number of deaths as well as severe damages to critial infrastructure, hurting national and international economies. In order to protect the airport, the LAX police use a variety of defense methods, including vehicular checkpoints, police units which patrol the roads to the terminals as well as patrol inside the terminals with canines, and security screening for passengers.

Figure 2.1: LAX checkpoints

The ARMOR system (Assistant for Randomized Monitoring over Routes) (Jain, Tsai, Pita, Kiekintveld, Rathi, Tambe, & Ordóñez, 2010), which is based on Stackelberg security games, was built which aims at optimizing security resource allocation at LAX. Essentially, the system focuses on two of the security methods at LAX: (1) placing vehicle checkpoints on inbound roads to the LAX terminals and (2) scheduling patrols at the LAX terminals for canine units. The numbers of available vehicle checkpoints and canine units are limited and therefore, it is critical to optimally randomize the allocation of these security resources. Modeling this problem as a SSG,

(a) Ranger patrol (b) Poaching snare

Figure 2.2: Rangers conduct patrols over the park while poachers use snares to catch animals

the LAX police is the defender who protects terminals while terrorists are considered as the adversary that attempts to attack one of the terminals. Furthermore, there are totally eight different terminals at LAX with very different physical size, passenger loads and international/domestic flights, etc. Thus, each terminal can be represented as a target with different payoff values.

2.3 Motivating Domain: Wildlife Protection

Another major motivating domain for my research is the problem of wildlife protection. Wildlife protection is a global concern. Many species such as tigers and rhinos are in danger of extinction as a direct result of illegal harvesting (i.e., poaching). The removal of these and other species from the landscape threatens the functioning of natural ecosystems, hurts local and national economies, and has become an international security concern due to the unregulated profits of poachers flowing to terrorist organizations (on Foreign Affairs, 2015). To prevent wildlife poaching, conservation organizations attempt to protect wildlife parks with well-trained park rangers. In each time period (e.g., one month), park rangers conduct patrols within the park area to prevent poachers from capturing animals either by catching

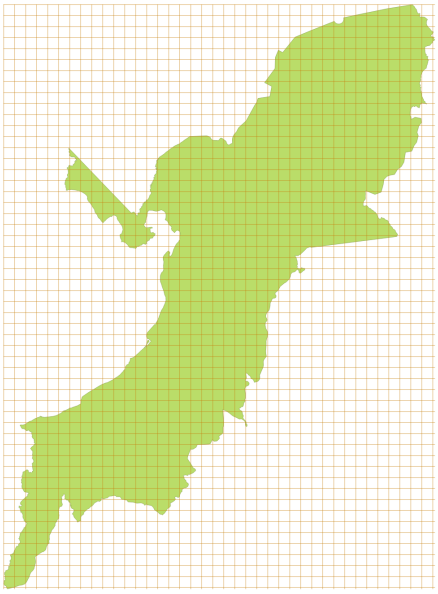


Figure 2.3: The Queen Elizabeth National Park (QENP) in Uganda

the poachers or by removing animals traps laid out by the poachers (Figure 2.2). During the rangers' patrols, poaching signs are collected and then can be used together with other domain features (e.g., animal density) to predict the poachers' behavior (Critchlow, Plumptre, Driciru, Rwetsiba, Stokes, Tumwesigye, Wanyama, & Beale, 2015; Fang et al., 2016). In essence, learning the poachers' behavior, anticipating where poachers often go for poaching, is critical for the rangers to generate effective patrols. (Montesh, 2013; Secretariat, 2013).

Previous work in security games has modeled the problem of wildlife protection as a SSG in which the rangers play in a role of the defender while the poachers are the attacker (Yang et al., 2014; Fang et al., 2015, 2016; Kar et al., 2015). The park area can be divided into a grid where each grid cell represents a target (Figure 2.3). The rewards and penalties of each target w.r.t the rangers and poachers can be determined based on domain features such as animal density, terrain slope, and distance to roads/rivers/villages, etc.

2.4 Baseline Solvers

In this section, I describe three baseline algorithms to compute the optimal strategy for the defender: 1) one algorithm computes a SSE assuming that the adversary is perfectly rational; 2) the other algorithm provides a robust defense strategy which bounds the utility loss of the defender for any potential deviation of the adversary's response from the optimal action (i.e., adversary is boundedly rational); and 3) the third algorithm focuses on addressing the worst-case scenaro of the boundedly rational adversary's response.

2.4.1 Compute Strong Stackelberg Equilibrium

The key assumption in Strong Stackelberg Equilibrium is that the adversary is a perfectly rational player who attempts to maximize his expected utility. The problem of finding the optimal

patrolling strategy for the defender against a perfectly rational attacker can be represented as the following Mixed Integer Linear Program (MILP):

$$\max_{\mathbf{x}\in\mathbf{X},\mathbf{h},\gamma} \gamma \tag{2.3}$$

$$s.t.\ \gamma \leq U_i^d(\mathbf{x}) + (1-h_i)M,\ \forall i \tag{2.4}$$

$$r \geq U_i^a(\mathbf{x}),\ \forall i \tag{2.5}$$

$$r \leq U_i^a(\mathbf{x}) + (1-h_i)M,\ \forall i \tag{2.6}$$

$$\sum_i h_i = 1,\ h_i \in \{0,1\} \tag{2.7}$$

where γ is the utility that the defender receives for playing the strategy $\mathbf{x}$ and r is the highest expected utility for the adversary over all the targets given the strategy $\mathbf{x}$. The variables $\mathbf{h} = \{h_i\}$ are binary variables which indicates whether the adversary attacks target i ($h_1 = 1$) or not ($h_i = 0$). Here, M is a very large constant which is chosen such that constraints (2.4) and (2.6) are only effective when $h_i = 1$. Constraint (2.4) ensures that the defender will receive the expected utility at target i if the adversary attacks that target. Moreover, constraints (2.5–2.7) enforce that the adversary will attack the target with highest expected utility. To that end, this MILP can be solved using any linear solver (e.g., CPLEX).

2.4.2 Compute Robust Optimal Defender Strategy

In many real-world security problems, the adversary is known to be boundedly rational; that is the adversary tends to deviate from the optimal action—he usually chooses a non-optimal target to attack. As a result, researchers have been pursuing alternative approaches to handle adversarys bounded rationality in SSGs. One leading approach is to apply robust optimization techniques to compute an optimal robust strategy for the defender. MATCH (Pita, John, Maheswaran, Tambe, & Kraus, 2012) is an exemplar of this robust approach. Essentially, MATCH computes a robust defender strategy by guaranteeing a bound on the defender's loss in her expected value if the adversary deviates from his optimal choice. More specifically, the defender's loss is constrained to be no more than a factor of β times the adversary's loss in his expected value. The key parameter β describes how much the defender is willing to sacrifice when the adversary deviates from

the optimal action. Keeping this constraint in mind, MATCH attempts to compute an optimal patrolling strategy for the defender based on the following MILP:

$$\max_{\mathbf{x}\in\mathbf{X},\mathbf{h},\eta,\gamma} \quad \gamma \tag{2.8}$$

$$\text{s.t.} \sum_{i=1}^{N} h_i = 1, h_i \in \{0,1\}, \qquad \forall i \tag{2.9}$$

$$0 \leq \eta - U_i^a(\mathbf{x}) \leq M(1-h_i) \tag{2.10}$$

$$\gamma - U_i^d(\mathbf{x}) \leq M(1-h_i) \tag{2.11}$$

$$\gamma - U_i^d(\mathbf{x}) \leq \beta\left[\eta - U_i^a(\mathbf{x})\right], \qquad \forall i \tag{2.12}$$

where γ is the defender's utility that MATCH attempts to maximize and η is highest expected utility for the adversary over all the targets. In particular, $\mathbf{h} = \{h_i\}$ are binary variables which indicate whether the adversary attacks target i ($h_i = 1$) or not ($h_i = 0$). Constraints (2.9–2.10) enforces that the adversary will attack the best target with highest expected utility (i.e., $h_i = 1$). Constraint (2.11) ensures that the defender receives the expected utility at target i if the adversary attacks that target. Finally, constraint (2.12) is the most important constraint which ensures that the defender's utility loss is no more than a factor of β times the adversarys loss in his expected value if the adversary deviates from the optimal action.

2.4.3 Compute Maximin Optimal Defender Strategy

Maximin is a standard robust algorithm which focuses on the assumption that the adversary will choose any target to attack. Essentially, Maximin attempts to maximize the defender's expected utility under the worst-case scenario of an attack by solving the following optimization problem:

$$\max_{\mathbf{x}} \gamma \tag{2.13}$$

$$s.t.\ \gamma \leq U_i^d(\mathbf{x}),\ \forall i. \tag{2.14}$$

where γ is the defender's utility that we want to maximize and constraint (2.14) enforces that the adversary will attack the target which provides the lowest expected utility for the defender.

2.5 Modeling Adversary Bounded Rationality

In addition to the robust approach, the second leading approach to address the adversary's bounded rationality is integrating models of human (adversary) decision making into the game-theoretic algorithms. One key method to evaluate the performance of behavioral models in predicting the adversary's decision making is via conducting human subject experiments.

2.5.1 The Quantal Response Behavioral Model

In SSGs, the adversary's bounded rationality is often modeled via behavior models such as Quantal Response (McFadden, 1972; McKelvey & Palfrey, 1995). In particular, the BRQR algorithm (Yang, Kiekintveld, Ordonez, Tambe, & John, 2011) subscribes to modeling human decision making; it computes an optimal strategy for the defender assuming that the adversary's response follows the QR model. The QR model predicts a stochastic distribution of the adversary response: the greater the expected value of a target the more likely the adversary will attack that target. Specifically, QR predicts the probability that the adversary will attack a target i as follows:

$$y_i = \frac{e^{\lambda U_i^a(\mathbf{x})}}{\sum_j e^{\lambda U_j^a(\mathbf{x})}} \tag{2.15}$$

QR's key parameter λ represents the level of rationality in adversary's response: as λ increases, the predicted response by the QR model converges to the optimal action of the adversary. For example, when $\lambda = 0$, the adversary attacks each target following the uniform distribution. On the other hand, when $\lambda = +\infty$, the adversary is perfectly rational.

Finally, based on QR, the BRQR algorithm attempts to compute an optimal patrolling strategy for the defender given that the adversary's response follows QR. This computation can be done by solving the following optimization problem:

$$\max_{\mathbf{x}\in\mathbf{X}} \sum_i \frac{e^{\lambda U_i^a(\mathbf{x})}}{\sum_j e^{\lambda U_j^a(\mathbf{x})}} U_i^d(\mathbf{x}) \tag{2.16}$$

$$\tag{2.17}$$

2.5.2 Human Subject Experiments

In order to evaluate the prediction accuracy of behavioral models in SSGs, the most common approach is to conduct human subject experiments. A simulated online SSG, called "The guards and treasures" has previously been used as the platform for human subject experiments (Yang et al., 2011; Pita et al., 2012). I also use it in my experiments. The game is designed to simulate the security scenario at the LAX airport, which has eight terminals that can be targeted in an attack. Figure 2.4 shows the interface of the game. Before playing the game, all the subjects are given detailed instructions about how to play. In each game, the subjects are asked to select one target to attack, given the following information: subject's reward and penalty at each target, the probability that a target will be covered by the guard, and the reward and penalty of the defender at each target.

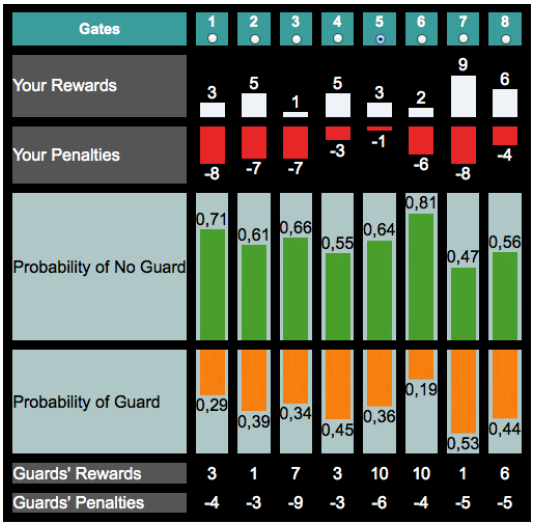


Figure 2.4: Game Interface

The parameters of behavioral models (e.g., λ of QR) are then estimated based on responses of human subjects in the experiments. Finally, given the estimated parameters, the behavioral models are integrated into computing an optimal patrolling strategy for the defender. The human subject experiments are also conducted to evaluate the solution quality of the defender's generated strategies.

Chapter 3

Related Work

3.1 Adversary Behavioral Modeling

In real-world security problems, the adversary's decision may be governed by his bounded rationality (March, 1978; Conlisk, 1996) due to effects such as task complexity and the interplay between emotion and cognition, which may cause him to deviate from the optimal action. In SSGs, different behavioral models have been proposed to capture the attacker's behavior. The Quantal Response model (QR) is one of the most popular behavioral models which attempts to predict a stochastic distribution of the attacker's responses (McFadden, 1972; McKelvey & Palfrey, 1995). In general, QR predicts the probability that the attacker will attack each target with the intuition that the higher expected utility of a target, the more likely that the attacker will choose that target. In addition to QR, Prospect Theory (Kahneman & Tversky, 1979) is another decision theory that is used to model the decision making of human adversaries (Yang et al., 2011). Recent work has followed up with extending QR and Prospect theory to predict the adversary's behavior in different security domains such as wildlife protection (Kar et al., 2015) and prevention of opportunistic crimes (Abbasi, Short, Sinha, Sintov, Zhang, & Tambe, 2015).

Recent research has therefore focused on developing algorithms for handling adversary bounded rationality in SSGs using human behavior model. In particular, BRQR (Yang et al., 2011) is a leading algorithm which subscribes to modeling human decision making; it computes an optimal strategy for the defender assuming that the adversary's response follows QR. In contrast, instead of using a behavior model, a recent model-free algorithm MATCH (Pita et al., 2012) computes a robust defender strategy by guaranteeing a bound on the defender's loss in her

expected value if the adversary deviates from his optimal choice. A comparison of these two algorithms by (Pita et al., 2012), involving 104 simulated security settings, showed that MATCH significantly outperforms BRQR. This result leads to an open research question on whether there is any value in modeling the adversary's behavior in security games.

In addition to QR, there are other lines of research which focus on building models of criminal behavior in urban crime (De Bruin, Cocx, Kosters, Laros, Kok, et al., 2006; Nath, 2006; Oatley, Ewart, & Zeleznikow, 2006; Zhang, Sinha, & Tambe, 2015) or opponent behavior in poker (Ganzfried & Sandholm, 2011; Southey, Bowling, Larson, Piccione, Burch, Billings, & Rayner, 2012). However, these models are specifically designed for these domains, which rely on the complete past crime/game data as well as intrinsic domain characteristics. Another line of research focuses on adversarial plan recognition (Avrahami-Zilberbrand & Kaminka, 2014), which can be applied for computer intrusion detection and detection of anomalous activities, etc. This line of work does not learn model parameters as well as do any patrol planning.

3.2 Solutions for Wildlife Protection

Previous work focuses on computing the optimal patrolling strategy for the rangers given that poachers' behavior is predicted based on existing adversary behavioral models (Yang et al., 2014; Fang et al., 2015, 2016; Kar et al., 2015). However, these models make several limiting assumptions, including (a) all poaching signs (e.g., snares) are perfectly observable by the rangers; (b) poachers' activities in one time period are independent of their activities in previous or future time periods; (c) the number of poachers is known. To understand the limiting nature of these assumptions, consider the issue of observability. The rangers' capability of making observations over a large geographical area is limited. For example, the rangers usually follow certain paths/trails to patrol; they can only observe over the areas around these paths/trails which means that they may not be able to make observations in other further areas. In addition, in areas such as dense forests, it is difficult for the rangers to search for snares. As a result, there may be still poaching activities happening in areas where rangers did not find any poaching sign. Therefore, relying entirely on the rangers' observations would lead to an inaccurate prediction of the poachers' behavior, hindering the rangers' patrol effectiveness. Furthermore, when modeling the poachers' behavior, it

is critical to incorporate important aspects that affect the poachers' behavior including time dependency of the poachers' activities and patrolling frequencies of the rangers. Lastly, the rangers are unaware of the total number of attackers in the park.

In ecology research, while previous work mainly focused on estimating the animal density (MacKenzie, Nichols, Lachman, Droege, Andrew Royle, & Langtimm, 2002), there are a few works which attempt to model the spatial distribution of the economic costs/benefits of illegal hunting activities in the Serengeti national park (Hofer, Campbell, East, & Huish, 2000) or the threats to wildlife and how these change over time in QENP (Critchlow et al., 2015). However, these models also have several limitations. First, the proposed models do not consider the time dependency of the poachers' behaviors. These models also do not consider the effect of the rangers' patrols on poaching activities. Furthermore, the prediction accuracy of the proposed models is not measured. Finally, these works do not provide any solution for generating the rangers' patrolling strategies with a behavioral model of the poachers.

3.3 Robust Stackelberg Games

Maximin-based method. Maximin-based algorithms for addressing uncertainties in SSGs focus on maximizing the defender's utility against the worst case of uncertainties. All previous works following this approach attempt to compartmentalize uncertainties and apply different algorithms to only address a particular type of uncertainty. One simple robust algorithm for dealing with uncertainty in the adversary's bounded rationality is Maximin, which assumes that the adversary can choose any arbitrary strategy Given Maximin can generate extremely conservative strategies, BRASS (Pita et al., 2009) provided an advance to handle uncertainty due to adversary bounded rationality: BRASS assumes that the adversary can attack any targets which provide within ϵ of the maximum expected utility for the adversary where ϵ is a given constant.

A later robust algorithm, RECON (Yin et al., 2011), shifted focus away from bounded rationality. RECON only deals with uncertainty in the defender's execution and the adversary's observation. Kiekintveld et al,, on the other hand, proposed a new robust algorithm called ISG which only focused on uncertainty in the adversary's payoffs (Kiekintveld et al., 2013). The most

recent algorithm, monotonic maximin (Jiang et al., 2013), attempts to deal with uncertainty exclusively in the adversary' bounded rationality. In particular, it attempts to generalize the Quantal Response model of bounded rationality which was proposed in behavioral economics and has been applied to SSGs. In many cases such as counter-terrorism domains, the defender does not have sufficient data on the adversary's behaviors to accurately estimate the parameters of QR models. Instead, monotonic maximin assumes that the adversary can choose any strategy that has the following monotonicity property (which is satisfied by all known variants of QR models): the higher the expected utility of a target, the more likely the adversary will attack that target.

The above discussion summarizes *major* thrusts to handle robustness in SSGs as reported in the literature that I unify in my work. Recent research has explored addressing other types of uncertainties (An, Brown, Vorobeychik, & Tambe, 2013; An, Tambe, Ordonez, Shieh, & Kiekintveld, 2011), but these have yet to provide robust algorithms. Furthermore, my work is exactly focused on trying to remedy such salami-slicing of handling of uncertainty.

Minimax regret-based method. Minimax regret is a less conservative and powerful robust method for handling uncertainties. In particular, it attempts to minimize the maximum "regret" or distance in terms of the utility loss of a decision (e.g., defender's strategy) from the actual optimal decision for any instance within the uncertainty. Minimax regret has been used in a variety of settings, including game-theoretic ones, especially those involving utility function uncertainty (Salo & Hamalainen, 2001; Boutilier, Patrascu, Poupart, & Schuurmans, 2006; Hyafil & Boutilier, 2004; Renou & Schlag, 2010). It has also proven to be a very effective driver of preference elicitation (Braziunas & Boutilier, 2010; Boutilier, 2013).

3.4 Bayesian Stackelberg Games

Bayesian game is a model of probabilistic uncertainty in games (Harsanyi, 1967), and has been applied to SSGs for modeling the adversary's payoff, the adversary's observation and the defender's execution, (Kiekintveld, Marecki, & Tambe, 2011; Yin & Tambe, 2012a). Furthermore, (Yin & Tambe, 2012a) handles a combination of such uncertainties in SSGs by discretizing this continuous uncertainty space and solving the resulting Bayesian Stackelberg games with discrete

follower types. Thus, its solution quality depends on the number of samples. Additionally, (Yin & Tambe, 2012a) *does not integrate* uncertainty due to adversary's bounded rationality.

Chapter 4

Modeling Attacker Decision Making

Researchers have been pursuing alternative approaches to handle adversary's bounded rationality in SSGs (Pita, Jain, Tambe, Ordóñez, & Kraus, 2010; Yang et al., 2011; Pita et al., 2012). Two competing approaches have emerged to address human bounded rationality in SSGs. One approach integrates models of human decision-making into algorithms for computing an optimal strategy for the defender; the other adopts robust optimization techniques to intentionally avoid adversary modeling. The BRQR algorithm (Yang et al., 2011), based on modeling adversary decision-making with the Quantal Response (QR) (McKelvey & Palfrey, 1995) model, leads to significantly better defender strategies than any previous leading contenders. However, the more recent robust algorithm MATCH (Pita et al., 2012) outperforms BRQR. It is indeed surprising that despite the long history of modeling success of QR, MATCH still performs better, even when significant amount of data were used to tune the key parameter in QR and no tuning was done to MATCH's key parameter.

Thus, there is now an important open question of whether there is any value in adversary modeling in SSGs. My first contribution in answering this question builds on the significant support for QR (Haile, Hortacsu, & Kosenok, 2008; Choi, Gale, & Kariv, 2012): I hypothesize that QR's stochastic response is crucial in building a human decision-making model. Where I part company with the original QR model, however, is in its assumption that human stochastic response is based on expected value. Instead, I propose a new model based on integration of a novel *Lens model* into QR, called the LensQR model.[1] I show that the LensQR model, given learned parameters (from limited data), has superior predictive power compared to the QR model. I then derive the

[1]The new model is original called SUQR; I adopt the new name LensQR in recognition of its two key components: the Lens utility and the QR model. The resulting algorithm, SU-BRQR, is also updated with the new name LensBRQR.

LensBRQR algorithm, similar to BRQR, to compute the defender strategy assuming the adversary response follows the LensQR model. I evaluate LensBRQR's performance by conducting two sets of experiments using an online game with Amazon Mechanical Turk (AMT) workers and show that: (i) LensBRQR significantly outperforms MATCH in previously used settings; (ii) LensBRQR usually outperforms (and always performs at least as well as) improved versions of MATCH such as ones offering it the same Lens functions or tuning its key parameter.

LensBRQR's parameters were learned from previously available (albeit limited) game data; I now test LensBRQR in domains without the benefit of such *a-priori* data. Indeed, while some domains of SSG -based application, e.g., deterring fare evasion (Yin, Jiang, Johnson, Tambe, Kiekintveld, Leyton-Brown, Sandholm, & Sullivan, 2012) or forest protection (Johnson, Fang, & Tambe, 2012), could provide significant amounts of data to tune LensBRQR, would I be better off with MATCH or other algorithms in applications that do not? My second contribution answers this question by conducting experiments with security intelligence experts, where I do not have any previous modeling data. These experts, who serve as proxies for real-world adversaries, serve in the best Israeli Intelligence Corps unit or are alumna of that unit, and are found to be more rational than the AMT workers. Against these experts, LensBRQR with its earlier learned parameters, significantly outperforms both an algorithm assuming perfect adversary rationality (Paruchuri et al., 2008) and (to a more limited extent) MATCH. Finally, my third contribution tests LensBRQR in a new large game with AMT workers. I show that LensBRQR with previously learned parameters still outperforms MATCH; and learning from more data, LensBRQR performance can be further improved.

4.1 The LensQR Model

Essentially, the new LensQR model is based on a novel integration of the *Brunswik Lens model* into QR. The Lens model is a well-known model which has been extensively used to study human judgements accross different decision-making domains such as medicine, business, education, and psychology over five decades (Karelaia & Hogarth, 2008; Kaufmann & Athanasou, 2009; Grove & Meehl, 1996; Dawes, Faust, & Meehl, 1989). Previous studies of the Lens model show

that linear models can provide good high-level representations of human judgements. Essentially, the Lens model suggests that human judgments depend on a linear combination of multiple observable features. Recall that in an SSG, the information presented to the human subject for each choice includes (Section 2.5.2): the marginal coverage on target i (x_i); the subject's reward and penalty (R_i^a,P_i^a); the defender's reward and penalty (R_i^d, P_i^d). Therefore, based on the Lens model, I measure the utility of an attacker as a linear combination of these three important attributes, which can be formulated as follows:

$$\hat{U}_i^a = w_1 x_i + w_2 R_i^a + w_3 P_i^a \tag{4.1}$$

where (w_1, w_2, w_3) are the model parameters which indicate the importance of the corresponding features w.r.t the attacker's decision making. The novelty of applying the Lens model is that I can combine the values (rewards/penalty) and *probabilities*. While unconventional at first glance, as shown later, this model actually leads to higher prediction accuracy than the classic expected value function. A possible explanation for that is that humans might be driven by simple heuristics in their decision making. Indeed, several studies in other research domains have demonstrated the prediction power of simple combination of features (Meehl, 1963; Dawes, 1979) while complex models could possibly lead to over-fitting issues (Meehl, 1963). Other alternatives to this 3-feature Lens utility function are feasible, e.g., including all the information presented to the subjects ($\hat{U}_i^a = w_1 x_i + w_2 R_i^a + w_3 P_i^a + w_4 R_i^d + w_5 P_i^d$), which I discuss later.

I modify the QR model by replacing the classic expected value function with the Lens function, leading to the LensQR model. In the LensQR model, the probability that the adversary chooses to attack target i, y_i, is given by:

$$y_i = \frac{e^{\lambda \hat{U}_i^a}}{\sum_j e^{\lambda \hat{U}_j^a}} = \frac{e^{\lambda(w_1 x_i + w_2 R_i^a + w_3 P_i^a)}}{\sum_j e^{\lambda(w_1 x_j + w_2 R_j^a + w_3 P_j^a)}} \tag{4.2}$$

Given that the attacker's responses follows LensQR, the problem of finding the optimal strategy for the defender can therefore be formulated as follows:

$$\begin{aligned} &\max_{\mathbf{x}} \sum_{i=1}^{N} y_i \left[x_i R_i^d + (1 - x_i) P_i^d \right] \\ &\text{s.t.} \sum_{i=1}^{N} x_i \leq K, 0 \leq x_i \leq 1 \end{aligned} \tag{4.3}$$

where K is the number of security resources. Here, the objective is to maximize the defender's expected value given that the adversary chooses to attack each target with a probability according to the LensQR model. Constraint (4.3) ensures that the coverage probabilities on all the targets satisfy the resource constraint. Given that this optimization problem is similar to BRQR, I use the same approach as BRQR to solve it (Yang et al., 2011). I refer to the resulting algorithm as LensBRQR.

4.1.1 Learning LensQR Parameters

Without loss of generality, I set $\lambda = 1$. I employ Maximum Likelihood Estimation (MLE) (Hastie, Tibshirani, & Friedman, 2009) to learn the parameters (w_1, w_2, w_3). Given the defender strategy $\mathbf{x}$ and K samples of the players' choices, the log-likelihood of (w_1, w_2, w_3) is given by:

$$logL(w_1, w_2, w_3|\mathbf{x}) = \sum_{k=1}^{K} \log\left[y_{i_k}(w_1, w_2, w_3|\mathbf{x})\right]$$

where i_k is the target that is chosen in sample k and y_{i_k} $(w_1, w_2, w_3|\mathbf{x})$ is the probability that the adversary chooses the target i_k. Let K_i be the number of subjects attacking target i. Then the log-likelihood function can be reformulated as follows:

$$logL(w_1, w_2, w_3|\mathbf{x}) = \sum_{i=1}^{N} K_i \log\left[y_i(w_1, w_2, w_3|\mathbf{x})\right]$$

Combining with Equation 4.2, I obtain:

$$\begin{aligned} &logL(w_1, w_2, w_3|\mathbf{x}) \qquad (4.4)\\ &= w_1\left(\sum_{i=1}^{N} K_i x_i\right) + w_2\left(\sum_{i=1}^{N} K_i R_i^a\right) + w_3\left(\sum_{i=1}^{N} K_i P_i^a\right) - K\log\left(\sum_{i=1}^{N} e^{w_1 x_i + w_2 R_i^a + w_3 P_i^a}\right) \end{aligned}$$

In essence, $logL(w_1, w_2, w_3|\mathbf{x})$ is a concave function: I can show that the Hessian matrix of $logL(w_1, w_2, w_3|\mathbf{x})$ is negative semi-definite. Thus, this function has an unique local maximum point and I can hence use a convex optimization solver to compute the optimal weights (w_1, w_2, w_3), e.g., *fmincon* in Matlab.

4.1.2 Prediction Accuracy of the LensQR Model

As in some real-world security environments, I would want to learn parameters of my LensQR model based on limited data. To that end, I used a training set of five payoff structures and two

Table 4.1: Prediction Accuracy

QR	3-parameter LensQR	5-parameter LensQR
8%	51%	44%

algorithms MATCH and BRQR (10 games in total) from (Pita et al., 2012) to learn the parameters of the Lens utility function and the alternatives. In total, 33 human subjects played these 10 games using the setting of 8 targets and 3 guards from my on-line game. The parameters that I learnt are: (w_1, w_2, w_3)=$(-9.85, .37, .15)$ for the 3-parameter LensQR function; and $(w_1, w_2, w_3, w_4, w_5)$ = $(-8.23, .28, .12, .07, .09)$ for the 5-parameter function.

I ran a Pearson's chi-squared goodness of fit test (Greenwood & Nikulin, 1996) in a separate test set which includes 100 payoff structures in (Pita et al., 2012) to evaluate the prediction accuracy of the two proposed models as well as the classic QR model. The test examines whether the predicted distribution of the players' choices fits the observation. I set $\lambda = .76$ for QR model, the same as what was learned in (Yang et al., 2011). The percentages of the payoff structures that fit the predictions of the three models (statistical significance $\alpha = 0.05$) are displayed in Table 4.1. The table clearly shows that the new LensQR model (with the Lens utility function in Equation 4.1 predicts the human behavior more accurately than the classic QR model. In addition, even with more parameters, the prediction accuracy of the 5-parameter LensQR model does not improve. Given this result and my 3-parameter model demonstrated superiority (as I will show in my Experiments section), I leave efforts to further improve the LensQR model for future work.

4.2 Improving MATCH

Since LensQR better predicts the distribution of the subjects' choices than the classic QR, and as shown later, LensBRQR outperforms MATCH, it is natural to investigate the integration of the Lens utility function into MATCH. In particular, I replace the expected value of the adversary with

Lens utility function. Therefore, the adversary's loss caused by his deviation from the optimal solution is measured with regard to the Lens utility function.

$$\max_{\mathbf{x},\mathbf{h},\eta,\gamma} \quad \gamma \tag{4.5}$$

$$\text{s.t.} \sum_{i=1}^{N} x_i \leq R, 0 \leq x_i \leq 1, \qquad \forall t \tag{4.6}$$

$$\sum_{i=1}^{N} h_i = 1, h_i \in \{0,1\}, \qquad \forall i \tag{4.7}$$

$$0 \leq \eta - (w_1 x_i + w_2 R_i^a + w_3 P_i^a) \leq M(1-h_i) \tag{4.8}$$

$$\gamma - \left[x_i R_i^d + (1-x_i) P_i^d\right] \leq M(1-h_i) \tag{4.9}$$

$$\gamma - \left[x_i R_i^d + (1-x_i) P_i^d\right] \leq \beta \left[\eta - (w_1 x_i + w_2 R_i^a + w_3 P_i^a)\right], \qquad \forall i \tag{4.10}$$

I refer to this modified version as LensMATCH, which is shown in Equations (4.5)-(4.10), that attempts to maximize the defender's expected value. In particular, h_i represents the adversary's target choice, η represents the maximum Lens utility for the adversary. In addition, γ represents the expected value for the defender if the adversary responds optimally and M is a large constant.

Constraint (4.8) finds the optimal strategy (target) for the adversary. In constraint (4.9), the defender's expected value is computed when the attacker chooses his optimal strategy. The key idea of LensMATCH is in constraint (4.10). Essentially, it guarantees that the loss of the defender's expected value caused by adversary's deviation is no more than a factor of β times the loss of the adversary's subjective utility.

Selecting β for MATCH: In MATCH, the parameter β is the key that decides how much the defender is willing to lose if the adversary deviates from his optimal strategy. Pita et al. set β to 1.0, leaving its optimization for future work. I propose a method to estimate β based on the LensQR model which is outlined in Algorithm 1.

In this method, K values of β are uniformly sampled within the range (0, MaxBeta). For each sampled value of β, the optimal strategy $\mathbf{x}$ for the defender is computed using MATCH. Given this mixed strategy $\mathbf{x}$, the defender's expected value, γ, is computed assuming that the adversary will respond stochastically according to the LensQR model. The β leading to the highest defender expected value is chosen. In practice, I set MaxBeta to 5, to provide an effective bound on the defender loss, given that penalties/rewards of both players range from -10 to 10; and K to 100,

Algorithm 1: SELECTING β

Initialize $\gamma^* \leftarrow -\infty$;
for *k = 1 to K* **do**
  $\beta \leftarrow$ *Sample*$([0, \texttt{MaxBeta}], k)$, $\mathbf{x} \leftarrow$ MATCH(β);
  $\gamma \leftarrow \sum_i y_i U_i^d$;
  if $\gamma \geq \gamma^*$ **then**
    $\gamma^* \leftarrow \gamma, \beta^* \leftarrow \beta$;
  end
end
return (β^*, γ^*);

which gives a grid size of 0.05 for β for the range of $(0, 5)$. I refer to the algorithm with *carefully selected* β as MATCHBeta.

4.3 Experimental Results

The tested algorithms in my experiments include: LensBRQR, MATCH, LensMATCH, MATCH-Beta, LensMATCHBeta, i.e., MATCH embedded with both Lens utility and selecting β, and DOBSS, i.e., a robust algorithm against perfectly rational opponents.

4.3.1 Results with AMT Workers, 8-target Games

My first experiment compares LensBRQR against MATCH and its improvements, in the setting where I learned the parameters of the LensQR model, i.e., the 8-target and 3-guard game with the AMT workers. In this 8-target game setting, for each game, my reported average is over at least 45 human subjects. The experiments were conducted on the AMT system. When two algorithms are compared, I ensured that identical human subjects played both on the same payoff structures. Participants were paid a base amount of US $\$1.00$. In addition, each participant was given a bonus based on their performance in the games to motivate them. Similar to (Pita et al., 2012)'s work, I ensured that players were not choosing targets arbitrarily by having each participant play two extra trivial games (i.e., games in which there is a target with the highest adversary reward and lowest adversary penalty and lowest defender coverage probability). Players' results were removed if they did not choose that target.

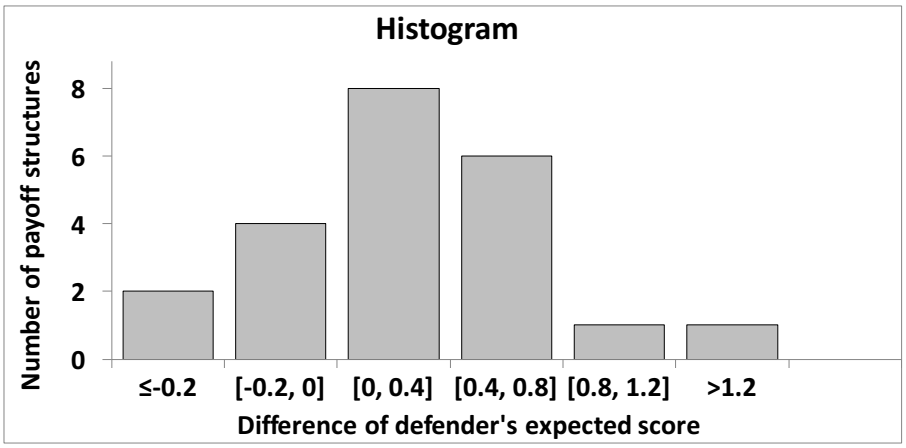
I generated the payoff structures based on covariance games in GAMUT (Nudelman, Wortman, Shoham, & Leyton-Brown, 2004). In covariance games, I can adjust the covariance value

$r \in [-1, 1]$ to control the correlation between rewards of players. I first generate 1000 payoff structures with r ranging from -1 to 0 by 0.1 increments (100 payoff structures per value of r). Then, for each of the 11 r values, I select 2 payoff structures ensuring that the strategies generated by each candidate algorithm (e.g., LesBRQR and versions of MATCH) are not similar to each. One of these two has the maximum and the other has the median sum of 1-norm distances between defender strategies generated by each pair of the algorithms. This leads to a total of 22 payoff structures. By selecting the payoffs in this way, I explore payoff structures with different levels of the 1-norm distance between generated strategies so as to obtain accurate evaluations with regard to performance of the tested algorithms. I evaluate the statistical significance of my results using the bootstrap-t method (Wilcox, 2002).

4.3.1.1 LensBRQR vs MATCH

This section evaluates the impact of the new subjective utility function via a head-to-head comparison between LensBRQR and MATCH. In this initial test, the β parameter of MATCH was set to 1.0 as in (Pita et al., 2012). Figure 4.1a first shows all available comparison results for completeness (without regard to statistical significance). More specifically, I show the histogram of the difference between LensBRQR and MATCH in the average defender expected reward over all the choices of the participants. The x-axis shows the range of this difference in each bin and the y-axis displays the number of payoff structures (out of 22) that belong to each bin. For example, in the third bin from the left, the average defender expected value achieved by LensBRQR is higher than that achieved by MATCH, and the difference ranges from 0 to 0.4. There are 8 payoffs that fall into this category. Overall, LensBRQR achieves a higher average expected defender reward than MATCH in the 16 out of the 22 payoff structures.

In Figure 4.1b, the second column shows the number of payoffs where LensBRQR outperforms MATCH with statistical significance ($\alpha = .05$). The number of payoff structures where MATCH is better than LensBRQR with statistical significance is shown in the fourth column. In the 22 payoff structures, LensBRQR outperforms MATCH 13 times with statistical significance while MATCH defeats LensBRQR only once; in the remaining 8 cases, no statistical significance is obtained either way. This result stands in stark contrast to (Pita et al., 2012)'s result and directly answers the question I posed at the beginning of this paper: there is indeed value to integrating



(a) All comparison data

	LensBRQR	Draw	MATCH
$\alpha = .05$	13	8	1

(b) Results with statistical significance

Figure 4.1: LensBRQR vs MATCH, AMT workers, 8 targets

models of human decision making in computing defender strategies in SSGs, but use of LensQR rather than traditional QR models is crucial.

Table 4.2: Performance comparison, $\alpha = .05$

	LensMATCH	MATCHBeta	LensMATCHBeta
MATCH	3, 11	1, 6	1, 8
LensBRQR	8, 2	8, 2	5, 3

4.3.1.2 LensBRQR vs Improved MATCH

In Table 4.2, I compare MATCH and LensBRQR against the three improved versions of MATCH: LensMATCH, MATCHBeta, and LensMATCHBeta (i.e., MATCH with both the Lens utility function and the selected β) when playing my 22 selected payoff structures. Here, I only report results that hold with statistical significance ($\alpha = .05$). The first number in each cell in Table 4.2 shows the number of payoffs (out of 22) where the row algorithm obtains a higher average defender expected reward than the column algorithm; the second number shows where the column algorithm outperforms the row algorithm. For example, the second row and second column shows that MATCH outperforms LensMATCH in 3 payoff structures with statistical significance while LensMATCH defeats MATCH in 11. Table 4.2 shows that the newer versions of MATCH

achieve a significant improvement over MATCH. Additionally, LensBRQR retains a significant advantage over both LensMATCH and MATCHBeta. For example, LensBRQR defeats LensMATCH in 8 out of the 22 payoff structures with statistical significance, as shown in Table 4.2; in contrast, LensMATCH is better than LensBRQR only twice.

Although LensBRQR in this case does not outperform LensMATCHBeta to the extent it does against MATCH (i.e., LensBRQR performs better than LensMATCHBeta only 5 times with statistical significance while LensMATCHBeta is better than LensBRQR thrice (Table 4.2)), LensBRQR remains the algorithm of choice for the following reasons: (a) LensBRQR does perform better than LensMATCHBeta in more cases with statistical significance; (b) selecting the β parameters in LensMATCHBeta can be a significant computational overhead for large games given that it requires testing many values of β. Thus, I could just prefer LensBRQR .

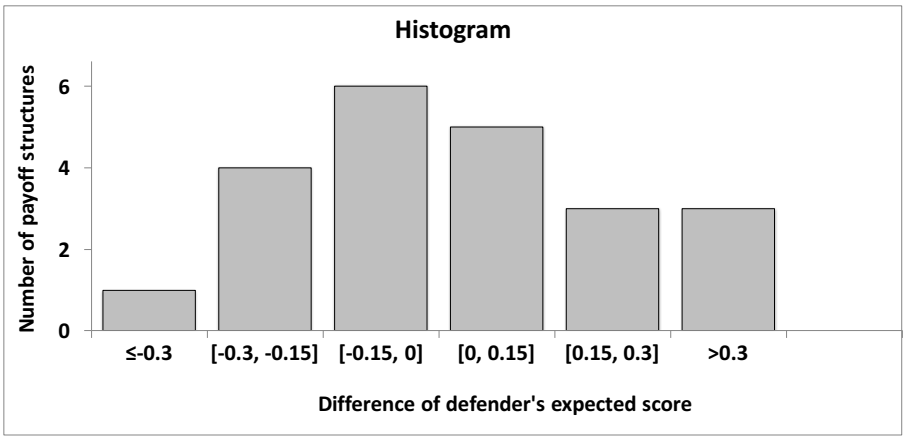
4.3.2 Results with New Experimental Scenarios

All previous experiments are based on the 8-target and 3-guards game, which were motivated by the LAX security scenario (Tambe, 2011). In addition, the games have been played by AMT workers or college students. To evaluate the performance of the LensQR model in new scenarios, I introduce two new experimental settings: in one the experiments are conducted against a new type of human adversary, i.e., security intelligence experts; and in the other, I change the game to 24 targets and 9 guards.

4.3.2.1 Security Intelligence Experts, 8-target games

In this section, I evaluate my algorithm with security intelligence experts who serve in the best Israeli Intelligence Corps unit or are alumna of that unit. My purpose is to examine whether LensBRQR will work when I so radically change the subject population to security experts. I use the same 22 payoff structures and the same Lens utility function as in the previous experiment with AMT workers. Each result below is averaged over decisions of 27 experts.

LensBRQR vs DOBSS. DOBSS (Paruchuri et al., 2008) is an algorithm for optimal defender strategies against perfectly rational opponents. DOBSS performed poorly in 8-target games against AMT workers(Pita et al., 2010; Yang et al., 2011). However, would DOBSS perform better in comparison to LensBRQR against security experts? My results show that LensBRQR is



(a) All comparison data

	LensBRQR	Draw	MATCH
$\alpha = .05$	6	13	3

(b) Results with statistical significance

Figure 4.2: LensBRQR vs MATCH, security experts

better than DOBSS in all 22 tested payoff structures; 19 times with statistical significance. Thus, even these experts did not respond optimally (as anticipated by DOBSS) against the defender's strategies.

LensBRQR vs MATCH. Figure 4.2a shows that LensBRQR obtains a higher expected defender reward than MATCH in 11 payoff structures against my experts. Furthermore, LensBRQR performs better than MATCH in 6 payoff structures with statistical significance while MATCH is better than LensBRQR only in 3 payoff structures with statistical significance (Figure 4.2b). These results still favor LensBRQR over MATCH, although not as much as when playing against AMT workers (as in Figure 4.1).

Nonetheless, what is crucially shown in this section is that changing the subject population to security experts does not undermine LensBRQR completely; in fact, despite using parameters from AMT workers, LensBRQR is still able to perform better than MATCH. I re-estimate the parameters of the Lens function using the data of experts. The result is: $w_1 = -11.0, w_2 = 0.54$, and $w_3 = 0.35$. This result shows that while the experts evaluated all the criteria differently from the AMT workers they gave the same importance level to the three parameters. Because of limited access to experts, I could not conduct experiments with these re-estimated parameters; I will show the impact of such re-estimation in my next experimental setting.

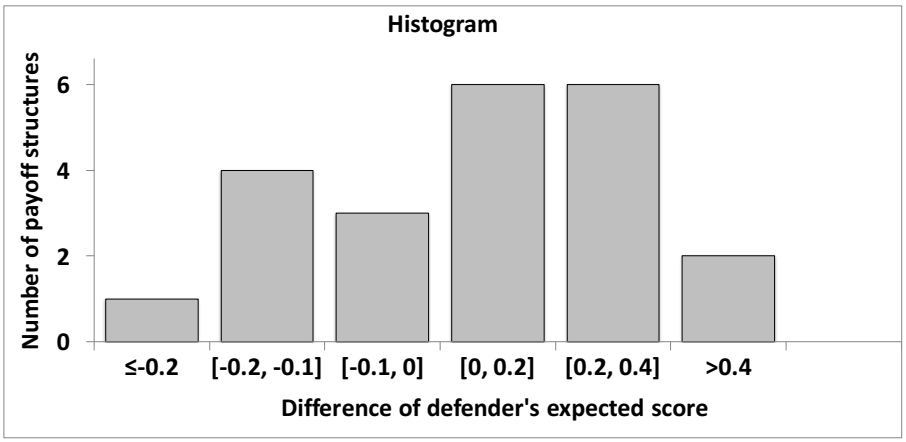
Bounded Rationality of Human Adversaries. I now compare the AMT workers and security experts using the traditional metric of "rationality level" of the QR model. To that end, I revert to the QR-model with the expected value function to measure how close these players are to perfect rationality. In particular, I use QR's λ parameter as a criterion to measure their rationality. I use all the data from AMT workers as well as experts on the chosen 22 games in previous experiments to learn the λ parameter. I get $\lambda = 0.77$ with AMT workers and $\lambda = 0.91$ with experts. This result implies that security intelligence experts tend to be more rational than AMT workers (the higher the λ, the closer the players are to perfect rationality). Indeed, in 34 of 44 games, experts obtains a higher expected value than AMT workers. Out of these, their expected value is higher than AMT workers 9 times with statistical significance while AMT workers is higher only once ($\alpha = .05$). Nonetheless, the λ for experts of 0.91 suggests that the experts do not play with perfect rationality (perfect rational $\lambda = \infty$).

4.3.2.2 AMT Workers, 24-target Games

In this section, I focus on examining the performance of the algorithms in large games, i.e., 24 targets and 9 defender resources. I expect that the human adversaries may change their behaviors because of tedious evaluation of risk and benefit for each target. Two algorithms were tested: LensBRQR, MATCH. I first run experiments with the new Lens utility function learned previously using the data of the 8-target game.

LensBRQR vs MATCH with Parameters Learned from the 8-target Games. Figure 4.3a shows that LensBRQR obtains a higher average defender expected value than MATCH in 14 out of 22 payoff structures while MATCH is better than LensBRQR in 8 payoff structures. These averages are reported over 45 subjects. In addition, as can be seen in Figure 4.3b, LensBRQR performs better than MATCH with statistical significance 8 times while MATCH outperforms LensBRQR 3 times. While LensBRQR does perform better than MATCH, its superiority over MATCH is not as much as it was in previous 8-target games.

I can hypothesize based on these results that the learned parameters of the 8-target games do not predict human behaviors as well in the 24-target games. Therefore, I re-estimate the values of the parameters of the Lens utility function using the data of the previous experiment in the 24-target games. The training data contains 388 data points. This re-estimating results in



(a) All comparison data

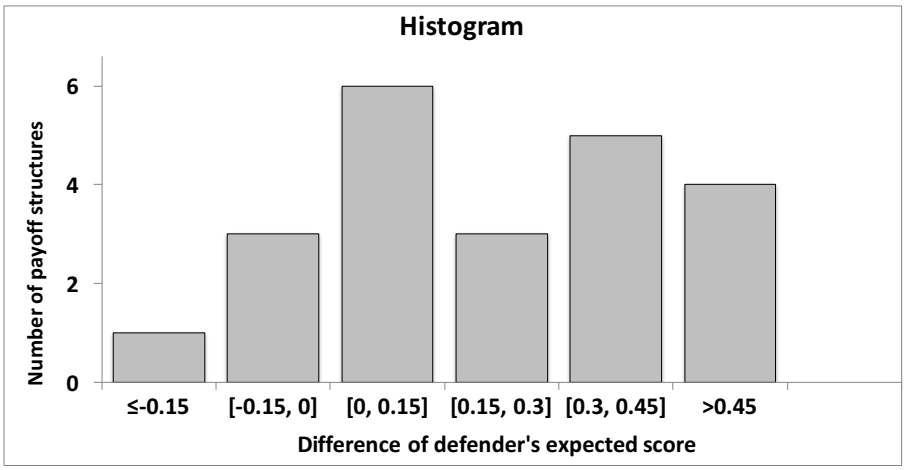
	LensBRQR	Draw	MATCH
$\alpha = .05$	8	11	3

(b) Results with statistical significance

Figure 4.3: LensBRQR vs MATCH, 24 targets, original

$w_1 = -15.29, w_2 = .53, w_3 = .34$. Similar to the experts case, the weights in 24-target games are different from the ones in 8-target games but their order of importance is the same.

LensBRQR vs MATCH with Re-estimated Parameters. In this experiment, I evaluate the impact of the new Lens utility function with the re-estimated parameters on the performance of LensBRQR in comparison with MATCH. Figure 4.4a shows that LensBRQR outperforms MATCH in 18 payoff structures while MATCH defeats LensBRQR in only 4 cases. Moreover, it can be seen in Figure 4.4b that LensBRQR defeats MATCH with statistical significance 11 times while MATCH defeats LensBRQR only once with statistical significance. In other words, the new weights of the Lens utility function indeed help improve the performance of LensBRQR. This result demonstrates that a more accurate Lens utility function can help improve LensBRQR's performance.



(a) All comparison data

	LensBRQR	Draw	MATCH
$\alpha = .05$	11	10	1

(b) Results with statistical significance

Figure 4.4: LensBRQR vs MATCH, 24 targets, re-estimated

4.4 Summary

This chapter demonstrates the importance of modeling human adversary decision making in SSGs using a novel integration of the *Lens utility function* with the Quantal Response model. The resulting new behavioral model is called LensQR. Through extensive experiments, the chapter provides the following contributions: (i) I show that my LensBRQR algorithm, which involves the new LensQR behavioral model, significantly outperforms both MATCH and its improved versions; (ii) I are the first to present experimental results with security intelligence experts, and find that even though the experts are more rational than AMT workers, LensBRQR performs better than its competition against the experts; (iii) I show the advantage of LensBRQR in a new game setting and demonstrate that additional data can further boost the performance of LensBRQR over MATCH.

Chapter 5

Preventing Poaching in Wildlife Protection

As mentioned in Chapter 3, previous work has began to apply SSGs for wildlife protection (Yang et al., 2014; Fang et al., 2015, 2016), wherein existing behavioral models in SSGs are used to predict the poachers' behavior. However, since these models (including LensQR introduced in the previous chapter) were developed with standard SSGs in mind, they have several limiting assumptions when applying for predicting poachers' behavior, such as (a) all poaching signs are perfectly observable by the rangers; (b) poachers' activities in one time period are independent of their activities in previous or future time periods; (c) the number of poachers is known. In this chapter, I introduce a new behavioral model of the poachers which addresses all these limitations.

5.1 Overview

Recently, an SSG-based patrolling decision-aid called PAWS has been deployed to protect wildlife in south-east Asia (Fang et al., 2016). PAWS focuses on generating effective patrols for the rangers, taking into account the complex topographic conditions of Asian forests. Despite its successful application, PAWS is known to suffer from several limitations. First, PAWS relies on my behavior model, LensQR, which is shown to have several limiting assumptions when applying in wildlife protection (Fang et al., 2016). Second, since LensQR has traditionally only relied on three or four domain attributes in its modeling, it has not been able to provide a detailed analysis of the impact of environmental and terrain features on poacher behavior, and thus such analysis of real-world data has been lacking in the literature. Third, richer adversary models would also require new patrol generation algorithms that improve upon what is used in PAWS.

To that end, I have built a new predictive anti-poaching tool, CAPTURE (Comprehensive Anti-Poaching tool with Temporal and observation Uncertainty REasoning) which attempts to address all aforementioned limitations in PAWS while providing the following three key contributions. My first area of contribution relates to CAPTURE's addressing LensQR's limitations in modeling adversary behavior. More specifically, CAPTURE introduces a new behavioral model which takes into account the rangers' imperfect detection of poaching signs, called LensQR-Poacher. Additionally, LensQR-Poacher incorporates the dependence of the poachers' behavior on their activities in the past into the component for predicting the poachers' behavior. Moreover, I adopt logistic models to formulate the two components of LensQR-Poacher. This enables capturing the aggregate behavior of attackers without requiring a known number of poachers. Finally, CAPTURE considers a richer set of domain features in addition to the three/four features used in LensQR in analyzing the poachers' behavior. Second, I provide two new heuristics to reduce the computational cost of learning adversary models in CAPTURE, namely *parameter separation* and *target abstraction*. The first heuristic divides the set of model parameters into separate subsets and then iteratively learns these subsets of parameters separately while fixing the values of the other subsets. This heuristic decomposes the learning process into less complex learning components which help in speeding up the learning process with no loss in accuracy. The second heuristic of target abstraction works by leveraging the continuous spatial structure of the wildlife domain, starting the learning process with a coarse discretization of forest area and gradually using finer discretization instead of directly starting with the most detailed representation, leading to improved runtime overall.

My third contribution lies in computing the optimal patrolling strategy of the rangers given the new behavioral model. Specifically, I provide a new game-theoretic algorithm for single/multiple-step patrolling plans wherein the poachers' actions (which follow the LensQR-Poacher model) are recursively explored in multiple time steps. Finally, I extensively evaluate the prediction accuracy of my new LensQR-Poacher model based on a detailed analysis of the largest dataset of real-world defender-adversary interactions collected by rangers in Queen Elizabeth National Park (QENP) over 12 years. In fact, this is the largest such study in the security games literature. The experimental results show that my model is superior to existing models in predicting the

poachers' behavior, demonstrating the advances of my model over the previous state-of-the-art models. CAPTURE will be tested in Uganda in early 2016.

5.2 Poacher Behavioral Learning

Security agencies protecting wildlife have a great need for tools that analyze, model and predict behavior of poachers. Such modeling tools help the security agencies gain situational awareness, and decide general strategies; in addition, these agencies also find it useful to have patrol planning tools that are built based on such models. The key here is that in wildlife protection areas around the world, these security agencies have collected large amounts of data related to interactions betIen defenders (patrollers) and adversaries (poachers). In my work, I focus on QENP (Yang et al., 2014; Critchlow et al., 2015), where in collaboration with the Wildlife Conservation Society (WCS) and Uganda Wildlife Authority (UWA), I have obtained 12 years of ranger-collected data (that is managed in database MIST/SMART).

In CAPTURE, I introduce a new hierarchical behavioral model, LensQR-Poacher, to predict the poachers' behavior in the wildlife domain, taking into account the challenge of rangers' imperfect observation. Overall, the new model consists of two layers. One layer models the probability the poachers attack each target wherein the temporal effect on the poachers' behaviors is incorporated. The next layer predicts the conditional probability of the rangers detecting any poaching sign at a target given that the poachers attack that target. These two layers are then integrated to predict the rangers' final observations. In the LensQR-Poacher model, I incorporate the effect of the rangers' patrols on both layers, i.e., how the poachers adapt their behaviors according to rangers' patrols and how the rangers' patrols determine the rangers' detectability of poaching signs. Furthermore, I consider the poachers' past activity in reasoning about future actions of the poachers. I also include different domain features to predict either attacking probabilities or detection probabilities or both.

5.2.1 LensQR-Poacher: Hierarchical Behavioral Model

I denote by T the number of time steps, N the number of targets, and K the number of domain features. At each time step t, each target i is associated with a set of feature values $\mathbf{x_{t,i}} =$

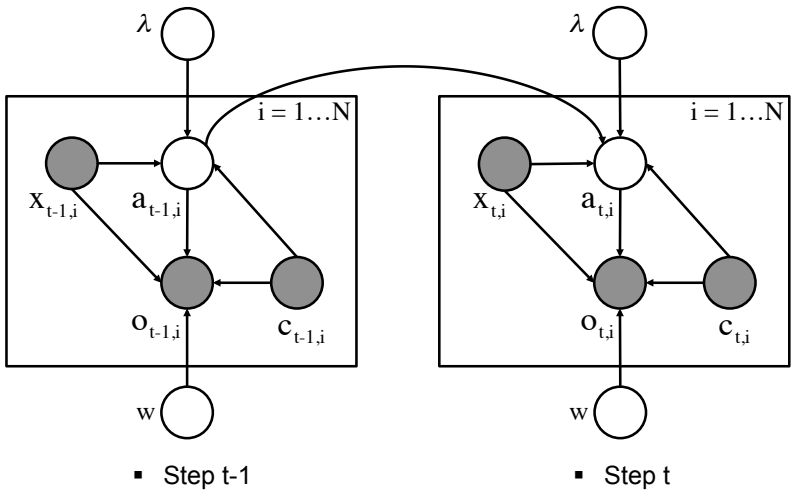


Figure 5.1: Dependencies among LensQR-Poacher modeling elements

$\{x^k_{t,i}\}$ where $k = 1 \dots K$ and $x^k_{t,i}$ is the value of the k^{th} feature at (t, i). In addition, $c_{t,i}$ is defined as the coverage probability of the rangers at (t, i). When the rangers patrol target i in time step t, they have observation $o_{t,i}$ which takes an integer value in $\{-1, 0, 1\}$. Specifically, $o_{t,i} = 1$ indicates that the rangers observe a poaching sign at (t, i), $o_{t,i} = 0$ means that the rangers have no observation and $o_{t,i} = -1$ when the rangers do not patrol at (t, i). Furthermore, I define $a_{t,i} \in \{0, 1\}$ as the actual action of poachers at (t, i) which is hidden from the rangers. Specifically, $a_{t,i} = 1$ indicates the poachers attack at (t, i); on the other hand, $a_{t,i} = 0$ means the poachers did not attack at (t, i). In this work, I only consider the situation of attacked or not (i.e., $a_{t,i} \in \{0, 1\}$); the case of multiple-level attacks is left for future work. Moreover, I mainly focus on the problem of false negative observations, meaning that there may still exist poaching activity at locations where the rangers found no sign of poaching. I make the reasonable assumption that there is no false positive observation, meaning that if the rangers found any poaching sign at a target, the poachers did attack that target. In other words, I have $p(a_{t,i} = 1|o_{t,i} = 1) = 1$ and $p(o_{t,i} = 1|a_{t,i} = 0) = 0$.

The graphical representation of the new LensQR-Poacher model is shown in Figure 5.1 wherein the directed edges indicate the dependence betIen elements of the model. The grey nodes refer to known elements for the rangers such as domain features, the rangers' coverages and observations while the white nodes represent the unknown elements such as the actual actions of poachers. The elements $(\lambda, \mathbf{w})$ are model parameters which I will explain later.

My new LensQR-Poacher graphical model is a significant advance over previous models from behavioral game theory, such as QR/LensQR, and similarly models from conservation biology (Hofer et al., 2000; Critchlow et al., 2015). First, unlike LensQR/QR which consider poachers behavior to be independent betIen different time steps, I assume that the poachers' actions $a_{t,i}$ depends on the poachers' activities in the past $a_{t-1,i}$ and the rangers' patrolling strategies $c_{t,i}$. This is because poachers may tend to come back to the areas they have attacked before. Second, LensQR-Poacher considers a much richer set of domain features $\{x^k_{t,i}\}$ that have not been considered earlier but are relevant to my domain, e.g., slope and habitat. Third, another advance of CAPTURE is modeling the observation uncertainty in this domain. I expect that the rangers' observations $o_{t,i}$ depend on the actual actions of the poachers $a_{t,i}$, the rangers' coverage probabilities $c_{t,i}$ and domain features $\{x^k_{t,i}\}$. Finally, I adopt the logistic model (Bishop, 2006) to predict the poachers' behaviors; one advantage of this model compared to LensQR/QR is that it does not assume a known number of attackers and models probability of attack at every target independently. Thus, given the actual action of poachers, $a_{t-1,i}$, at previous time step $(t-1,i)$, the rangers' coverage probability $c_{t,i}$ at (t,i), and the domain features $\mathbf{x_{t,i}} = \{x^k_{t,i}\}$, I aim at predicting the probability that poachers attack (t,i) as follows:

$$p(a_{t,i} = 1 | a_{t-1,i}, c_{t,i}, \mathbf{x_{t,i}}) = \frac{e^{\lambda'[a_{t-1,i}, c_{t,i}, \mathbf{x_{t,i}}, 1]}}{1 + e^{\lambda'[a_{t-1,i}, c_{t,i}, \mathbf{x_{t,i}}, 1]}} \tag{5.1}$$

where $\lambda = \{\lambda_k\}$ is the $(K+3) \times 1$ parameter vector which measure the importance of all factors towards the poachers' decisions. λ_{K+3} is the free parameter and λ' is the transpose vector of λ. In essence, compared to Equation 4.2 where LensQR was seen to only use three features, I now have a weighted sum over a much larger number of features as is appropriate in my wildlife domain.

Furthermore, if the poachers attack at (t,i), I predict the probability that the rangers can detect any poaching signs as follows:

$$p(o_{t,i} = 1 | a_{t,i} = 1, c_{t,i}, \mathbf{x_{t,i}}) = c_{t,i} \times \frac{e^{\mathbf{w}'[\mathbf{x_{t,i}}, 1]}}{1 + e^{\mathbf{w}'[\mathbf{x_{t,i}}, 1]}} \tag{5.2}$$

where the first term is the probability that the rangers are present at (t,i) and the second term indicates the probability that the rangers can detect any poaching sign when patrolling at (t,i). Additionally, $\mathbf{w} = \{w_k\}$ is the $(K+1) \times 1$ vector of parameters which indicates the significance of domain features in affecting the rangers' probability of detecting poaching signs. $\mathbf{w}'$ is

transpose of $\mathbf{w}$. In QENP specifically, LensQR-Poacher employs seven features: animal density, distances to rivers/roads/villages, net primary productivity (NPP), habitat and slope to predict attacking/detection probabilities.

In the following, I will explain my approach for learning the parameters $(\lambda, \mathbf{w})$ of my hierarchical model. I use $p(a_{t,i} = 1|a_{t-1,i}, c_{t,i})$ and $p(o_{t,i} = 1|a_{t,i} = 1, c_{t,i})$ as the abbreviations of the LHSs in Equations 5.1 and 5.2. The domain features $\mathbf{x_{t,i}}$ are omitted in all equations for simplification.

5.2.2 Parameter Estimation

Due to the presence of unobserved variables $\mathbf{a} = \{a_{t,i}\}$, I use the standard Expectation Maximization (EM) method in order to estimate $(\lambda, \mathbf{w})$. In particular, EM attempts to maximize the log-likelihood that the rangers can have observations $\mathbf{o} = \{o_{t,i}\}$ given the rangers' coverage probabilities $\mathbf{c} = \{c_{t,i}\}$ and domain features $\mathbf{x} = \{\mathbf{x_{t,i}}\}$ for all time steps $t = 1, \ldots, T$ and targets $i = 1, \ldots, N$ which is formulated as follows:

$$\max_{\lambda, \mathbf{w}} \log p(\mathbf{o}|\mathbf{c}, \mathbf{x}, \lambda, \mathbf{w}) \tag{5.3}$$

The standard EM procedure (Bishop, 2006) is to start with an initial estimate of $(\lambda, \mathbf{w})$ and iteratively update the parameter values until a locally optimal solution of (5.3) is reached. Many restarts are used with differing initial values of $(\lambda, \mathbf{w})$ to find the global optimum. Each iteration of EM consists of two key steps:

- **E** step: compute $p(\mathbf{a}|\mathbf{o}, \mathbf{c}, (\lambda, \mathbf{w})^{\mathbf{old}})$
- **M** step: update values of the parameters: $(\lambda, \mathbf{w})^{\mathbf{old}}) = (\lambda^*, \mathbf{w}^*)$ where $(\lambda^*, \mathbf{w}^*) = \underset{\lambda, \mathbf{w}}{\operatorname{argmax}} \sum_{\mathbf{a}} p(\mathbf{a}|\mathbf{o}, \mathbf{c}, (\lambda, \mathbf{w})^{\mathbf{old}}) \log(p(\mathbf{o}, \mathbf{a}|\mathbf{c}, \lambda, \mathbf{w}))$.

In my case, the **E** (Expectation) step attempts to compute the probability that the poachers take actions $\mathbf{a} = \{a_{t,i}\}$ given the rangers' observations $\mathbf{o}$, the rangers' patrols $\mathbf{c}$, the domain features $\mathbf{x} = \{\mathbf{x_{t,i}}\}$, and current values of the model parameters $(\lambda, \mathbf{w})^{\mathbf{old}}$. The **M** (Maximization) step tries to maximize the expectation of the logarithm of the complete-data $(\mathbf{o}, \mathbf{a})$ likelihood function given the action probabilities computed in the **E** step and updates the value of $(\lambda, \mathbf{w})^{\mathbf{old}}$ with the obtained maximizer.

Although I can decompose the log-likelihood, the EM algorithm is still time-consuming due to the large number of targets and parameters. Therefore, I use two novel ideas to speed up the algorithm: *parameter separation* for accelerating the convergence of EM and *target abstraction* for reducing the number of targets.

5.2.2.1 Parameter Separation

Observe that the objective in the **M** step can be split into two additive parts as follows:

$$\begin{aligned}
&\sum_{\mathbf{a}} p(\mathbf{a}|\mathbf{o},\mathbf{c},(\lambda,\mathbf{w})^{\mathbf{old}})\log(p(\mathbf{o},\mathbf{a}|\mathbf{c},\lambda,\mathbf{w})) \qquad (5.4)\\
&\quad=\sum_{t,i}\sum_{a_{t,i}} p(a_{t,i}|\mathbf{o},\mathbf{c},(\lambda,\mathbf{w})^{\mathbf{old}})\log p(o_{t,i}|a_{t,i},c_{t,i},\mathbf{w})\\
&\quad+\sum_{t,i}\sum_{a_{t,i}}\sum_{a_{t-1,i}} p(a_{t,i},a_{t-1,i}|\mathbf{o},\mathbf{c},(\lambda,\mathbf{w})^{\mathbf{old}})\log p(a_{t,i}|a_{t-1,i},c_{t,i},\lambda)
\end{aligned}$$

In (5.4), the first component is obtained as a result of decomposing w.r.t the detection probabilities of the rangers at every (t,i) (Equation 5.2). The second one results from decomposing according to the attacking probabilities at every (t,i) (Equation 5.1). Importantly, the first component is only a function of $\mathbf{w}$ and the second component is only a function of λ. Following this split, for my problem, the **E** step reduces to computing the following two quantities:

$$\textbf{Total probability: } p(a_{t,i}|\mathbf{o},\mathbf{c},(\lambda,\mathbf{w})^{\mathbf{old}}) \qquad (5.5)$$

$$\textbf{2-step probability: } p(a_{t,i},a_{t-1,i}|\mathbf{o},\mathbf{c},(\lambda,\mathbf{w})^{\mathbf{old}}) \qquad (5.6)$$

which can be computed by adapting the Baum-Welch algorithm (Bishop, 2006) to account for missing observations, i.e., $o_{t,i}=-1$ when rangers do not patrol at (t,i). This can be done by introducing $p(o_{t,i}=-1|a_{t,i},c_{t,i}=0)=1$ when computing (5.5) and (5.6).

More importantly, as shown in (5.4), the structure of my problem allows for the decomposition of the objective function into two separate functions w.r.t attack parameters λ and detection parameters $\mathbf{w}$: $F^d(\mathbf{w})+F^a(\lambda)$ where the detection function $F^d(\mathbf{w})$ is the first term of the RHS in Equation 5.4 and the attack function $F^a(\lambda)$ is the second term. Therefore, instead of maximizing $F^d(\mathbf{w})+F^a(\lambda)$, I decompose each iteration of EM into two **E** steps and two **M** steps that enables maximizing F^d and F^a separately as follows:

- **E1** step: compute total probability

- **M1** step: $\mathbf{w}^* = \text{argmax}_{\mathbf{w}} F^d(\mathbf{w})$; update $\mathbf{w^{old}} = \mathbf{w}^*$
- **E2** step: compute 2-step probability
- **M2** step: $\lambda^* = \text{argmax}_{\lambda} F^a(\lambda)$; update $\lambda^{\mathbf{old}} = \lambda^*$

Note that the detection and attack components are simpler functions compared to the original objective since these components only depend on the detection and attack parameters respectively. Furthermore, at each EM iteration, the parameters get closer to the optimal solution due to the decomposition since the attack parameter is now updated based on the new detection parameters from the **E1/M1** steps instead of the old detection parameters from the previous iteration. Thus, by decomposing each iteration of EM according to attack and detection parameters, EM will converge more quickly without loss of solution quality. The convergence and solution quality of the separation can be analyzed similarly to the analysis of multi-cycle expected conditional maximization (Meng & Rubin, 1993).

Furthermore, the attack function $F^a(\lambda)$ is shown to be concave by Proposition 1, allowing us to easily obtain the global optimal solution of the attacking parameters λ at each iteration of EM.

Proposition 1. *$F^a(\lambda)$ is concave in the attack parameters λ.*

Proof. As shown in (5.4), the attack function (the second term on the RHS of (5.4)) is the expectation of the logarithm of the attacking probability, $\log p(a_{t,i}|a_{t-1,i}, \lambda)$, at (t, i). This logarithm function has the following formulations according to Equation 5.1:

$$\begin{aligned}
\log p(a_{t,i}{=}1|a_{t-1,i}, c_{t,i}, \lambda) &= \lambda'[a_{t-1,i}, c_{t,i}, \mathbf{x_{t,i}}, 1] - \log(1{+}e^{\lambda'[a_{t-1,i}, c_{t,i}, \mathbf{x_{t,i}}, 1]}) \\
\log p(a_{t,i}{=}0|a_{t-1,i}, c_{t,i}, \lambda) &= -\log(1{+}e^{\lambda'[a_{t-1,i}, c_{t,i}, \mathbf{x_{t,i}}, 1]}) \qquad (5.7)
\end{aligned}$$

which are a concave functions in λ (its Hessian matrix is semi-negative definite). Since a linear combination (with positive weights) of concave functions is also a concave function, the attack function, $F^a(\lambda)$, is concave in the attack parameters λ. □

5.2.2.2 Target Abstraction

My second idea is to reduce the number of targets via target abstraction. Previous work in network security and poker games has also applied abstraction for reducing the complexity of solving

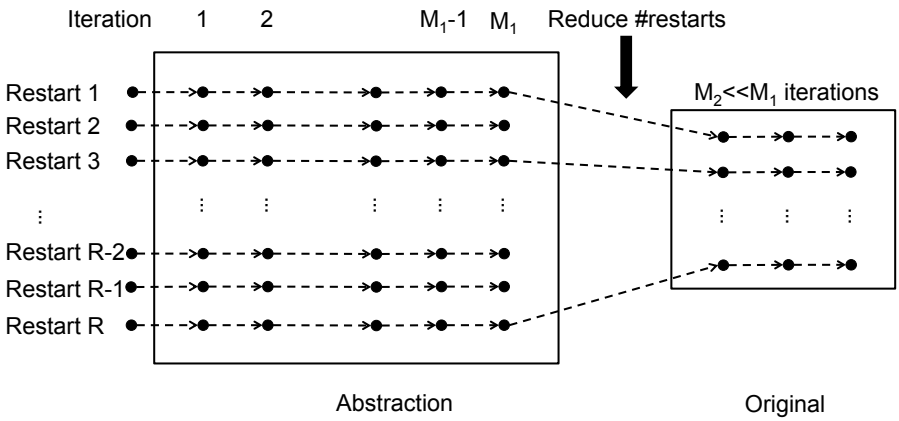


Figure 5.2: Target Abstraction

these games by exploring intrinsic properties of the games (Basilico & Gatti, 2011; Sandholm & Singh, 2012). In CAPTURE, by exploiting the spatial connectivity betIen grid cells of the conservation area, I can divide the area into a smaller number of grid cells by merging each cell in the original grid with its neighbors into a single bigger cell. The corresponding domain features are aggregated accordingly. Intuitively, neighboring cells tend to have similar domain features. Therefore, I expect that the parameters learned in both the original and abstracted grid would expose similar characteristics. Hence, the model parameters estimated based on the abstracted grid could be effectively used to derive the parameter values in the original one.

In this work, I leverage the values of parameters learned in the abstracted grid in two ways: (i) reduce the number of restarting points (i.e., initial values of parameters) for reaching different local optimal solutions in EM; and (ii) reduce the number of iterations in each round of EM. The idea of target abstraction is outlined in Figure 5.2 wherein each black dot corresponds to a set of parameter values at a particular iteration given a specific restarting points. At the first stage, I estimate the parameter values in the abstracted grid given a large number of restarting points R, assuming that I can run M_1 EM iterations. At the end of the first stage, I obtain R different sets of parameter values; each corresponds to a local optimal solution of EM in the abstracted grid. Then at the second stage, these sets of parameter values are used to estimate the model parameters in the original grid as the following: (i) only a subset of K resulting parameter sets which refer to the top local optimal solutions in the abstracted grid are selected as initial values of parameters in the original grid; and (ii) instead of running M_1 EM iterations again, I only proceed with $M_2 << M_1$ iterations in EM since I expect that these selected parameter values are already

Ill learned in the abstracted grid and thus could be considered as *warm restarts* in the original grid.

5.3 Patrol Planning

Once the model parameters $(\lambda, \mathbf{w})$ are learned, I can compute the optimal patrolling strategies for the rangers in next time steps taking into account the LensQR-Poacher model. I consider two circumstances: 1) single-step patrol planning in which the rangers only focus on generating the patrolling strategy at the next time step and 2) multiple-step patrol planning for generating strategies for the next $\Delta T > 1$ time steps, given the rangers' patrol and observation history and domain features. While the former provides a one-step patrolling strategy with an immediate but short-term benefit, the latter generates strategies across multiple time steps with a long-term benefit. I leave the choice of which planning option to use for the rangers given the cost/benefit trade-off betIen the two. The key challenge in designing strategies for the rangers given the LensQR-Poacher model is that I need to take into account new aspects of the modeling of the adversary. These include the rangers' detection uncertainty and the temporal dependency of the poachers' activities. This challenge leads to a complicated non-convex optimization problem to compute the optimal patrolling strategy for the rangers; I provide novel game-theoretic algorithms to solve it.

I suppose that the rangers have an observation history $\mathbf{o} = \{o_{t',i}\}$ for $t' = 1, \ldots, T$ and $i = 1, \ldots, N$. Similar to standard SSGs, I assume that if the poachers successfully attack at (t, i), the rangers receive a penalty $P^d_{t,i}$. Conversely, if the rangers successfully confiscate poaching tools at (t, i), the rangers obtain a reward $R^d_{t,i}$. Therefore, the rangers' expected utility at (t, i) if the poachers attack at (t, i) is computed as follows where $p(o_{t,i} = 1 | a_{t,i} = 1, c_{t,i})$ is the rangers' detection probability at (t, i) as shown in Equation 5.2:

$$U^d_{t,i} = p(o_{t,i} = 1 | a_{t,i} = 1, c_{t,i}) \times [R^d_{t,i} - P^d_{t,i}] + P^d_{t,i} \tag{5.8}$$

I now explain in detail my new game-theoretic algorithms. The rangers' past patrols at (t', i) for for $t' = 1, \ldots, T$ and $i = 1, \ldots, N$ are already known and thus can be omitted in all following mathematical formulations for simplification.

5.3.1 Single-step Patrol Planning

Given the rangers' observation history $\mathbf{o}$ and the model parameters $(\lambda, \mathbf{w})$, the problem of computing the optimal strategies at the next time step $T+1$ can be formulated as follows:

$$\max_{\{c_{T+1,i}\}} \sum\nolimits_i p(a_{T+1,i}=1|\mathbf{o}, c_{T+1,i}) \times U^d_{T+1,i} \tag{5.9}$$

$$s.t.\ 0 \le c_{T+1,i} \le 1,\ i = 1 \dots N \tag{5.10}$$

$$\sum\nolimits_i c_{T+1,i} \le B \tag{5.11}$$

where B is the maximum number of ranger resources and $p(a_{T+1,i}=1|\mathbf{o}, c_{T+1,i})$ is the probability that the poachers attack at $(T+1, i)$ given the rangers' observation history $\mathbf{o}$ and the rangers' coverage probability $c_{T+1,i}$. Since the poachers' behaviors depends on their activities in the past (which is hidden to the rangers), I need to examine all possible actions of the poachers in previous time steps in order to predict the poachers' attacking probability at $(T+1, i)$. Hence, the attacking probability $p(a_{T+1,i}=1|\mathbf{o}, c_{T+1,i})$ should be computed by marginalizing over all possible actions of poachers at (T, i) as follows:

$$p(a_{T+1,i}=1|c_{T+1,i}, \mathbf{o}) = \sum\nolimits_{a_{T,i}} p(a_{T+1,i}=1|a_{T,i}, c_{T+1,i}) \times p(a_{T,i}|\mathbf{o}) \tag{5.12}$$

where $p(a_{T+1,i}|a_{T,i}, c_{T+1,i})$, which is computed in (5.1), is the attacking probability at $(T+1, i)$ given the poachers' action $a_{T,i}$ at (T, i) and the rangers' coverage probability $c_{T+1,i}$. In addition, $p(a_{T,i}|\mathbf{o})$ is the total probability at (T, i) which can be recursively computed based on the Baum-Welch approach as discussed in Section 5.2. Overall, (5.9 – 5.11) is a non-convex optimization problem in the rangers' coverage probabilities $\{c_{T+1,i}\}$. Fortunately, each additive term of the rangers' utility in (5.9) is a separate sub-utility function of the rangers' coverage, $c_{T+1,i}$, at $(T+1, i)$:

$$f_i(c_{T+1,i}) = p(a_{T+1,i}=1|\mathbf{o}, c_{T+1,i}) \times U^d_{T+1,i} \tag{5.13}$$

Therefore, I can piecewise linearly approximate $f_i(c_{T+1,i})$ and represent (5.9 – 5.11) as a Mixed Integer Program which can be solved by CPLEX. The details of piecewise linear approximation can be found at (Yang, Ordonez, & Tambe, 2012). Essentially, the piecewise linear approximation method provides an $O(\frac{1}{K})$-optimal solution for (5.9 – 5.11) where K is the number of piecewise segments (Yang et al., 2012).

5.3.2 Multi-step Patrol Planning

In designing multi-step patrol strategies for the rangers, there are two key challenges in incorporating the LensQR-Poacher model that I need to take into account: 1) the time dependence of the poachers' behavior; and 2) the actual actions of the poachers are hidden (unobserved) from the rangers. These two challenges make the problem of planning multi-step patrols difficult as I show below.

Given that the rangers have an observation history $\mathbf{o} = \{o_{t'i}\}$ for $t' = 1, \ldots, T$ and $i = 1 \ldots N$, the rangers aim at generating patrolling strategies $\{c_{t,i}\}$ in next ΔT time steps where $t = T+1, \ldots, T+\Delta T$. Then the problem of computing the optimal patrolling strategies for next ΔT time step $T+1, \ldots, T+\Delta T$ can be formulated as follows:

$$\max_{\{c_{t,i}\}} \sum\nolimits_{t,i} p(a_{t,i} = 1|\mathbf{o}, c_{T+1\ldots t,i}) U^d_{t,i} \tag{5.14}$$

$$s.t.\ 0 \leq c_{t,i} \leq 1,\ t = T+1 \ldots T+\Delta T,\ i = 1 \ldots N \tag{5.15}$$

$$\sum\nolimits_{i} c_{t,i} \leq B,\ t = T+1 \ldots T+\Delta T. \tag{5.16}$$

where $p(a_{t,i} = 1|\mathbf{o}, c_{T+1\ldots t,i})$ is the attacking probability at (t, i) given the rangers' coverages at (t', i) where $t' = T+1, \ldots, t$ and observation history $\mathbf{o} = \{o_{t',i}\}$ where $t' = 1, \ldots, T$. Because of the two aforementioned challenges, I need to examine all possible actions of the poachers in previous time steps in order to compute the attacking probability at (t, i), $p(a_{t,i} = 1|\mathbf{o}, c_{T+1\ldots t,i})$. my idea is to recursively compute this attacking probability via the attacking probabilities at previous time steps as follows:

$$p(a_{t,i} = 1|\mathbf{o}, c_{T+1\ldots t,i}) = \sum_{a_{t-1,i}} p(a_{t,i}|a_{t-1,i}, c_{t,i}) \times p(a_{t-1,i}|\mathbf{o}, c_{T+1\ldots t-1,i}) \tag{5.17}$$

where the initial step is to compute the total probability $p(a_{T,i}|\mathbf{o})$ by using the Baum-Welch approach. Here, the objective in (5.14) can be no longer divided into separate sub-utility functions of a single coverage probability at a particular (t, i) because of the time dependency of the poachers' behaviors. Thus, I can not apply piecewise linear approximation as in the single-step patrol planning for solving (5.14 – 5.16) quickly. In this work, I use non-convex solvers (i.e., fmincon in MATLAB) to solve (5.14 – 5.16).

In (Fang et al., 2015), the dependence of the attacker's actions on the defender's patrolling strategies in the past is also considered; they assume that the attacker's responses follow the

LensQR model while the attacker perceives the defender's current strategy as a weighted linear function of the defender's strategies in the past. They also assume that these weights are known, thereby making the computational problem easy. In contrast, I make the more realistic assumption that the poachers are influenced by their own past observations and my learning algorithm learns the weights corresponding to such influence from the data. Unfortunately, this makes the problem of planning multistep patrols more difficult as shown before.

5.4 Experimental Results

I aim to (i) extensively assess the prediction accuracy of the LensQR-Poacher model compared to existing models based on real-world wildlife/poaching data; (ii) examine the runtime performance of learning the new model; and (iii) evaluate the solution quality of the CAPTURE planning for generating patrols. In the following, I provide a brief description of the real-world wildlife data used.

5.4.1 Real-world Wildlife/Poaching Data

To learn the poachers' behavior, I use the wildlife data collected by the rangers over 12 years from 2003 to 2014 in QENP (Figure 5.3 with animal density). This work is accomplished in collaboration with the Wildlife Conservation Society (WCS) and Uganda Wildlife Authority (UWA). While patrolling, the park rangers record information such as locations (latitude/longitude), times, and observations (e.g., signs of human illegal activities). Figure 5.4 shows an example of data samples w.r.t the rangers' patrols, including patrol IDs, patrol days (i.e., which day of the patrols—each patrol may last several days), waypoints, dates and times, observations, observation codes, and locations in terms of (longitude, latitude)

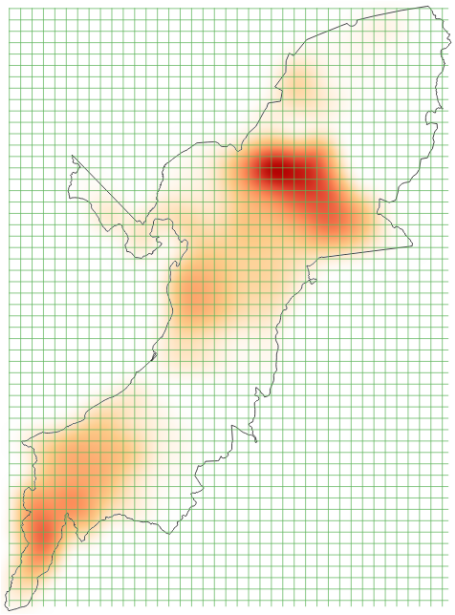


Figure 5.3: QENP with animal density

using the World Geodetic System (WGS84). Similar to the work of (Critchlow et al., 2015), I also divide collected human signs into six different groups: commercial animal (i.e., human signs such as snares which refer to poaching commercial animals such as buffalo, hippo and elephant), non-commercial animal, fishing, encroachment, commercial plant, and non-commercial plant. In this work, I mainly focus on two types of human illegal activities: commercial animal and non-commercial animal which are major threats to key species of concern such as elephants and hippos.

Patrol ID	Patrol Day	Waypoint	Date Time	Observation	Observation Code	Latitude	Longitude
5691	1	1	2/4/12 8:17	Poaching sign	Snares	-0.141765	29.831301
5691	1	2	2/4/12 8:43	Poaching	Hunting	-0.139047	29.824643
5691	1	3	2/4/12 9:09	Poaching	Hunting	-0.148023	29.81837
5691	1	4	2/4/12 9:35	Elephant	Sighting	-0.149393	29.819606

Figure 5.4: Data samples of rangers' patrols

ID	Longitude	Latitude	Habitat	NPP	Slope	Road Distance	Town Distance	River Distance	Animal Density
1	1.25e+05	9.92e+06	0.2	1.1744	0.0105	61.16	6513.4	594.49	2.8288
2	1.25e+05	9.921e+06	0.2	1.2587	0.0064	625.55	7035.6	439.24	4.3833
3	1.25e+05	9.922e+06	0.2	1.3040	0.0062	1213.10	7653.4	197.55	5.9379
4	1.26e+05	9.92e+06	0.2	1.1227	0.0196	761.61	5648.1	1373.50	3.8863

Figure 5.5: Data samples of domain features

The poaching data is then divided into the four different groups according to four seasons in Uganda: dry season I (Jun, July, and August), dry season II (December, January, and February), rainy season I (March, April, and May), and rainy season II (September, October, November). I aim at learning behaviors of the poachers w.r.t these four seasons as motivated by the fact that the poachers' activities usually vary seasonally. In the end, I obtain eight different categories of wildlife data given that I have the two poaching types and four seasons. Furthermore, I use seven domain features in learning the poachers' behavior, including animal density, slope, habitat, net primary productivity (NPP), and locations of villages/rivers/roads provided by (Critchlow et al., 2015).

I divide the park area into a $1km \times 1km$ grid consisting of more than 2500 grid cells ($\approx 2500km^2$). Domain features and the rangers' patrols and observations are then aggregated into the grid cells. Figure 5.5 shows an example of data samples w.r.t domain features at four grid cells together with these cells' id and locations (i.e., the longitude and latitude (UTM grid zone 36S) of the cells' centers) after aggregation. I also refine the poaching data by removing all abnormal data points such as the data points which indicate that the rangers conducted patrols outside the QENP park or the rangers moved too fast, etc. Since I attempt to predict the poachers' actions in the future based on their activities in the past, I apply a time window (i.e., five years) with an 1-year shift to split the poaching data into eight different pairs of training/test sets. For example, for the (commercial animal, rainy season I) category, the oldest training/test sets correspond to four-year data (2003–2006) w.r.t this category for training and one-year (2007) data for testing. In addition, the latest training/test sets refer to the four years (2010–2013) and one year (2014) of data respectively. In total, there are eight different training/test sets for each of my eight data categories.

5.4.2 Behavioral Learning

Prediction Accuracy. In this work, I compare the prediction accuracy of seven models: 1) LensQR-Poacher (LensQR-Poacher with parameter separation); 2) P-Abstract (LensQR-Poacher with parameter separation and target abstraction); 3) P-NoTime (LensQR-Poacher with parameter separation and without the component of temporal effect); 4) Logit (Logistic Regression); 5) LensQR; 6) SVM (Support Vector Machine); and 7) Non-Lipschitz (Parametric Lipschitz (Sinha, Kar, & Tambe, 2016)). I use AUC (Area Under the Curve) to measure the prediction accuracy of these behavioral models. Based on ROC plots of data, AUC is a standard and common statistic in machine learning for model evaluation (Bradley, 1997). Essentially, AUC refers to the probability that a model will weight a random positive poaching sample higher than a random negative poaching sample in labeling these samples as positive (so, higher AUC values are better). For each data category (w.r.t poaching types and poaching seasons), the AUC values of all the models are averaged over the eight test sets as explained in Section 5.4.1. I also show the average prediction accuracy over all seasons. I use bootstrap-t (Wilcox, 2002) to measure the statistical significance of my results.

Models	Rainy I	Rainy II	Dry I	Dry II	Average
LensQR-Poacher	0.76	0.76	0.74	0.73	0.7475
P-Abstract	0.79	0.76	0.74	0.67	0.74
P-NoTime	0.71	0.75	0.67	0.71	0.71
Logit	0.53	0.59	0.57	0.60	0.5725
LensQR	0.53	0.59	0.56	0.62	0.575
SVM	0.61	0.59	0.51	0.66	0.5925
Lipschitz	0.64	0.58	0.55	0.42	0.5457

Table 5.1: AUC: Commercial Animal

The results are shown in Tables 5.1 and 5.2. I can infer the following key points from these tables. First, and most important, LensQR-Poacher improves performance over the state of the art, which is LensQR and SVM. LensQR-Poacher's average AUC in Table 5.1 (essentially this is over 32 data points of eight test sets over four seasons) is 0.7475 vs 0.575 for LensQR, and in Table 5.2 is 0.74 vs 0.57 for LensQR. This clearly shows a statistically significant ($\alpha = 0.05$) advance in my modeling accuracy. This improvement illustrates that all the four advances in LensQR-Poacher mentioned in Section 5.1 — addressing observation error, time dependence, detailed domain features and not requiring a firm count of poachers beforehand – have indeed led to a significant advance in LensQR-Poacher's performance. I can now attempt to understand the contributions of each of LensQR-Poacher's improvements, leading to the next few insights. Second, comparison of LensQR-Poacher with P-NoTime which only addresses the challenge of observation bias demonstrates the importance of considering time dependence. Third, while parameter separation does not cause any loss in solution quality as discussed in Section 5.2.2, Tables 5.1 and 5.2 shows that the prediction accuracy of LensQR-Poacher with target abstraction is good in general except for Dry season II with Commercial Animal. As I show later, parameter separation and target abstraction help in speeding up the runtime performance of learning the LensQR-Poacher model.

Fourth, the results of the model parameter values in the LensQR-Poacher model show that all these domain features substantially impact the poachers' behaviors. For example, one learning result on the model parameters corresponding to the category (non-commercial animal/dry season I) in 2011 is $(0.33, 1.46, -2.96, -1.97, 1.88, -0.78, 0.36)$ for domain features (habitat, NPP, slope, road distance, town distance, water distance, and animal density), -1.40 for the rangers' coverage probability and 4.27 for the poachers' past action. Based on these learned weights, I

Models	Rainy I	Rainy II	Dry I	Dry II	Average
LensQR-Poacher	0.76	0.70	0.78	0.72	0.74
P-Abstract	0.76	0.70	0.74	0.70	0.725
P-NoTime	0.72	0.68	0.75	0.70	0.7125
Logit	0.52	0.63	0.57	0.52	0.56
LensQR	0.54	0.62	0.58	0.54	0.57
SVM	0.42	0.50	0.55	0.56	0.5075
Lipschitz	0.60	0.47	0.55	0.43	0.5125

Table 5.2: AUC: Non-Commercial Animal

Year	2009	2010	2011	2012	2013	2014
weight	-10.69	-4.35	-0.7	-2.21	-1.78	-17.39

Table 5.3: Patrol weights in recent years

can interpret how these domain features affect the poachers' behavior. Specifically, the negative weights for road/water distances indicates that the poachers tend to poach at locations near roads/water. In addition, the resulting positive weight for the poachers' past actions indicates that the poachers are more likely to attack the targets which Ire attacked before. Furthermore, the resulting negative weight for the rangers' patrols also shows that the poachers' activity is influenced by the rangers' patrols, i.e., the poachers are less likely to attack targets with higher coverage probability of the rangers. Lastly, the ranger-poacher interaction changes over time as indicated by different negative weights of the rangers' patrols across different years (Table 5.3). For example, the patrol weight corresponding the category (non-commercial animal/dry season II) in 2014 is -17.39 while in 2013 is -1.78, showing that rangers' patrols have more impact on the poachers' behavior in 2014 than in 2013. This is the first time there is a real-world evidence which shows the impact of ranger patrols on poacher behavior.

Runtime Performance. I compare the runtime performance of learning the LensQR-Poacher model in three cases: 1) learning without both heuristics of parameter separation and target abstraction; 2) learning with parameter separation only; and 3) learning with both heuristics. In my experiments, for the first two cases, I run 20 restarting points and 50 iterations in EM. In the third case, I first run 20 restarting points and 40 iterations in EM with target abstraction. In particular, in target abstraction, I aggregate or interpolate all domain features as Ill as the rangers' patrols into $4km \times 4km$ grid cells while the original grid cell size is $1km \times 1km$. Then given the results

in the abstracted grid, I only select 5 results of parameter values (which correspond to the top five prediction accuracy results w.r.t the training set). I use these results as restarting points for EM in the original grid and only run 10 iterations to obtain the final learning results in the original grid.

Heuristics	Average Runtime
None	1419.16 mins
Parameter Separation	333.31 mins
Parameter Separation w/ Target Abstraction	222.02 mins

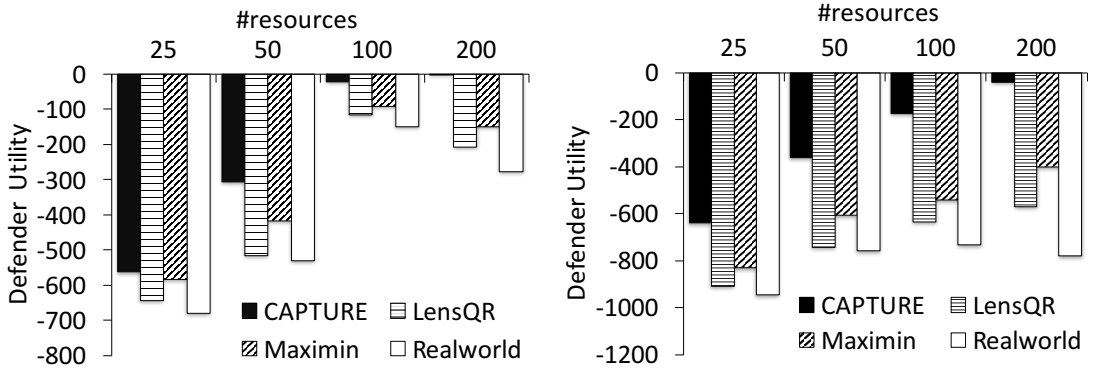
Table 5.4: LensQR-Poacher Learning: Runtime Performance

The results are shown in Table 5.4 which are averaged over 64 training sets (statistically significant ($\alpha = 0.05$)). In Table 5.4, learning LensQR-Poacher model parameters with parameter separation is significantly faster (i.e., 4.25 times faster) than learning LensQR-Poacher without this heuristic. This result clearly shows that reducing the complexity of the learning process (by decomposing it into simpler sub-learning components via parameter separation) significantly speeds up the learning process of LensQR-Poacher. Furthermore, the heuristic of target abstraction helps LensQR-Poacher in learning even faster although the result is not as substantial as with parameter separation, demonstrating the advantage of using this heuristic.

5.4.3 Patrol Planning

Based on the LensQR-Poacher model, I apply my CAPTURE planning algorithm (Section 5.3) to compute the optimal patrolling strategies for the rangers. The solution quality of my algorithm is evaluated based on the real-world QENP domain in comparison with LensQR (i.e., optimal strategies of the rangers against LensQR-based poachers), Maximin (maximin strategies of the rangers against worst-case poacher responses), and Real-world patrolling strategies of the rangers. The real-world strategies are derived from the four seasons in years 2007 to 2014. Given that CAPTURE's prediction accuracy is the highest among all the models, in my experiments, I assume that the poachers' responses follow my model. Given the QENP experimental settings, the reward of the rangers at each target are set to be zero while the penalty is the opposite of the animal density (i.e., zero-sum games). I assess the solution quality of all algorithms according to different number of the rangers' resources (i.e., number of targets the rangers can cover during a patrol). The

real-world patrolling strategies are normalized accordingly. Moreover, I also consider different number of time steps for generating patrols.



(a) 2 time steps (b) 4 time steps

Figure 5.6: Solution quality of CAPTURE-based planning

The experimental results are shown in Figure 5.6 which are averaged over all years and seasons. In Figure 5.6, the x-axis is the number of the rangers' resources and the y-axis is the aggregated utility the rangers receive over two and four time steps (seasons) for playing CAPTURE, LensQR, Maximin, and Real-world patrolling strategies respectively. As shown in Figure 5.6, my CAPTURE planning algorithm provides the highest utility for the rangers (with statistical significance ($\alpha = 0.05$)). Especially when the number of the rangers' resources increases, the CAPTURE planning algorithm significantly improves the quality of the rangers' patrolling strategies. Furthermore, my CAPTURE algorithm provides patrolling strategies which take into account the temporal effect on the poachers' behaviors. As a result, when the number of time steps increases (Figure 5.6(b)), my algorithm enhances its solution quality compared to the others.

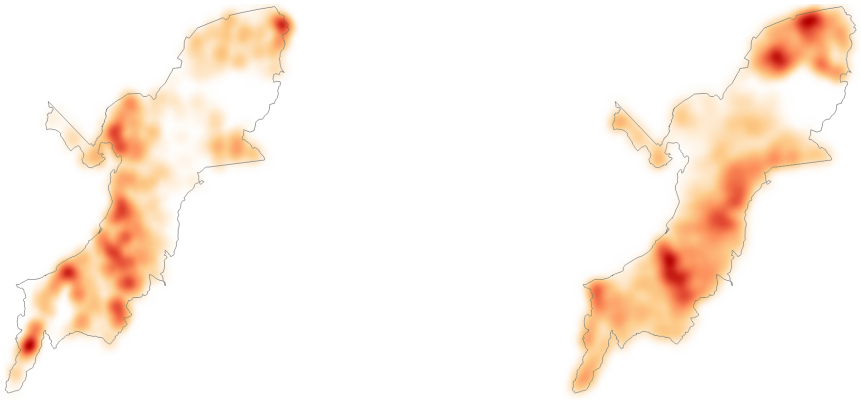
5.5 CAPTURE-based Application

CAPTURE tool is available for the rangers to predict the poachers' behavior and design optimal patrol schedules. Not all the regions are equally attractive to the poachers, so it is beneficial to detect the hotspots and favorite regions for poachers and protect those areas with higher probability. The general work-flow for this software could be itemized as: 1) Aggregating previously gathered data from the park to create a database that includes domain features, poaching signs and rangers'

effort to protect the area; 2) Pre-processing of the data points; 3) Running the CAPTURE tool to predict the attacking probability, rangers' observation over the area and generate the optimal patrol strategy; and 4) Post-processing of the results and generating the related heatmaps.

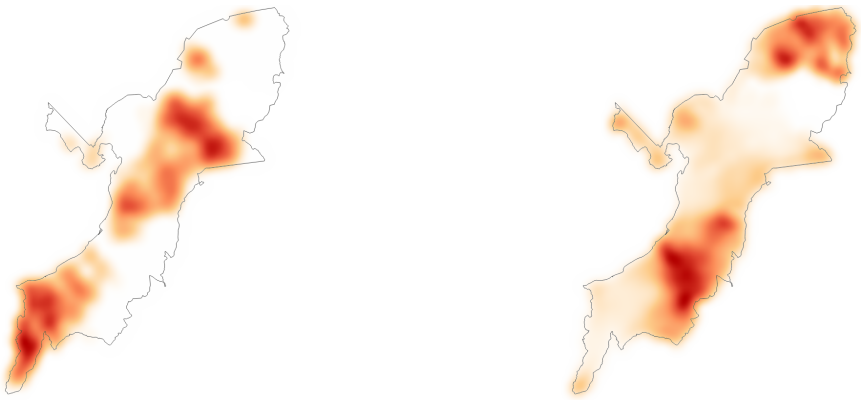
To compare the optimal strategy generated by the single-step patrol planning algorithm provided by CAPTURE and current real strategy deploying over the area, I plotted the related heatmaps according to the defender coverage, shown in Figure 5.7(a) and Figure 5.8(a). The darker the area, the greater chance to be covered by the rangers. Also, I used LenQR-Poacher to predict the probability of the attack based on these patrol strategies. These heatmaps are shown in Figure 5.7(b) and Figure 5.8(b). The darker regions on the map demonstrate the more attractive regions to the poachers.

I can see the following key points based on the heatmaps: (i) The optimal patrol strategy covers more of the regions with higher animal density (for instance south-west and middle parts of the park as shown in Figure 5.3). So the deployment of the optimal strategy would result in more protection to areas with higher animal density, as shown in Figure 5.8(a) and 5.8(b). (ii) The poaching heatmap shows significantly higher predicted activity of attackers against human generated patrols in regions with higher animal density, as shown in Figure 5.7(a) and 5.7(b).



(a) Patrol strategy (b) Attack probability

Figure 5.7: Heatmaps by CAPTURE (based on the real patrol strategy)



(a) Patrol strategy (b) Attack probability

Figure 5.8: Heatmaps by CAPTURE (based on the optimal strategy)

5.6 Summary

I propose a new predictive anti-poaching tool, CAPTURE. Essentially, CAPTURE introduces a novel hierarchical model, LensQR-Poacher, to predict the poachers' behaviors. The LensQR-Poacher model provides a significant advance over the state-of-the-art in modeling poachers in security games (Fang et al., 2016) and in conservation biology (Hofer et al., 2000; Critchlow et al., 2015) via 1) addressing the challenge of imperfect observations of the rangers; 2) incorporating the temporal effect on the poachers' behaviors; and 3) not requiring a known number of attackers. I provide two new heuristics: parameter separation and target abstraction to reduce the computational complexity in learning the model parameters. Furthermore, CAPTURE incorporates a new planning algorithm to generate optimal patrolling strategies for the rangers, taking into account the new complex poacher model. Finally, this application presents an evaluation of the largest sample of real-world data in the security games literature, i.e., over 12-years of data of attacker defender interactions in QENP. The experimental results demonstrate the superiority of my model compared to other existing models. CAPTURE will be tested in QENP in early 2016.

Chapter 6

Maximin-based Solutions for Security Games

This chapter will cover another major contribution of my work, which focuses on providing new robust maximin-based solutions for addressing multiple types of uncertainties that naturally arise in security games. These types of uncertainties include: 1) uncertainty in the attacker's payoff; 2) uncertainty in the attacker's rationality; and 3) uncertainty in the defender's strategy (w.r.t the defender's execution and the attacker's observation). In general, two different approaches have been pursued in previous work to handle uncertainties. The first approach models uncertainties using probability distributions and solves the resulting Bayesian Stackelberg game models (Kiekintveld et al., 2011; Yin & Tambe, 2012b); the second takes a robust optimization approach of maximizing defender expected utility under the worst case resulting from such uncertainties (Jiang et al., 2013; Kiekintveld et al., 2013; Pita et al., 2009; Yin et al., 2011). While the first approach assumes a known distribution of uncertainties beforehand, the second does not assume such prior knowledge. Since in many real world domains, including applications in counter-terrorism, we may lack data to generate a prior distribution, I thus focus on the second approach in this chapter.

As mentioned previously, one key weakness of all previous work in robust optimization in SSGs is that previous work compartmentalizes the uncertainties. The lack of a unified framework implies that existing algorithms suffer losses in solution quality in handling uncertainties in real-world security situations – where multiple types of uncertainties may exist simultaneously. This chapter remedies these weaknesses of state-of-the-art algorithms when addressing uncertainties in SSGs by providing the following key contributions. First, I am the first to present a unified computational framework – a single core problem representation – for handling the different types of uncertainties, as addressed so far in SSGs, and their combinations. Second,

based on this unified framework, I present a unified algorithmic framework from which I can derive different "unified" robust algorithms which address any combination of uncertainties in my framework, avoiding the compartmentalization mentioned above – no other previous algorithm can handle these combinations of uncertainties. Third, exploiting new insights from my framework, I present fast approximate algorithms for handling different subsets of uncertainties in large-scale security games. Finally, my experiments show the solution quality and runtime advantages of my algorithms.

6.1 A Unified Robust Framework

6.1.1 The Space of Uncertainties in SSGs

I first summarize the major types of uncertainties that have been studied in previous work as a 3-dimensional *uncertainty space*, shown in Figure 6.1. As shown in Figure 6.1, the three dimensions of the uncertainty space are: 1) uncertainty in the adversary's payoff; 2) uncertainty related to the defender's strategy; and 3) uncertainty in the adversary's rationality. These dimensions refer to three key aspects which directly affect both the defender and the adversary's utilities. The origin point of the uncertainty space corresponds to the case with perfectly rational adversary and no uncertainty in the adversary's payoff or related to the defender's strategy.

Uncertainty in the adversary's payoff has been addressed by ISG (Kiekintveld et al., 2013). Uncertainty in the defender's strategy can be classified into 2 cases, both addressed by RECON (Yin et al., 2011): 1) uncertainty in the defender's execution in which the executed strategy is different from the planned strategy of the defender; and 2) uncertainty in the adversary's observation of the defender's executed strategy.

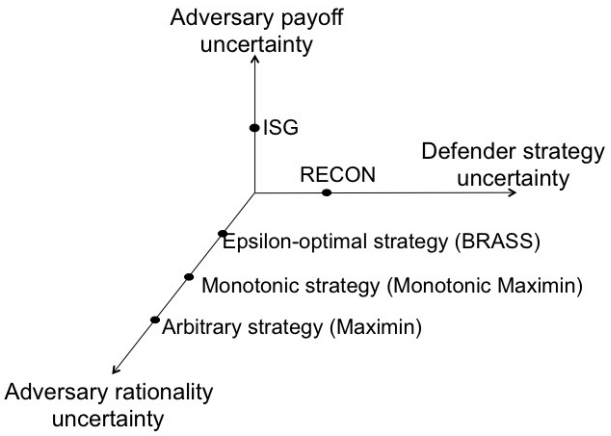


Figure 6.1: The uncertainty space

In the dimension of uncertainty in the adversary's rationality, the existing methods can be classified into 3 cases: 1) the adversary can choose any arbitrary strategy (Maximin); 2) the adversary can choose any strategy that satisfies the monotonic property (monotonic maximin (Jiang et al., 2013)); and 3) the adversary can choose any $\epsilon-$optimal strategy (BRASS (Pita et al., 2009)). It is known that computing the defender's strategy with the first and third cases is equivalent to a special case of uncertainty in adversary's payoff (Kiekintveld et al., 2013; Pita et al., 2009). Therefore, when dealing with rationality uncertainty, I will focus only on the case of monotonic adversary, i.e., monotonic maximin. As can be seen, the existing robust solution concepts attempt to address only a specific type of uncertainty and thus lie on axes of the space. Thus, I can identify combinations of different uncertainties which correspond to points not on any of the axes, that have not been addressed by previous works.

6.1.2 A General Formulation of Uncertainty Sets

The existing robust solution concepts for SSGs all follow a standard robust-optimization approach: first represent the uncertainty in the system as an *uncertainty set* of possible models, then choose the decision variables (defender strategy $\mathbf{x}$ in my case) such that the objective (defender utility) is optimized given the worst-case model from the uncertainty set. The main difference among the solution concepts is the way uncertainty sets are defined for different types of uncertainties. For example, in ISG, the uncertainty sets are intervals of adversary payoffs; in RECON, there is an hyper-rectangular uncertainty set around $\mathbf{x}$ representing the strategy executed by the defender, and another hyper-rectangular uncertainty set representing the defender strategy perceived by the adversary. In this work, I follow the same approach as previous work to represent each individual type of uncertainty in SSGs. In particular, given the defender's planned strategy $\mathbf{x}$, the real strategy that is executed lies within the range $H(\mathbf{x}) = \{\hat{\mathbf{x}} : \hat{x}_i \in [x_i - \gamma_i, x_i + \gamma_i] \cap [0, 1]\}$. Moreover, given that strategy, $\hat{\mathbf{x}}$, the defender's strategy perceived by the adversary according to his observations lies within the range $[\hat{x}_i - \eta_i, \hat{x}_i + \eta_i] \cap [0, 1]$. As a result, the defender's final strategy at target i perceived by the adversary can be any value within the range $[x_i - \gamma_i - \eta_i, x_i + \gamma_i + \eta_i] \cap [0, 1]$ where (γ_i, η_i) are given constants. Similarly, given an assumed adversary's payoff at target i, (R_i^a, P_i^a), the adversary's actual reward and penalty will lie within the ranges $[R_i^a - \alpha_i, R_i^a + \alpha_i]$ and $[P_i^a - \beta_i, P_i^a + \beta_i]$ respectively, where (α_i, β_i) are given

constants. Finally, in many cases such as counter-terrorism domains, the defender does not have sufficient data on the adversary's behaviors to accurately estimate the parameters of QR models. Instead, monotonic maximin assumes that the adversary can choose any strategy $\mathbf{y} \in \mathbf{Y}$ that has the following monotonicity property (which is satisfied by all known variants of QR models): $U_i^a(\mathbf{x}) \leq U_j^a(\mathbf{x}) \implies y_i \leq y_j$. In other words, the higher the expected utility of a target, the more likely the adversary will attack that target.

The first key component of my unified framework is a unified formulation of uncertainty sets for SSGs that captures all major existing approaches. Consider an SSG where all or any subset of the aforementioned uncertainties may be present. I begin by examining my objective function, which is the defender's expected utility $\sum_i y_i U_i^d(\mathbf{x})$. In general, $U_i^d(\mathbf{x})$ is affected by the uncertainty about the execution of defender strategies. The adversary strategy $\mathbf{y}$ will generally depend on his expected utilities $U_i^a(\mathbf{x})$ for all actions i, as well as how he makes decisions based on these expected utilities. Naturally, $\mathbf{y}$ is affected by the uncertainty about adversary rationality. Also, the uncertainties about adversary payoffs and adversary's observation of defender's strategy both affect $U_i^a(\mathbf{x})$, which in turn affects $\mathbf{y}$; furthermore, since the uncertainty about the defender's executed strategy will affect the adversary's observation of it, that in turn also affects $\mathbf{y}$.

Based on the above observations, I build an uncertainty set that captures all uncertainties that affect $U_i^d(\mathbf{x})$, and another for uncertainties that affect $\mathbf{y}$. The former task is simpler since only the execution uncertainty affects $U_i^d(\mathbf{x})$. The task of defining an uncertainty set for $\mathbf{y}$ is more complex because multiple types of uncertainties are involved. First of all, recall that execution uncertainty indirectly affects $\mathbf{y}$; but since I have already represented execution uncertainty using $H(\mathbf{x})$ above, I take an executed defender strategy $\hat{\mathbf{x}} \in H(\mathbf{x})$ as the input of my definition for the uncertainty set for $\mathbf{y}$. Specifically, given an executed defender strategy $\hat{\mathbf{x}} \in H(\mathbf{x})$, I define $\Psi(\hat{\mathbf{x}}) \subseteq \mathbf{Y}$ as the set of possible adversary strategies, resulting from all or any subset of uncertainties about adversary payoffs, adversary's observation, and adversary's rationality. In this paper I will consider the case with all uncertainties (Sections 6.1.3 and 6.2) as well as special cases with subsets of uncertainties (Sections 6.3 and 6.4), so it is important to have a definition of the uncertainty set $\Psi(\hat{\mathbf{x}})$ that is general and versatile. With that motivation in mind, I define the general form of $\Psi(\hat{\mathbf{x}})$ as follows.

Definition 2. *Given $\hat{\mathbf{x}} \in H(\mathbf{x})$, the uncertainty set $\Psi(\hat{\mathbf{x}}) \subseteq \mathbf{Y}$ is represented as a set of linear constraints on the adversary strategy $\mathbf{y}$. Specifically, there are K potential linear constraints on $\mathbf{y}$, each of which is activated or not depending on $\hat{\mathbf{x}}$. Formally,*

$$\Psi(\hat{\mathbf{x}}) = \{\mathbf{y} \in Y : \ D_k(\hat{\mathbf{x}}) \implies A_k(\mathbf{y}) \geq 0, k = \overline{1, K}\},$$

where $D_k(\hat{\mathbf{x}})$ is a disjunction (logical OR) of a set of conditions: $D_k(\hat{\mathbf{x}}) = \vee_s(D_{ks}(\hat{\mathbf{x}}) \geq 0)$[1] where $D_{ks} : \mathbf{X} \to \mathbb{R}$ are known scalar piecewise linear functions of $\mathbf{x}$. Finally, $A_k : \mathbf{Y} \to \mathbb{R}$ are known scalar linear functions of $\mathbf{y}$, i.e., $A_k(\mathbf{y}) = \sum_i \sigma_{ik} y_i$.

Then, the robust optimization problem of maximizing defender utility given worst-case uncertainty can be formulated as follows:

$$\textbf{P1}: \ \max_{\mathbf{x}} \min_{\hat{\mathbf{x}} \in H(\mathbf{x})} \min_{\mathbf{y} \in \Psi(\hat{\mathbf{x}})} \sum_i y_i U_i^d(\hat{\mathbf{x}})$$

Next, I show that P1 is sufficiently general, i.e., it can capture combinations of the previously-studied types of uncertainties.

6.1.3 Representation of Combined Uncertainties

I will focus on two cases in which specific formulations of the uncertainty set are different: 1) combinations of uncertainties with a rational adversary, i.e., uncertainty in the adversary's payoff and the defender's strategy; and 2) combinations of all other uncertainties with a monotonic adversary. Other points in the uncertainty space can then be separated into these two cases.

Consider an SSG where there is uncertainty in the adversary's payoff, i.e., for each target i, the adversary's reward and penalty lie within the range $[R_i^a - \alpha_i, R_i^a + \alpha_i]$ and $[P_i^a - \beta_i, P_i^a + \beta_i]$ respectively. Furthermore there is uncertainty about execution and adversary's observation of the defender strategy as in RECON, with the former represented by $H(\mathbf{x})$ and the latter represented by intervals $[\hat{x}_i - \eta_i, \hat{x}_i + \eta_i] \cap [0, 1]$ for all targets $i = \overline{1, T}$.

[1] *As we will see later in the paper, for certain types of uncertainties I will use strict inequalities $D_{ks}(\hat{\mathbf{x}}) > 0$ instead of weak inequalities $D_{ks}(\hat{\mathbf{x}}) \geq 0$.*

Given the defender's executed strategy $\hat{\mathbf{x}}$, the adversary's utility at target i will vary within the range $[\hat{U}^a_{min}(\hat{\mathbf{x}}, i), \hat{U}^a_{max}(\hat{\mathbf{x}}, i)]$ where the lower and upper bounds $\hat{U}^a_{min}(\hat{\mathbf{x}}, i)$ and $\hat{U}^a_{max}(\hat{\mathbf{x}}, i)$ are computed as the following:

$$\hat{U}^a_{min}(\hat{\mathbf{x}}, i) = (R^a_i - \alpha_i)(1 - \hat{x}^{max}_i) + (P^a_i - \beta_i)\hat{x}^{max}_i \tag{6.1}$$

$$\hat{U}^a_{max}(\hat{\mathbf{x}}, i) = (R^a_i + \alpha_i)(1 - \hat{x}^{min}_i) + (P^a_i + \beta_i)\hat{x}^{min}_i \tag{6.2}$$

where $\hat{x}^{max}_i = \min\{1, \hat{x}_i + \eta_i\}$ and $\hat{x}^{min}_i = \max\{0, \hat{x}_i - \eta_i\}$ are the lower bound and upper bound of the defender's coverage probability at target i w.r.t the adversary's observation.

Rational Adversary. Overall, target i could be potentially attacked by the adversary only when $\hat{U}^a_{max}(\hat{\mathbf{x}}, i) \geq \max_j\{\hat{U}^a_{min}(\hat{\mathbf{x}}, j)\}$. Otherwise, there always exists a target j even with all the uncertainties such that the adversary's utility at target j is greater than at target i, which means that the adversary will never attack target i. Therefore, in the uncertainty set, I have $K = T$ potential linear constraints with $A_k(\mathbf{y}) = -y_k$ and $D_k(\hat{\mathbf{x}}) = \vee_{s=\overline{1,T}}(\hat{U}^a_{min}(\hat{\mathbf{x}}, s) - \hat{U}^a_{max}(\hat{\mathbf{x}}, k) > 0)$ where $k = \overline{1, T}$. Then $(D_k(\hat{\mathbf{x}}) \implies A_k(\mathbf{y}) \geq 0)$ means that given target k, if there is a target $s \in \{1, 2, \ldots, T\}$ such that $\hat{U}^a_{min}(\hat{\mathbf{x}}, s) - \hat{U}^a_{max}(\hat{\mathbf{x}}, k) > 0$, then $y_k = 0$.

Example 1. *Figure 6.2 shows an example of a 2-target game with uncertainty intervals of the attacker's payoffs, the defender's execution and the attacker's observation. In particular, the uncertainty intervals of the attacker's reward and penalty at target* 1 *are* $[4, 6]$ *and* $[-4, -2]$. *Similarly, at target* 2*, we obtain intervals* $[0, 2]$ *and* $[-3, -1]$. *The defender's planned strategy is to protect target* 1 *and* 2 *with* 30% *and* 70% *of the time. Due to uncertainty, the actual strategy that the defender executes is to protect these two targets with probabilities lying within* $[0.2, 0.4]$ *and* $[0.6, 0.8]$ *respectively. Finally, the attacker can observe that the defender is protecting target* 1 *with probability within* $[0.1, 0.5]$ *and target* 2 *with probability within* $[0.5, 0.9]$. *As a result, suppose that the executed strategy is* $\hat{\mathbf{x}}$*, given the uncertainty set and the adversary is perfectly rational, there are two potential linear constraints, which is formulated as follows:*

$$\hat{U}^a_{min}(\hat{\mathbf{x}}, 2) > \hat{U}^a_{max}(\hat{\mathbf{x}}, 1) \implies y_1 = 0 \tag{6.3}$$

$$\hat{U}^a_{min}(\hat{\mathbf{x}}, 1) > \hat{U}^a_{max}(\hat{\mathbf{x}}, 2) \implies y_2 = 0 \tag{6.4}$$

Let's consider a particular executed strategy of the defender within the uncertainty interval, $\hat{\mathbf{x}} = \{0.2, 0.8\}$*, then the actual strategy of the defender which is observed by the adversary belongs*

Defender	Adversary			
	Target 1	**Target 2**	**Plan**	**Execution**
Target 1	**4, [-4, -2]**	**-1, [0, 2]**	**0.3**	**[0.2, 0.4]**
Target 2	**-5, [4, 6]**	**2, [-3, -1]**	**0.7**	**[0.6, 0.8]**
Observation	**[0.1, 0.5]**	**[0.5, 0.9]**		

Figure 6.2: A 2-target game with uncertainties

to the intervals $[0.1, 0.3]$ *and* $[0.7, 0.9]$ *of protecting target* 1 *and* 2 *respectively. Therefore, the lower and upper bounds of the attacker's utilities at target* 1 *and* 2 *is computed as:*

$$\hat{U}^a_{min}(\hat{\mathbf{x}}, 1) = 4 \times (1 - 0.3) + (-4) \times 0.3 = 1.6 \tag{6.5}$$

$$\hat{U}^a_{max}(\hat{\mathbf{x}}, 1) = 6 \times (1 - 0.1) + (-2) \times 0.1 = 5.2 \tag{6.6}$$

$$\hat{U}^a_{min}(\hat{\mathbf{x}}, 2) = 0 \times (1 - 0.9) + (-3) \times 0.9 = -2.7 \tag{6.7}$$

$$\hat{U}^a_{max}(\hat{\mathbf{x}}, 2) = 2 \times (1 - 0.7) + (-1) \times 0.7 = -0.1 \tag{6.8}$$

Since $\hat{U}^a_{max}(\hat{\mathbf{x}}, 2) < \hat{U}^a_{min}(\hat{\mathbf{x}}, 1)$*, only target* 1 *could be attacked by the adversary. Therefore, only the second potential constraint is activated while the first one is not activated. As a result, the set of possible adversary strategies* $\Psi(\hat{\mathbf{x}})$ *is* $\Psi(\hat{\mathbf{x}}) \equiv \{\mathbf{y} \in \mathbf{Y} : \ y_2 \leq 0\}$.

Monotonic Adversary. Overall, because the adversary is monotonic, for any pair of targets (i, j), the following constraint must hold: $\hat{U}^a_{min}(\hat{\mathbf{x}}, i) \geq \hat{U}^a_{max}(\hat{\mathbf{x}}, j) \implies y_i \geq y_j$. Conversely, there is no constraint on the attacking probability between target i and j if $\hat{U}^a_{min}(\hat{\mathbf{x}}, i) < \hat{U}^a_{max}(\hat{\mathbf{x}}, j)$ and $\hat{U}^a_{min}(\hat{\mathbf{x}}, j) < \hat{U}^a_{max}(\hat{\mathbf{x}}, i)$. Therefore, in the uncertainty set, I have $\frac{T(T-1)}{2}$ potential linear constraints indexed by $k = (i, j), i \neq j, i, j = \overline{1, T}$ with $A_k(\mathbf{y}) = y_i - y_j$ and $D_k(\hat{\mathbf{x}}) = (\hat{U}^a_{min}(\hat{\mathbf{x}}, i) - \hat{U}^a_{max}(\hat{\mathbf{x}}, j) \geq 0)$. In this case, there is only one condition in $D_k(\hat{\mathbf{x}})$, i.e., $s = 1$.

Example 2. *Given the same example as shown in Figure 6.2 but the adversary is monotonic, there are two potential linear constraints, which is formulated as follows:*

$$\hat{U}^a_{min}(\hat{\mathbf{x}}, 2) \geq \hat{U}^a_{max}(\hat{\mathbf{x}}, 1) \implies y_2 \geq y_1 \tag{6.9}$$

$$\hat{U}^a_{min}(\hat{\mathbf{x}}, 1) \geq \hat{U}^a_{max}(\hat{\mathbf{x}}, 2) \implies y_1 \geq y_2 \tag{6.10}$$

Since $\hat{U}^a_{max}(\hat{\mathbf{x}}, 2) < \hat{U}^a_{min}(\hat{\mathbf{x}}, 1)$*, only the second constraint is activated while the first one is not activated. As a result, we obtain the set of possible adversary strategies* $\Psi(\hat{\mathbf{x}})$ *is* $\Psi(\hat{\mathbf{x}}) \equiv \{\mathbf{y} \in \mathbf{Y} : \ y_1 \geq y_2\}$.

6.1.4 Uncertainty Dimension Reduction

Overall, **P1** is a multi-level complicated optimization problem which involves multiple and inter-dependent uncertainty variables. My first novel idea is to reduce the number of dimensions in the uncertainty space to simplify **P1** in terms of reducing the complexity of finding the worst-case scenario for the defender due to uncertainties. In general, it is not straightforward to integrate the three dimensions of the uncertainty space into a single uncertainty dimension as to simplify the max-min-min problem **P1** to a single maximin problem. Indeed, these uncertainty dimensions are inter-dependent which are difficult to unify because both the defender's executed strategy $\hat{\mathbf{x}}$ and the adversary's strategy $\mathbf{y}$ directly involve in the objective function $\sum_i y_i U^d_i(\hat{\mathbf{x}})$ while the feasible region $\Psi(\hat{\mathbf{x}})$ of $\mathbf{y}$ depends on $\hat{\mathbf{x}}$. As an important contribution, I show that the multi-dimension uncertainty space can be mapped into a uni-dimension space of the adversary's strategies (Figure 6.3). In other words, **P1** can be reformulated as a *single maximin* problem based on which I propose a unified robust algorithmic framework described in Section 6.2.

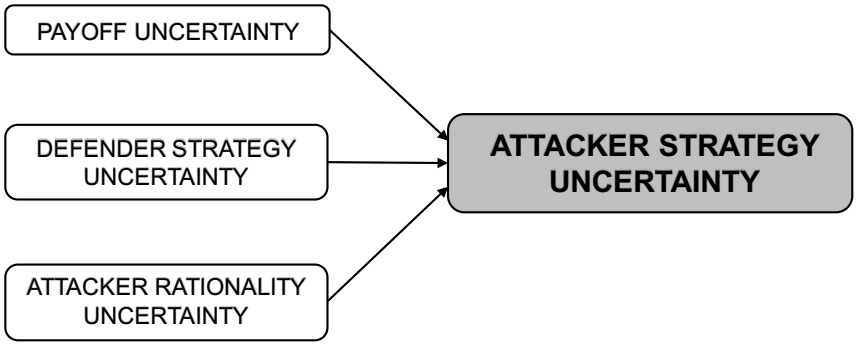


Figure 6.3: Mapping 3-dimensional to 1-dimensional uncertainty space

Given the defender's original strategy, $\mathbf{x}$, the adversary's utility at target i lies within the range $[U^a_{min}(\mathbf{x}, i), U^a_{max}(\mathbf{x}, i)]$ where $U^a_{min}(\mathbf{x}, i)$ and $U^a_{max}(\mathbf{x}, i)$ can be represented as the following:

$$U^a_{min}(\mathbf{x}, i) = (R^a_i - \alpha_i)(1 - x^+_i) + (P^a_i - \beta_i)x^+_i \tag{6.11}$$

$$U^a_{max}(\mathbf{x}, i) = (R^a_i + \alpha_i)(1 - x^-_i) + (P^a_i + \beta_i)x^-_i \tag{6.12}$$

where $x_i^+ = \min\{1, x_i + \gamma_i + \eta_i\}$ and $x_i^- = \max\{0, x_i - \gamma_i - \eta_i\}$. I define the set of the adversary's strategies, $L(\mathbf{x})$, as the following:

- Monotonic adversary: $L(\mathbf{x}) = \{\mathbf{y} \in \mathbf{Y} : U^a_{min}(\mathbf{x}, i) \geq U^a_{max}(\mathbf{x}, j) \implies y_i - y_j \geq 0\}$
- Rational adversary: $L(\mathbf{x}) = \{\mathbf{y} \in \mathbf{Y} : \vee_i (U^a_{min}(\mathbf{x}, i) - U^a_{max}(\mathbf{x}, j) > 0) \implies -y_j \geq 0\}$.

In addition, I define $U^d_{min}(\mathbf{x}, i)$ which is the lowest possible value of defender expected utility given that defender chose $\mathbf{x}$ and attacker chose i as follows:

$$U^d_{min}(\mathbf{x}, i) = \max\{0, x_i - \gamma_i\}(R^d_i - P^d_i) + P^d_i \tag{6.13}$$

Theorem 1. ***P1*** *is equivalent to the following single maximin problem* ***P2****:*

$$\textbf{\textit{P2}}: \max_{\mathbf{x} \in \mathbf{X}} \min_{\mathbf{y} \in L(\mathbf{x})} \sum_i y_i U^d_{min}(\mathbf{x}, i)$$

Proof. In this proof, I focus on combinations of uncertainties with a monotonic adversary which is the most difficult uncertainty case. Any other uncertainty cases could be solved in a similar way. Given the defender's original strategy, $\mathbf{x}$, denote $v_1 = \min_{\hat{\mathbf{x}} \in H(\mathbf{x})} \min_{\mathbf{y} \in \Psi(\hat{\mathbf{x}})} \sum_i y_i U^d_i(\hat{\mathbf{x}})$ and $v_2 = \min_{\mathbf{y} \in L(\mathbf{x})} \sum_i y_i U^d_{min}(\mathbf{x}, i)$, in the following, I show that $v_1 = v_2$.

Step 1. I will first prove that $v_1 \leq v_2$. The optimal solution of the inner minimization, $\min_{\mathbf{y} \in L(\mathbf{x})} \sum_i y_i U^d_{min}(\mathbf{x}, i)$, is an extreme point of $L(\mathbf{x})$. Moreover, all inequalities of $L(\mathbf{x})$ are of the form $y_i \geq 0$ and $y_i \geq y_j$ for certain pairs of targets (i, j). Any extreme point of $L(\mathbf{x})$ satisfies the following condition: For all (i, j), if $y_i, y_j > 0$, then $y_i = y_j$. Denote $I_a(\mathbf{x}) = \{i : y^*_i > 0,\ \mathbf{y}^* = \operatorname{argmin}_{\mathbf{y} \in L(\mathbf{x})} \sum_i y_i U^d_{min}(\mathbf{x}, i)\}$ be the support of the optimal strategy of the adversary given the defender's original strategy $\mathbf{x}$, I have $\forall i \in I_a(\mathbf{x}), y^*_i = \frac{1}{|I_a(\mathbf{x})|}$ and $\forall j \notin I_a(\mathbf{x}), y^*_j = 0$. Moreover, the following constraint must hold:

$$\forall i \in I_a(\mathbf{x}), j \notin I_a(\mathbf{x}) : U^a_{max}(\mathbf{x}, i) > U^a_{min}(\mathbf{x}, j). \tag{6.14}$$

Given the defender's original strategy $\mathbf{x}$ and the corresponding support $I_a(\mathbf{x})$, consider the following defender's executed strategy:

$$\hat{\mathbf{x}}^*_i = \begin{cases} \max\{0, x_i - \gamma_i\} & \text{, if } i \in I_a(\mathbf{x}) \\ \min\{1, x_i + \gamma_i\} & \text{, otherwise} \end{cases}$$

According to the definition of $H(\mathbf{x})$, I have: $\hat{\mathbf{x}}^* \in H(\mathbf{x})$. In addition, I have: $U^a_{max}(\hat{\mathbf{x}}^*, i) = U^a_{max}(\mathbf{x}, i)$ for all $i \in I_a(\mathbf{x})$ and $U^a_{min}(\hat{\mathbf{x}}^*, j) = U^a_{min}(\mathbf{x}, j)$ for all $j \notin I_a(\mathbf{x})$. Therefore, according to (6.14), I have $\mathbf{y}^* \in \Psi(\hat{\mathbf{x}}^*)$. On the other hand, for all $i \in I_a(\mathbf{x})$, I obtain $U^d_i(\hat{\mathbf{x}}^*) = U^d_{min}(\mathbf{x}, i)$. As the result, we obtain the following inequality:

$$v_1 \leq \min_{\mathbf{y} \in \Psi(\hat{\mathbf{x}}^*)} \sum_i y_i U^d_i(\hat{\mathbf{x}}^*) \leq \sum_i y^*_i U^d_i(\hat{\mathbf{x}}^*) = \frac{\sum\limits_{i \in I_a(\mathbf{x})} U^d_{min}(\mathbf{x}, i)}{|I_a(\mathbf{x})|} = v_2.$$

Step 2. Now, I am going to prove that $v_1 \geq v_2$. Let $\hat{\mathbf{x}} \in H(\mathbf{x})$ be an executed strategy of the defender, I have: $U^a_{min}(\mathbf{x}, i) \leq \hat{U}^a_{min}(\hat{\mathbf{x}}, i)$ and $U^a_{max}(\mathbf{x}, i) \geq \hat{U}^a_{max}(\hat{\mathbf{x}}, i)$, for all i. Therefore, given any pair of targets (i, j), if $U^a_{min}(\mathbf{x}, i) \geq U^a_{max}(\mathbf{x}, j)$, then $\hat{U}^a_{min}(\hat{\mathbf{x}}, i) \geq \hat{U}^a_{max}(\hat{\mathbf{x}}, j)$ which implies that $\forall \mathbf{y} \in \Psi(\hat{\mathbf{x}}), y_i \geq y_j$. As the result, $\forall \mathbf{y} \in \Psi(\hat{\mathbf{x}})$, I obtain the following condition: $U^a_{min}(\mathbf{x}, i) \geq U^a_{max}(\mathbf{x}, j) \implies y_i \geq y_j$. According to the definition of $L(\mathbf{x})$, I have: $\mathbf{y} \in L(\mathbf{x})$. Therefore, $L(\mathbf{x}) \supseteq \Psi(\hat{\mathbf{x}})$. As the result, $L(\mathbf{x}) \supseteq \bigcup\limits_{\hat{\mathbf{x}} \in H(\mathbf{x})} \Psi(\hat{\mathbf{x}})$. Because $L(\mathbf{x}) \supseteq \bigcup\limits_{\hat{\mathbf{x}} \in H(\mathbf{x})} \Psi(\hat{\mathbf{x}})$ and $U^d_{min}(\mathbf{x}, i) \leq U^d_i(\hat{\mathbf{x}})$ for all i, I obtain:

$$v_2 = \min_{\mathbf{y} \in L(\mathbf{x})} \sum_i y_i U^d_{min}(\mathbf{x}, i) \leq \min_{\mathbf{y} \in \bigcup\limits_{\hat{\mathbf{x}} \in H(\mathbf{x})} \Psi(\hat{\mathbf{x}})} \sum_i y_i U^d_{min}(\mathbf{x}, i) \tag{6.15}$$

$$= \min_{\hat{\mathbf{x}} \in H(\mathbf{x})} \min_{\mathbf{y} \in \Psi(\hat{\mathbf{x}})} \sum_i y_i U^d_{min}(\mathbf{x}, i) \leq \min_{\hat{\mathbf{x}} \in H(x)} \min_{\mathbf{y} \in \Psi(\hat{\mathbf{x}})} \sum_i y_i U^d_i(\hat{\mathbf{x}}) = v_1 \tag{6.16}$$

By combining Step 1 and Step 2, I show that $v_1 = v_2$ for all $\mathbf{x}$. Therefore, $P1 \equiv P2$. □

To that end, as $L(\mathbf{x})$ exhibits the same structure as $\Psi(\hat{\mathbf{x}})$, I can represent this uncertainty set as a set of potential linear constraints which are specified according to:

$$L(\mathbf{x}) = \{\mathbf{y} \in Y : \ D_k(\mathbf{x}) \implies A_k(\mathbf{y}) \geq 0, k = \overline{1, K}\}$$

In particular, in the case of combined uncertainties with a rational adversary, for all $k = \overline{1, T}$, I obtain a set of potential linear constraints with:

$$A_k(\mathbf{y}) = -y_k$$
$$D_k(\mathbf{x}) = \vee_s(U^a_{min}(\mathbf{x}, s) - U^a_{max}(\mathbf{x}, k) > 0)$$

Similarly, in the case of combined uncertainties with a monotonic adversary, for all $k = (i,j), i,j = \overline{1,T}$, the set of potential linear constraints are determined based on:

$$A_{(i,j)}(\mathbf{y}) = y_i - y_j$$
$$D_{(i,j)}(\mathbf{x}) = (U^a_{min}(\mathbf{x},i) - U^a_{max}(\mathbf{x},j) \geq 0)$$

Example 3. *In the 2-target game shown in Figure 6.2, given the defender's planned strategy,* $\mathbf{x} = \{0.3, 0.7\}$*, the lower and upper bounds of the attacker's utilities at targets* 1 *and* 2 *w.r.t the defender's planned strategy are computed as:*

$$U^a_{min}(\mathbf{x},1) = 4 \times (1 - 0.5) + (-4) \times 0.5 = 0 \tag{6.17}$$
$$U^a_{max}(\mathbf{x},1) = 6 \times (1 - 0.1) + (-2) \times 0.1 = 5.2 \tag{6.18}$$
$$U^a_{min}(\mathbf{x},2) = 0 \times (1 - 0.9) + (-3) \times 0.9 = -2.7 \tag{6.19}$$
$$U^a_{max}(\mathbf{x},2) = 2 \times (1 - 0.5) + (-1) \times 0.5 = 0.5 \tag{6.20}$$

In the case that the adversary is perfectly rational, there are two potential linear constraints which are determined as follows:

$$U^a_{min}(\mathbf{x},1) - U^a_{max}(\mathbf{x},2) > 0 \implies -y_2 \geq 0$$
$$U^a_{min}(\mathbf{x},2) - U^a_{max}(\mathbf{x},1) > 0 \implies -y_1 \geq 0$$

Since $U^a_{min}(\mathbf{x},1) - U^a_{max}(\mathbf{x},2) < 0$ *and* $U^a_{min}(\mathbf{x},2) - U^a_{max}(\mathbf{x},1) < 0$*, both the constraints are not activated. As a result, the uncertainty set of the adversary's strategies is:* $L(\mathbf{x}) = \{\mathbf{y} \in \mathbf{Y}\}$

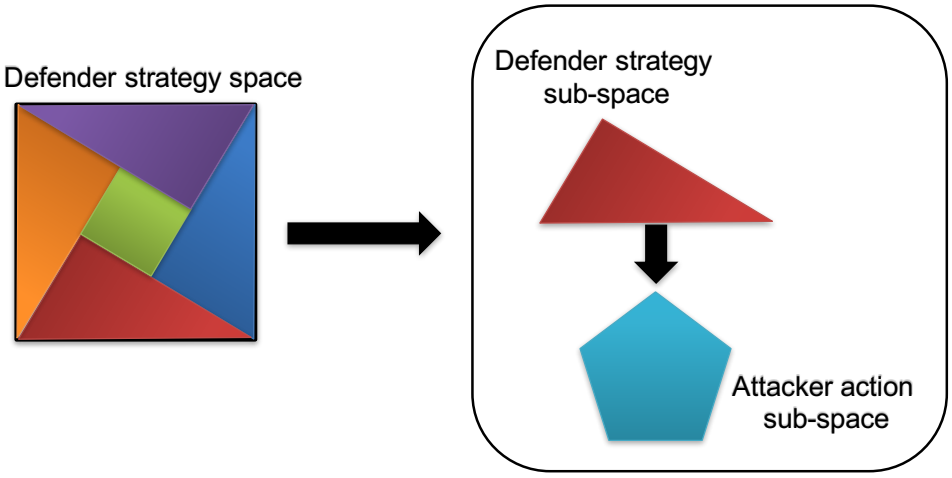
In the case that the adversary is monotonic, there are also two potential linear constraints which are determined as follows:

$$U^a_{min}(\mathbf{x},1) - U^a_{max}(\mathbf{x},2) \geq 0 \implies y_1 \geq y_2$$
$$U^a_{min}(\mathbf{x},2) - U^a_{max}(\mathbf{x},1) \geq 0 \implies y_1 \leq y_2$$

Again, since $U^a_{min}(\mathbf{x},1) - U^a_{max}(\mathbf{x},2) < 0$ *and* $U^a_{min}(\mathbf{x},2) - U^a_{max}(\mathbf{x},1) < 0$*, both the above constraints for the monotonic adversary are not activated. As a result, the uncertainty set of the adversary's strategies is specified as:* $L(\mathbf{x}) = \{\mathbf{y} \in \mathbf{Y}\}$.

6.2 Unified Robust Algorithm

Unlike the standard simple maximin problem for zero-sum games, in my maximin problem **P2**, the uncertainty set for the adversary's strategy, $L(\mathbf{x})$, depends on the defender's strategy $\mathbf{x}$, making **P2** significantly harder to solve. More specifically, each strategy of the defender $\mathbf{x}$ will lead to a different uncertainty set of the adversary's strategy $L(\mathbf{x})$ — which is determined by a different set of activated linear constraints. A naive approach to solve **P2** is to iterate over all possible strategies for the defender to find the optimal one. However, since the strategy space of the defender is infinite, this approach is infeasible. Therefore, I propose a divide-and-conquer method to overcome this challenge which decouples the dependency of the adversary's strategies on the defender's strategies. Essentially, I divide **P2** into sub-maximin problems; each sub-problem is associated with a subset of the defender's strategies and a uncertainty subset of the adversary's strategies which is independent from the defender's strategies (Figure 6.4). In other words, every strategy within each subset of the defender's strategies corresponds to the same uncertainty subset of the adversary's strategies. These sub-maximin problems thus can be solved using a standard optimization approach which I will explain later. Finally, **P2** is solved by combining the resulting sub-optimal solutions to find the global optimal solution.



(a) Cluster defender's strategy space.

(b) Divide **P2** into sub-maximin problems; each corresponds to a pair of defender and attacker's strategy sub-spaces.

Figure 6.4: Overview of URAC's divide-and-conquer

6.2.1 Divide-and-Conquer

I first define a subset $C(\mathbf{x})$ as $\{k \in \{1, 2, \ldots, K\} : D_k(\mathbf{x})\}$ which refers to the set of activated linear constraints on $\mathbf{y}$, i.e., $\forall k \in C(\mathbf{x}), A_k(\mathbf{y}) \geq 0$. I define a $T \times K$-constraint matrix $M(C(\mathbf{x}))$ as the following: for all $i = \overline{1, T}$ and $k = \overline{1, K}$:

$$M(C(\mathbf{x}))_{ik} = \begin{cases} \sigma_{ik} & \text{, if } k \in C(\mathbf{x}) \\ 0 & \text{, otherwise} \end{cases}$$

Proposition 2. *The set of the adversary's strategies, $L(\mathbf{x})$, can be formulated as $L(\mathbf{x}) = \{\mathbf{y} \in \mathbf{Y} : [M(C(\mathbf{x}))]' \mathbf{y} \geq 0\}$ where $[M(C(\mathbf{x}))]'$ is the transposed matrix of $M(C(\mathbf{x}))$.*

Proof. According to the definition of $M(C(\mathbf{x}))$, I have:

$$\left([M(C(\mathbf{x}))]' \mathbf{y}\right)_k = \begin{cases} \sum_i \sigma_{ik} y_i & \text{, if } k \in C(\mathbf{x}) \\ 0 & \text{, otherwise} \end{cases}$$

Therefore, $[M(C(\mathbf{x}))]' \mathbf{y} \geq 0 \iff \forall k \in C(\mathbf{x}),\ A_k(\mathbf{y}) \geq 0$. □

Example 4. *Let us consider an example of a 3-target game in which the adversary is monotonic. Given a defender's planned strategy $\mathbf{x}$, suppose that we obtain:*

$$U^a_{min}(\mathbf{x}, 1) \geq U^a_{max}(\mathbf{x}, 2)$$
$$U^a_{min}(\mathbf{x}, 1) \geq U^a_{max}(\mathbf{x}, 3)$$

which means that the conditions $D_{(1,2)}(\mathbf{x})$ and $D_{(1,3)}(\mathbf{x})$ are true. As a result, the constraints $A_{(1,2)}(\mathbf{y}) = y_1 - y_2 \geq 0$ and $A_{(1,3)}(\mathbf{y}) = y_1 - y_3 \geq 0$ must hold on $\mathbf{y}$. In other words, I have $C(\mathbf{x}) = \{(1, 2), (1, 3)\}$. Furthermore, since $A_{(i,j)}(\mathbf{y}) = y_i - y_j,\ \forall i, j = \overline{1, 3}$, the coefficients $\sigma_{ik} = 1$ and $\sigma_{jk} = -1$ with $k = (i, j)$. Thus, $M(C(\mathbf{x}))$ is represented as the following:

$$\begin{pmatrix} (1,2) & (1,3) & (2,1) & (2,3) & (3,1) & (3,2) \\ 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Although the constraint matrix depends on the defender's strategy $\mathbf{x}$, the set of $M(C(\mathbf{x}))$ is finite. Specifically, $M(C(\mathbf{x}))$ is a function of the subset $C(\mathbf{x}) \subseteq \{1, 2, ..., K\}$. In other

words, each subset of $\{1, 2, ..., K\}$ corresponds to a unique constraint matrix. Because the set $\{1, 2, ..., K\}$ has 2^K subsets, there are at most 2^K constraint matrices. My key idea of the unified algorithm is to conceptually divide the original optimization problem into multiple sub-optimization problems according to the constraint matrix $M(C(\mathbf{x}))$. Given a constraint matrix $M(C)$ where $C \subseteq \{1, 2, ..., K\}$, the corresponding sub-optimization problem is defined as

$$\max_{\mathbf{x} \in S_C^d} \min_{\mathbf{y} \in S_C^a} \sum_i y_i U_{min}^d(\mathbf{x}, i) \tag{6.21}$$

where S_C^d and S_C^a are in turn the sets of the defender's and adversary's strategies corresponding to C. In particular, $S_C^d = \{\mathbf{x} \in \mathbf{X} : C(\mathbf{x}) = C\} = \{\mathbf{x} \in \mathbf{X} : \vee_s(D_{ks}(\mathbf{x}) \geq 0)\ \forall k \in C,\ \wedge_s(D_{ks}(\mathbf{x}) < 0)\ \forall k \notin C\}$ and $S_C^a = \{\mathbf{y} \in \mathbf{Y} : [M(C)]'\mathbf{y} \geq 0\}$. In other words, for all $\mathbf{x} \in S_C^d$, $L(\mathbf{x}) = S_C^a$. Thus, the inner minimization of (6.21) can be represented as:

$$\min_{\mathbf{y}} \sum\nolimits_i y_i U_{min}^d(\mathbf{x}, i) \tag{6.22}$$

$$[M(C)]'\mathbf{y} \geq 0 \tag{6.23}$$

$$\sum\nolimits_i y_i = 1, 0 \leq y_i \leq 1. \tag{6.24}$$

Replacing this LP with its dual, the sub-optimization problem (6.21) can be formulated as the following single maximization problem:

$$\max_{\mathbf{x}, \boldsymbol{\theta}, t} t \tag{6.25}$$

$$(M(C)\boldsymbol{\theta})_i + t \leq U_{min}^d(\mathbf{x}, i),\ \forall i = \overline{1, T} \tag{6.26}$$

$$\mathbf{x} \in S_C^d \tag{6.27}$$

$$\boldsymbol{\theta} \geq 0. \tag{6.28}$$

where $\boldsymbol{\theta}$ is dually associated with the constraint (6.23).

Example 5. *In a 3-target game with a monotonial adversary, considering a constraint matrix $C = \{(1, 2), (1, 3)\}$, then the constraint matrix $M(C)$ can be constructed as shown in Example 4. The two sets of the defender's and the adversary's strategies are determined as follows:*

$$S_C^d = \{\mathbf{x} \in \mathbf{X} :\ U_{min}^a(\mathbf{x}, 1) \geq U_{max}^a(\mathbf{x}, 2) \text{ and } U_{min}^a(\mathbf{x}, 1) \geq U_{max}^a(\mathbf{x}, 3)\}$$

$$S_C^a = \{\mathbf{y} \in \mathbf{Y} :\ y_1 \geq y_2 \text{ and } y_1 \geq y_3\}$$

Finally, by introducing integer variables to encode the OR operators in S_C^d, each sub-optimization problem (6.25-6.28) can be solved as a MILP. The final optimization solution of **P2** can be computed as the maximum of all these sub-optimization problems. However, there is an exponential number, i.e., 2^K, of such sub-optimization problems that I need to solve. In the next section, I introduce a single MILP representation for more efficiently solving **P2**.

6.2.2 URAC: Unified Algorithmic Framework

Overall, I define an integer vector $\mathbf{z} \in \{0,1\}^K$ which encodes the set C. Specifically, given a subset $C \subseteq \{1,2,...,K\}$, then $z_k = 1$ if $k \in C$; otherwise, $z_k = 0$. As a result, I have: $M(C)_{ik} = z_k \sigma_{ik}$. Therefore, by using z to refer to 2^K sub-optimization problems, I obtain the general MILP (6.29-6.37).

Example 6. *In the case of a monotonic adversary, the integer vector* $\mathbf{z}$ *is defined as* $z_{(i,j)} = 1$ *when* $(i,j) \in C$ *for a pair of targets* (i,j)*, which indicates that the adversary will attack target* i *with higher probability than attacking target* j*. Conversely,* $z_{(i,j)} = 0$ *when* $(i,j) \notin C$*. On the other hand, in the case of a perfectly rational adversary, the integer vector* $\mathbf{z}$ *is defined as* $z_i = 1$ *when* $i \in C$ *for target* i*, implying that target* i *can be attacked by the adversary.*

Overall, this MILP with an instantiation of $\mathbf{z}$ is equivalent to a specific sub-optimization problem (6.25–6.28). Since the MILP optimizes over all possible values of $\mathbf{z}$, it computes the maximum of all these sub-optimization problems, which is the optimal solution of **P2**. In particular, constraints (6.26) and (6.28) correspond to constraints (6.30–6.31). In constraint (6.26), I can rewrite the first term $(M(C)\boldsymbol{\theta})_i$ as $\sum_k z_k \sigma_{ik} \theta_k$ which is a quadratic expression. I apply a standard technique to transform it to a linear expression. Specifically, I define a new variable $\phi_k = z_k \theta_k$. I have $z_k = 0 \implies \phi_k = 0$. On the other hand, $z_k = 1 \implies \phi_k = \theta_k$ where $0 \leq \theta_k$,

which means that I only need constraints $0 \leq \phi_k$. As a result, I have the following constraints on ϕ_k: $0 \leq \phi_k \leq N z_k$ where N is a sufficiently large constant.

$$\max_{\mathbf{x},\mathbf{z},\boldsymbol{\phi},\mathbf{q},t} \; t \tag{6.29}$$

$$\sum\nolimits_k \sigma_{ik}\phi_k + t \leq U^d_{min}(\mathbf{x}, i),\ \forall i \tag{6.30}$$

$$0 \leq \phi_k \leq N z_k,\ \forall k \tag{6.31}$$

$$r_k + (1 - z_k)N \geq 0,\ \forall k \tag{6.32}$$

$$r_k - z_k N < 0,\ \forall k \tag{6.33}$$

$$r_k \geq D_{ks}(\mathbf{x}),\ \forall k, s \tag{6.34}$$

$$r_k \leq D_{ks}(\mathbf{x}) + (1 - q_{ks})N,\ \forall k, s \tag{6.35}$$

$$\sum\nolimits_i x_i \leq R,\ x_i \in [0, 1] \tag{6.36}$$

$$\sum\nolimits_s q_{ks} = 1,\ \forall k, z_i, q_{ks} \in \{0, 1\}. \tag{6.37}$$

Furthermore, the feasible set of the defender's strategies, S^d_C, in constraint (6.27) is computed according to the constraints (6.32–6.36). Specifically, the constraint $\{\vee_s(D_{ks}(\mathbf{x}) \geq 0)\ \forall k \in C\}$ of S^d_C can be replaced as constraint (6.32) where $r_k = \max_s D_{ks}(\mathbf{x})$ can be determined by constraints (6.34–6.35). In addition, the constraint $\{\wedge_s(D_{ks}(\mathbf{x}) < 0)\ \forall k \notin C\}$ of S^d_C corresponds to constraint (6.33). MILP solvers generally can not directly deal with strict inequality constraints like (6.33). In my implementation, I replace (6.33) with $r_k + \epsilon - z_k N \leq 0,\ \forall k, s$, where ϵ is a small positive constant. This usage of ϵ is consistent with previous formulations such as RECON and ISG (Kiekintveld et al., 2013; Yin et al., 2011).

Finally, in order to express $D_{ks}(\mathbf{x})$ and $U^d_{min}(\mathbf{x}, i)$ which are piecewise linear functions of $\mathbf{x}$, I need extra integer variables. For example, I can compute $U^d_{min}(\mathbf{x}, i)$ (Equation (6.13)) using integer variable $h_i \in \{0, 1\}$ as the following:

$$P^d_i \leq U^d_{min}(\mathbf{x}, i) \leq P^d_i + h_i N \tag{6.38}$$

$$b_i \leq U^d_{min}(\mathbf{x}, i) \leq b_i + (1 - h_i)N \tag{6.39}$$

where $b_i = (x_i - \gamma_i)(R^d_i - P^d_i) + P^d_i$. I can determine $U^a_{min}(\mathbf{x}, i)$ and $U^a_{max}(\mathbf{x}, i)$ in $D_{ks}(\mathbf{x})$ in a similar way.

I refer to the MILP (6.29-6.37) as the Unified Robust Algorithmic framework for addressing unCertainties (URAC). By replacing $D_k(\mathbf{x})$ and $A_k(\mathbf{x})$ with specific formulations, I obtain a version of URAC for addressing a particular type of uncertainty, i.e., when $A_k(\mathbf{y}) = -y_k$ and $D_k(\mathbf{x}) = \vee_s(U^a_{min}(\mathbf{x}, s) - U^a_{max}(\mathbf{x}, k) > 0)$, the corresponding version of URAC addresses a combined uncertainty with a rational adversary. In fact, no previous robust algorithm could handle all these combinations of uncertainties.

MILP relaxation: I approximate the piecewise linear functions $U^a_{min}(\mathbf{x}, i), U^a_{max}(\mathbf{x}, i)$ and $U^d_{min}(\mathbf{x}, i)$ with linear functions to reduce the computational complexity of URAC, e.g., I can replace $U^d_{min}(\mathbf{x}, i)$ as $U^d_{min}(\mathbf{x}, i) = (x_i - \gamma_i)(R^d_i - P^d_i) + P^d_i$. I also approximate $U^a_{min}(\mathbf{x}, i)$ and $U^a_{max}(\mathbf{x}, i)$ similarly. I refer to this approximate algorithm as a-URAC.

6.3 A Scalable Robust Algorithm I

Although (a-)URAC can handle any type of uncertainty in the uncertainty space, these algorithms struggle to scale up to larger problems, due to having potentially large numbers of integer variables. Nevertheless, having the general formulation of uncertainty sets allows us to make the following observation: the constraint functions $A_k(\mathbf{y})$ exhibit two different important properties depending on the types of uncertainties under consideration: 1) in the case of combined uncertainties with a rational adversary, $A_k(\mathbf{y})$ imposes constraints on the targets separately, i.e., $A_k(\mathbf{y}) = -y_k$; 2) in the case of combined uncertainties with a monotonic adversary, $A_k(\mathbf{y})$ involves multiple targets into constraints, i.e., $A_{(i,j)}(\mathbf{y}) = y_i - y_j$. By using these properties, in the next two sections I present two scalable algorithms.

In the case of **combined uncertainties with a rational adversary** (Group 1), overall, I want to apply the binary search method to iteratively search through the space of the defender's utility. At each iteration of the binary search, I need to determine if there exists a feasible solution of the defender's strategy, $\mathbf{x}$, such that $\min_{\mathbf{y} \in L(\mathbf{x})} \sum_i y_i U^d_{min}(\mathbf{x}, i) \geq t$ where t is a given value. This corresponds to a feasibility version of the MILP (6.29-6.37), where given t I are asked to find a feasible $\mathbf{x}$. At a high level, because $A_k(\mathbf{y})$ includes only a single target k, constraints (6.30-6.35) of $\mathbf{x}$ can be decomposed into separate constraints of x_i. Then by examining the conditions on the defender's coverage at every target independently, I can determine if a utility value t is feasible.

In particular, constraints (6.30-6.35) provide conditions by which the linear constraint $A_j(\mathbf{y})$ at target j for all $j = \overline{1,T}$ is (not) activated. Denote by $r = \max_j U^a_{min}(\mathbf{x}, j)$ the maximum value of the adversary's lowest utilities over all targets. I call r the adversary's "cut-off" utility. In addition, I define $i = \text{argmax}_j\, U^a_{min}(\mathbf{x}, j)$ as the "cut-off" target. In fact, if t and i are known in advance, constraints (6.30-6.35) reduce to $x_j \geq x_j^{min}$, where x_j^{min} is the required minimum coverage probability of the defender at target j (which I will explain in detail later). As a result, I can determine the defender's minimum coverage, $\{x_j^{min}\}_j$, such that the lowest utility of the defender is t and the "cut-off" target is i. Therefore, given t, minimum amount of resources required is $R^{min} = \min_i\{\sum_j x_j^{min} | i \leftarrow \text{"cut-off" target}\}$.

Hence, t is a feasible utility for the defender only when $R^{min} \leq R$ (Constraint 6.36). By following the binary search approach, I obtain a scalable algorithm called $\delta-$Optimal Robust Algorithm for Addressing unCertainties ($\delta-$ORAC). This algorithm guarantees an $\delta-$optimal solution for addressing uncertainties in this group where δ is a given positive small value. $\delta-$ORAC is a generalization of ISG which only deals with uncertainty in the adversary's payoff (Kiekintveld et al., 2013). $\delta-$ORAC arises out of my unified framework and then with $\gamma = \eta = 0$, it becomes equivalent to ISG, whereas with $\alpha = \beta = 0$, it becomes a robust algorithm for dealing with uncertainty in the defender's strategy.

Finally, given a feasible utility t and the "cut-off" target i, x_j^{min} for all $j = \overline{1,T}$ can be determined as follows. As r_k can be computed as $r_k = \max_s D_{ks}(\mathbf{x}) = r - U^a_{max}(\mathbf{x}, k)$, the MILP constraints (6.30-6.35) reduce to the following conditions:

Defender utility condition for targets at which linear constraints are not activated: For all such targets j, I have $z_j = 0$. As $\sigma_{jk} = -1$ if $j = k$; otherwise, $\sigma_{jk} = 0$, constraint (6.30) can be reduced to the following:

$$-\phi_j + t \leq U^d_{min}(\mathbf{x}, j),\ \forall j = \overline{1,T}$$

which implies that if $z_j = 0$, $U^d_{min}(\mathbf{x}, j) \geq t$ or equivalently, $x_j^{min} \geq m_j^d$ where m_j^d is the minimum coverage of the defender on target j ensuring that $U^d_{min}(\mathbf{x}, j)$ is at least t. In particular, if $P_j^d \geq t$, then $m_j^d = 0$. Otherwise, $m_j^d = \frac{t - P_j^d}{R_j^d - P_j^d} + \gamma_j$.

In addition, constraint (6.33) can be formulated as the following:[2]

$$r - U^a_{max}(\mathbf{x}, j) - z_j N \leq 0,\ \forall j = \overline{1,T} \tag{6.40}$$

which implies that $U^a_{max}(\mathbf{x}, j) \geq r$ if $z_j = 0$.

As the linear constraint at the "cut-off" target i is always not activated, I have: $x_i^{min} = m_i^d$. Thus, the adversary's "cut-off" utility can be computed as $r = U^a_{min}(\mathbf{x}, i) = \max\{P_i^a - \beta_i, (R_i^a - \alpha_i)(1 - m_i^d - \gamma_i - \eta_i) + (P_i^a - \beta_i)(m_i^d + \gamma_i + \eta_i)\}$.

Adversary utility condition for targets at which linear constraints are activated: For all such targets j, I have $z_j = 1$. Constraint (6.32) can be simplified as the following:

$$r - U^a_{max}(\mathbf{x}, j) + (1 - z_j)N > 0,\ \forall j = \overline{1,T} \tag{6.41}$$

which implies that $U^a_{max}(\mathbf{x}, j) < r$ when $z_j = 1$. I approximate this constraint by $U^a_{max}(\mathbf{x}, j) \leq r - \epsilon$, where ϵ is a small positive constant.

Thus, if $z_j = 1$, then $x_j^{min} \geq m_j^a$ where $m_j^a = 0$ if $R_j^a + \alpha_j \leq r - \epsilon$. Otherwise, $m_j^a = \frac{R_j^a + \alpha_j - r + \epsilon}{R_j^a + \alpha_j - (P_j^a + \beta_j)} + \gamma_j + \eta_j$.

As z_j is either 0 or 1, I obtain: $x_j^{min} \geq \min\{m_j^d, m_j^a\}$.

Cut-off utility condition for all targets: Finally, I have:

$$U^a_{min}(\mathbf{x}, j) \leq r,\ \forall j \tag{6.42}$$

This constraint implies: $x_j^{min} \geq m_j^k$ for all target j where $m_j^k = \max\{0, \frac{R_j^a - \alpha_j - r}{R_j^a - \alpha_j - (P_j^a - \beta_j)} - \gamma_j - \eta_j\}$. In fact, the constraint (6.42) is equivalent to constraint (6.34).

As a result, I can determine the smallest necessary amount of defender's resources at every target j as follows:

$$x_j^{min} = \max\{m_j^k, \min\{m_j^d, m_j^a\}\} \tag{6.43}$$

Moreover, if $\{x_j^{min}\}_j$ satisfy constraint (6.36), the defender's strategy $\mathbf{x}$ with $x_j = x_j^{min}$ for all j is a feasible solution given t.

Example 7. *Given the example of a 2-target game with uncertainties shown in Figure 6.2 and the adversary is perfectly rational, let's consider a utility of the defender $t = 0.0$. Now, I am trying*

[2] As $D_k(\mathbf{x})$ refers to strict inequalities in this case of uncertainty (Section 6.1.4) which differs from Definition 2, constraint (6.40) is not a strict inequality as (6.33).

to determine if there exists a feasible strategy $\mathbf{x}$ *such that the defender receives a utility which is no less than* t *for playing* $\mathbf{x}$*. I consider two cases of the "cut-off" target:*

Case 1–Target 1 is the cut-off target. *There are three conditions that need to be satisfied:*

- ***Defender utility condition:*** *Since target 1 is the cut-off target, i.e.,* $U^a_{min}(\mathbf{x}, 1) \geq U^a_{min}(\mathbf{x}, 2)$*, the linear constraint at target 1 is not activated which means that* $U^d_{min}(\mathbf{x}, 1) \geq t \implies 4 \times (x_1 - 0.1) + (-5) \times (1 - x_1 + 0.1) \geq 0 \implies x_1 \geq 0.66 \implies m^d_1 \approx 0.66$*.*

  Furthermore, if the linear constraint at target 2 is not activated, I obtain: $U^d_{min}(\mathbf{x}, 2) \geq t \implies 2 \times (x_2 - 0.1) + (-1) \times (1 - x_2 + 0.1) \geq 0 \implies x_2 \geq \frac{1.3}{3} \implies m^d_2 = \frac{1.3}{3} \approx 0.43$*.*

  Finally, the adversary's cut-off utility is computed as $r = U^a_{min}(\mathbf{x}, 1) = \max\{-4, 4 \times (1 - m^d_1 - 0.2) + (-4) \times (m^d_1 + 0.2)\} \approx -2.84$*.*

- ***Adversary utility condition:*** *Since this condition is only applied for targets at which linear constraints are activated, I only need to examine target 2 (target 1 is the cut-off target at which linear constraints are always not activated). If the linear constraint at target 2 is activated, I obtain:* $U^a_{max}(\mathbf{x}, 2) \leq r \implies 2 \times (1 - x_2 + 0.2) + (-1) \times (x_2 - 0.2) \leq -2.84$ *which implies that* $m^a_2 \approx 1.81$*.*

- ***Cut-off utility condition:*** *Finally, the condition on the cut-off utility of the adversary* $U^a_{min}(\mathbf{x}, 2) \leq r \implies 0 \times (1 - x_2 - 0.2) + (-3) \times (x_2 + 0.2) \leq -2.84$ *which is hold for all* $x_2 \geq 0.75$ *This implies that* $m^k_2 \approx 0.75$*.*

As a result, when target 1 is the cut-off target, the minimum coverages of the defender at targets 1 and 2 are computed as:

$$x^{min}_1 = m^d_1 \approx 0.66$$
$$x^{min}_2 = \max\{m^k_2, \min\{m^d_2, m^a_2\}\} = 0.75$$

Case 2–Target 2 is the cut-off target. *The minimum coverages of the defender which satisfy that the defender's utility is at least* $t = 0.0$ *can be computed in a similar way.*

To that end, I will examine if at least one of the two cases will provide a feasible strategy for the defender; that is $\mathbf{x} \in \mathbf{X}$*. For example, in the case that target 1 is the cut-off target, I have* $x^{min}_1 + x^{min}_2 > 1$*. which means that the defender's utility* $t = 0.0$ *is infeasible.*

6.4 A Scalable Robust Algorithm II

I now turn to providing an approximate algorithm for the second group, which combines monotonic adversaries with other uncertainties. To that end, I first focus on approximate algorithm for the monotonic maximin problem without other uncertainties, i.e., $\max_{\mathbf{x}\in\mathbf{X}} \min_{\mathbf{y}\in L^m(\mathbf{x})} \sum_i y_i U^d(\mathbf{x}, i)$, by exploiting the structure of the feasible region which corresponds to $A_k(\mathbf{y})$, $L^m(\mathbf{x}) = \{\mathbf{y} \in \mathbf{Y} : U_i^a(\mathbf{x}) \geq U_j^a(\mathbf{x}) \implies y_i \geq y_j\}$.

The main computational difficulty of URAC for this case of uncertainty is due to its T^2 integer variables $z_k, k = (i, j), i, j = \overline{1, T}$. At a high level, my approach builds an alternative formulation with fewer integer variables. Given a defender strategy $\mathbf{x}$, the optimal solution of the inner minimization problem, $\mathbf{y}^*$, will be one of the extreme points of the polytope of $L^m(\mathbf{x})$ which means that $\forall i, j$ such that $y_i^*, y_j^* > 0$, then $y_i^* = y_j^*$. This implies that $L^m(\mathbf{x})$ has at most T extreme points (Jiang et al., 2013). In practice, I observe that the number of extreme points of $L^m(\mathbf{x}^*)$ where $\mathbf{x}^*$ is the optimal solution of the monotonic maximin problem is often much smaller than T. To exploit this observation, my idea is to find the optimal $\mathbf{x}^*$ within a subset $S_p^d \subseteq \mathbf{X}$ such that for each $\mathbf{x} \in S_p^d$, there are only p extreme points of $L^m(\mathbf{x})$ where $p \ll T$. Intuitively, having to consider fewer extreme points should make the computation simpler. Indeed, this optimization problem can be formulated as a MILP with only pT integer variables.

Specifically, I define S_p^d to be the set of $\mathbf{x}$ such that the targets can be clustered into p groups, each group having the same attacker expected utility. Formally, given $\mathbf{x} \in S_p^d$, define $G_1, G_2, ..., G_p$ as a partition of the T targets such that $\forall\ k = \overline{1, p}$, I have $U_i^a(\mathbf{x}) = U_j^a(\mathbf{x}), \forall i, j \in G_k$, and $\forall\ k < k'$, I have $U_i^a(\mathbf{x}) > U_j^a(\mathbf{x}), \forall i \in G_k, j \in G_{k'}$. Since the monotonic property implies that $U_i^a(\mathbf{x}) = U_j^a(\mathbf{x}) \implies y_i = y_j$, therefore $\forall i, j \in G_k$, I have $y_i = y_j$. I can then write the set of extreme points of $L(\mathbf{x})$ as $S_a(\mathbf{x}) = \{\mathbf{y}^k : y_i^k = \frac{1}{\sum_{r=1}^k |G_r|}, \forall i \in G_s, s \leq k;\ y_i^k = 0, \forall i \in G_s, s > k\}_{k=\overline{1,p}}$. Intuitively, each extreme point $\mathbf{y}^k$ corresponds to the case that the adversary only attacks targets belonging to group $G_1, G_2, \ldots, G_k$ with the same probability. As the optimal strategy of the adversary is an extreme point of $L(\mathbf{x})$, then it belongs to $S_a(\mathbf{x})$. In fact, $\mathbf{y}^* = \text{argmin}_{\mathbf{y}^k} \sum_i y_i^k U_i^d(\mathbf{x})$.

Denote by $B_p^d = \cup_{k=\overline{1,p}} S_k^d$ the set of the defender's strategy such that each $\mathbf{x} \in B_p^d$ will categorize the targets into no more than p groups. Finally, given that $\mathbf{x}^* \in B_p^d$, the monotonic

maximin problem becomes $\max_{\mathbf{x}\in B_p^d} \min_{\mathbf{y}\in S_a(\mathbf{x})} \sum_i y_i U_i^d(\mathbf{x})$, which can be encoded as the MILP formulated in (6.44-6.53), referred to as the Grouping Monotonic Maximin-p (GMM-p) where p indicates the maximum number of groups.

$$\max_{\mathbf{x},\mathbf{h},\mathbf{s},t,\mathbf{m}} \quad t \tag{6.44}$$

$$U_i^a(\mathbf{x}) + (1 - h_{k,i})N \geq m_k,\ \forall k, i \tag{6.45}$$

$$m_k \geq U_i^a(\mathbf{x}) - h_{k,i}N + \epsilon,\ \forall k, i \tag{6.46}$$

$$m_1 \geq U_i^a(\mathbf{x}),\ \forall i \tag{6.47}$$

$$m_k \geq U_i^a(\mathbf{x}) - h_{k-1,i}N,\ \forall k, i \tag{6.48}$$

$$h_{k,i} \geq h_{k-1,i},\ \forall k, i \tag{6.49}$$

$$s_{k,i} + t \leq U_i^d(\mathbf{x}) + (1 - h_{k,i})N,\ \forall k, i \tag{6.50}$$

$$-h_{k,i}N \leq s_{k,i} \leq h_{k,i}N,\ \sum\nolimits_i s_{k,i} = 0,\ \forall k, i \tag{6.51}$$

$$h_{p,i} = 1,\ \forall i, \sum\nolimits_i h_{1,i} \geq 1 \tag{6.52}$$

$$\sum\nolimits_i x_i \leq R, 0 \leq x_i \leq 1, h_{k,i} \in \{0, 1\},\ \forall k, i. \tag{6.53}$$

In this MILP, $\mathbf{h}_k$ ($k = \overline{1, p}$) is an integer vector which indicates targets belonging to individual groups. Specifically, if $h_{k,i} = 0$ and $h_{k+1,i} = 1$, target i must belong to group $k + 1$ and $h_{k',i} = 1,\ \forall k' \geq k + 1$ and $h_{k',i} = 0,\ \forall k' \leq k$. The variable $\mathbf{m}$ represents the adversary's expected utility for each group, i.e., all targets belonging to group k must have the adversary's expected utility equal to m_k. Variable t is the maximum utility of the defender and also the objective value for us to optimize. Finally, $\mathbf{s_k}$ is an auxiliary variable used for computing the defender's utility which is corresponding to a potential optimal strategy of the adversary.

Overall, constraints (6.45-6.49) guarantee that all targets in the same group will have the same adversary's expected utility and $\forall i \in G_k, j \in G_{k'}, k < k'$, I have: $U_i^a(\mathbf{x}) > U_j^a(\mathbf{x})$. Furthermore, constraints (6.50-6.51) guarantee that if t^* is the optimal objective value, then $t^* = \min_k\{\overline{U}^d(\mathbf{x}, k)\}$ where $\overline{U}^d(\mathbf{x}, k)$ is corresponding defender's utility to a potential optimal strategy $\mathbf{y}^k$ of the adversary in $S^a(\mathbf{x})$.

Theorem 2. *Denote by v_p^* the maximum utility of the defender returned by GMM-p. For all $p > p'$, I have: $v_p^* \geq v_{p'}^*$. Morever, there exists $1 \leq p \leq T$ such that $v_p^* = v^*$ where v^* is the maximum utility of the defender computed by monotonic maximin.*

Proof. Overall, the MILP (6.44-6.53) attempts to compute the optimal solution of the optimization problem: $\max_{\mathbf{x} \in B_p^d} \min_{\mathbf{y} \in S_a(\mathbf{x})} \sum_i y_i U_i^d(\mathbf{x})$.

In particular, constraints (6.45-6.49) guarantee that all targets in the same group will have the same adversary's expected utility and $\forall i \in G_k, j \in G_{k'}, k < k'$, I have: $U_i^a(\mathbf{x}) > U_j^a(\mathbf{x})$. In particular, if $h_{k-1,i} = 0$ and $h_{k,i} = 1$ which mean that target i belongs to group k, constraints (6.45) and (6.48) force $U_i^a(\mathbf{x}) = m_k$. If $h_{k,i} = 0$ which means that target i must belong to a group $k' > k$ which means $U_i^a(\mathbf{x}) < m_k$, constraint (6.46) guarantees that $U_i^a(\mathbf{x}) \leq m_k - \epsilon < m_k$ where ϵ is a given small positive number. Constraint (6.49) ensures that if $h_{k-1,i} = 1$, then $h_{k,i}$ must be equal to 1, being consistent with the definition of the integer variable vector $\mathbf{h_k}$. Constraint (6.52) guarantees that each target must belong to a certain group.

Furthermore, constraints (6.50-6.51) guarantee that if t^* is the optimal objective value, then $t^* = \min_{k=\overline{1,p}} \{\overline{U}^d(\mathbf{x}, k)\}$ where $\overline{U}^d(\mathbf{x}, k)$ is corresponding defender's utility to a potential optimal strategy $\mathbf{y}^k$ of the adversary in $S^a(\mathbf{x})$. In particular, in constraint (6.51), if $h_{k,i} = 0$, then $s_{k,i} = 0$. In addition, as $\sum_i s_{k,i} = 0$, I have: $\sum_{i \in \cup_{r=\overline{1,k}} G_r} s_{k,i} = 0$ (*). On the other hand, in constraint (6.50), $\forall i \in \cup_{r=\overline{1,k}} G_r$, which means that $h_{k,i} = 1$, I have: $s_{k,i} + t \leq U_i^d(\mathbf{x})$ (**). By taking the sum of (**) over all $i \in \cup_{r=\overline{1,k}} G_r$ and by using condition (*), I obtain the following derived inequality: for all $k = \overline{1,p}$, $t \leq \overline{U}^d(\mathbf{x}, k)$. As the objective of the MILP is to maximize t, the optimal values of $\{s_{k,i}\}_{k=\overline{1,p}, i=\overline{1,T}}$ will lead to $t^* = \min_{k=\overline{1,p}} \{\overline{U}^d(\mathbf{x}, k)\}$.

Therefore, GMM-p will find the optimal solution through B_p^d. In addition, I have the following property of B_p^d: $\forall T \geq p > p' \geq 1 : \ |B_p^d| \supseteq |B_{p'}^d|$. Therefore, $v_p^* \geq v_{p'}^*$. Finally, as any strategy of the defender will categorize the targets into p groups for some $p \in \{1, 2, ..., T\}$, the optimal strategy of the defender will belong to B_p for some p which implies that GMM-p will provide the optimal solution. □

In the case of **combination of monotonic adversary with other uncertainties**, the grouping idea can still be applied but a somewhat different approach is needed.

6.5 Experimental Results

I systematically generated payoff structures based on covariance games in GAMUT (Nudelman et al., 2004). I adjust the covariance value $r \in [-1.0, 0.0]$ with step size $\lambda = 0.1$ to control

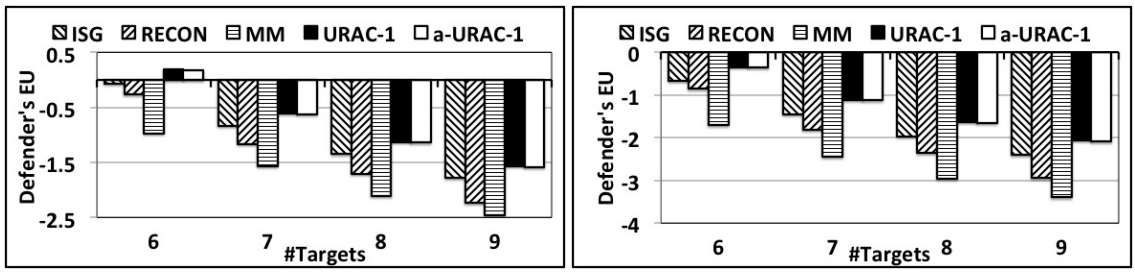
the correlation between rewards of players. The rewards and penalties of both the defender and the adversary are chosen within the ranges $[1, 10]$ and $[-10, -1]$ respectively. The experimental results are obtained using CPLEX on a 2.3 GHz machine with 4GB main memory. All comparison results except where noted with my algorithms are statistically significant under bootstrap-t ($\alpha = .05$) (Wilcox, 2002).

6.5.1 Solution quality

I show that my robust algorithms outperform other existing robust algorithms discussed in Chapter 3 in both small-scale and large-scale game scenarios, under conditions of low or high uncertainties, and given any combinations of uncertainties.

Small-scale domains: In my first set of experiments, I examine the performance of my algorithms in the case of small-scale games which are motivated by real-world domains such as LAX or Boston harbor (Tambe, 2011). I first examine the game settings in which uncertainties exist in all 3 dimensions of the uncertainty space (Figure 6.1): the adversary's payoff, the defender's strategy, and the adversary's rationality. Specifically, I examine two cases: 1) low uncertainty: $(\alpha = \beta = 0.1, \gamma = \eta = 0.01)$; and 2) high uncertainty: $(\alpha = \beta = 0.5, \gamma = \eta = 0.05)$. In both cases, the adversary responds monotonically. I evaluate the performance of URAC-1 and a-URAC-1 – versions addressing a combination of all uncertainties including monotonic adversary against ISG, RECON, and monotonic maximin (MM). For ISG and RECON, I search over the range of [0.1, 5.0] with step size $\lambda_1 = 0.2$ and the range of [0.01, 0.5] with step size $\lambda_2 = 0.02$ to find the parameter settings for $(\alpha, \beta, \gamma, \eta)$ that provide the highest defender's expected utility in my settings. In fact, when the sampled values of parameters are sufficiently large, i.e., $\alpha = \beta = 5.0$, ISG's optimal solution corresponds to Maximin's. In these figures, the results are averaged over 500 payoff structures.

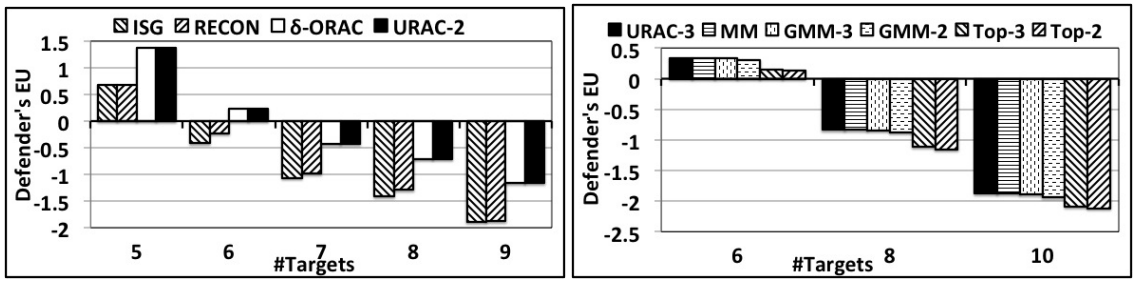
Figure 6.5 shows the defender's worst-case expected utility (y-axis) while varying the number of targets (x-axis). As shown in Figure 6.5, even though the parameters of both ISG and RECON are optimally tuned over the sampled values, both URAC-1 and a-URAC-1 still obtain significantly higher defender's expected utility. For example, in Figure 6.5(a), in the 6-target case, while ISG, RECON, and MM achieve a utility of -0.064, -0.2631, and -0.9672, respectively, the defender's utilities obtained by URAC-1 and a-URAC-1 are, in turn, 0.1892 and 0.1822.



(a) Uncertainty setting:$\alpha = \beta = .1, \gamma = \eta = .01$ (b) Uncertainty setting:$\alpha = \beta = .5, \gamma = \eta = .05$

Figure 6.5: Solution quality, all uncertainties

Now I switch to the case of a combination of a subset of or individual uncertainties. In this case, in addition to URAC, I evaluate the performance of my approximate algorithms. Figure 6.6(a) shows the solution quality of URAC-2 and $\delta-$ORAC where $\delta = 1e - 8$ in comparison with RECON and ISG in the case of combined uncertainties with *a rational adversary*. URAC-2 is a version of URAC to address this combination of uncertainties. The parameter values of both RECON and ISG are selected similarly to the previous experiment. In addition, I compare the solution quality of URAC-3, GMM-3, and GMM-2 with monotonic maximin (MM) and the top-K algorithms with K = 2, 3 when addressing the monotonic adversary without any additional uncertainty (Figure 6.6(b)). URAC-3 is a version of URAC corresponding to this case of uncertainty. Specifically, the top-K algorithms approximate the monotonic maximin solution; higher K achieves higher solution quality but runs more slowly. The Top-3 and Top-2 algorithms are chosen as they are the top performers (Jiang et al., 2013). In these figures, the results are averaged over 100 payoff structures.



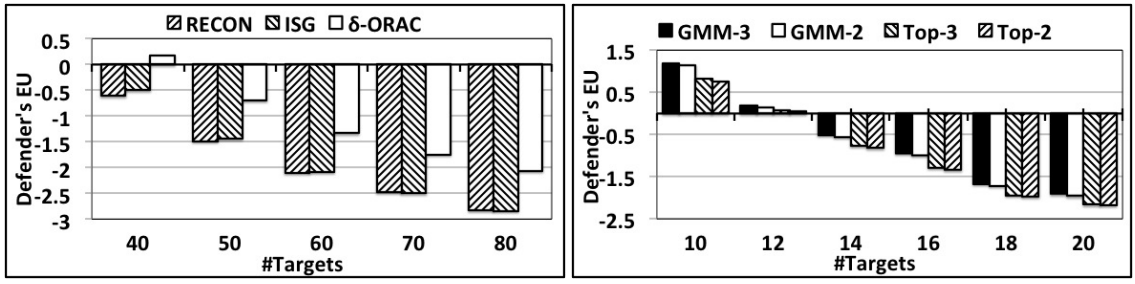
(a) Uncertainty setting:$\alpha = \beta = .5, \gamma = \eta = .05$ (b) Uncertainty setting: monotonic adversary

Figure 6.6: Solution quality, approximate algorithms

As shown in Figure 6.6, $\delta-$ORAC, GMM-2, and GMM-3 significantly outperform the other existing robust algorithms. Their solution quality is approximately the same as URAC-2 and

URAC-3, respectively. For example, in Figure 6.6(a), in the case of 9-target games, while the defender's utility obtained by both URAC-2 and $\delta-$ORAC is -1.16, ISG and RECON only achieve utilities of -1.89 and -1.87, respectively. These results show that my robust algorithms outperform other existing robust algorithms in terms of solution quality in small-scale games. The only exception is the isolated uncertainty of monotonic adversary where MM provides the exact optimal solution.

Large-scale domains: Here, I show that my approximate algorithms significantly outperform other robust algorithms for addressing a combination of a subset of or individual uncertainties. I examine 2 game settings: 1) combined uncertainties with a rational adversary (Figure 6.7(a)); and 2) monotonic adversary (Figure 6.7(b)). In these figures, the results are averaged over 100 payoff structures. Given the limited scalability of URAC to large games, I do not include its result. Figure 6.7 shows that my approximate algorithms obtain a significantly higher utility than other robust algorithms in large-scale games. For example, in Figure 6.7(a), in the case of 80-target games, $\delta-$ORAC achieves a utility of -2.06 while RECON and ISG obtain only -2.82 and -2.83, respectvely.

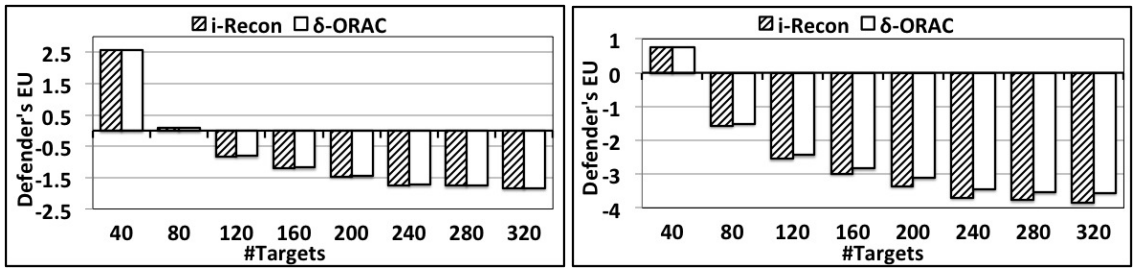


(a) Uncertainty setting:$\alpha = \beta = .5, \gamma = \eta = .05$ (b) Uncertainty setting: monotonic adversary

Figure 6.7: Solution quality, approximate algorithms

Furthermore, even when $\delta-$ORAC only attempts to address a specific type of uncertainty, i.e., uncertainty in the defender's strategy ($\alpha = \beta = 0.0$), it provides a higher solution quality than the fastest robust algorithm, i-RECON (Yin et al., 2011), for dealing with this type of uncertainty. In this experiment, $\delta-$ORAC guarantees to obtain an $\delta-$optimal solution with $\delta = 1e-8$ while i-RECON does not ensure any solution bound. As shown in Figure 6.8, while both algorithms achieve the similar expected utility when $\gamma = \eta = .01$, when the uncertainty increases, i.e., $\gamma = \eta = .05$, i-RECON obtains lower defender's utility than $\delta-$ORAC. For example, in Figure

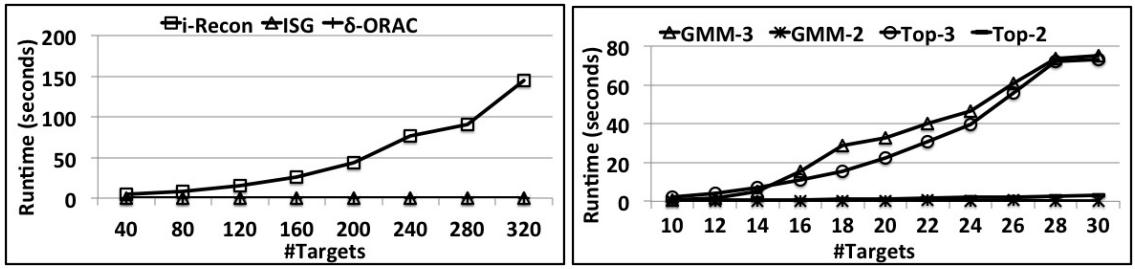
6.8(b), in the case of 320-target games, $\delta-$ORAC obtains a defender's utility of -3.54 on average while i-RECON achieves only -3.84. Overall, my robust algorithms significantly outperform the existing robust algorithms for addressing uncertainties.



(a) $\gamma = \eta = .01$ (b) $\gamma = \eta = .05$

Figure 6.8: Solution quality, uncertainty in defender's strategy

6.5.2 Runtime performance



(a) Uncertainty setting:$\alpha = \beta = .5, \gamma = \eta = .05$ (b) Uncertainty setting: monotonic adversary

Figure 6.9: Runtime performance, approximate algorithms

In addition to solution quality, I show that my approximate algorithms obtain an efficient runtime performance in comparison with other robust algorithms in large-scale games. The results are average over 100 payoff structures. In Figure 6.9, the y-axis indicates the runtime in seconds and the x-axis shows the number of targets. Figure 6.9(a) shows that $\delta-$ORAC runs significantly faster than i-RECON and its runtime is approximately the same as ISG; i-RECON's runtime grows quickly while $\delta-$ORAC's runtime is consistently fast as the number of targets increases. For example, i-RECON's runtime reaches 144.25 seconds while $\delta-$ORAC and ISG take only 0.05 and 0.046 seconds on average in 320-target games, respectively.

Furthermore, my GMM-p algorithm is shown to have approximately the same runtime as the Top-K algorithm (Figure 6.9(b)). While my approximate algorithms achieve higher quality without sacrificing runtime, some URAC versions are unable to scale-up. For example, when addressing a combination of all uncertainties (i.e., $\alpha = \beta = 0.5, \gamma = \eta = 0.05$, monotonic adversary), URAC-1's runtime is 85.17 seconds for 9-target games while my approximate algorithms take less than 1 second; of course, URAC-1 addresses a combination of uncertainties that no algorithm can.

6.6 Summary

In this chapter, I provide the following main contributions: 1) I present the first unified framework to handle all the uncertainties where robust maxmin algorithms have been defined in security games; 2) I provide a unified algorithmic framework from which I can derive different "unified" robust maximin algorithms to address combinations of these uncertainties; 3) I introduce approximate robust scalable algorithms; 4) I show through my experiments that my algorithms improve runtime performance and/or solution quality.

Chapter 7

Regret-based Solutions for Security Games

As shown in Chapter 6, a major approach to deal with uncertainties in SSGs is to use *strict uncertainty*, where the uncertain elements (e.g., the attacker payoffs) are assumed to lie within some interval. In this chapter, I develop novel robust optimization methods for addressing uncertainties in SSGs, specifically focusing on *Strict Uncertainty Payoff games (SPACs)* that rely on the *minimax regret* — a widely used decision criterion for decision making under uncertainty (Savage, 1972; Kouvelis & Yu, 1997; Boutilier et al., 2006). Essentially, minimax regret focuses on the loss with respect to *decision quality* over possible payoff realizations, making decisions with the tightest possible optimality guarantees.

Under the Bayesian perspective, it may be argued that minimax regret is too conservative. Minimax regret is also pointed out to not satisfy the principle of irrelevant alternatives or stochastic dominance (Savage, 1972; Luce & Raiffa, 2012; Quiggin, 1990). Yet, in many security domains, especially in wildlife protection, assessing a prior distribution over the payoff values (i.e., animal density) is often very difficult and hence the Bayesian approach is inappropiate. In addition, finding the equilibrium of a Bayesian Stackelberg game is NP-hard. On the other hand, in security domains, terrorist attacks could cause massive loss of life and thus security agencies may tend to be very cautious toward the worst case scenrio of uncertainty that affects their patrol effectiveness. Minimax regret helps in determining robust strategies and their vulnerabilities when a prior distribution is hard to estimate. Moreover, bounds on payoff values (or payoff uncertainty intervals) are easier to maintain. Using such bounds also leads to much less computationally expensive algorithms. Therefore, minimax regret is an appropriate robust method to deal with uncertainties in security games.

In fact, for these SPACs, robust optimization methods have been developed before using only the *maximin* decision criterion, in which a defender chooses a strategy that maximizes her worst-case utility over possible payoff realizations. Indeed, minimax regret has not yet been available to (security) policy makers—my work in this chapter makes it a viable criterion for generating new, less conservative, candidate defensive strategies. Unfortunately, operationalizing minimax regret involves complex, non-convex optimization for which efficient algorithms do not exist.

I thus develop novel, efficient algorithms for approximating minimax regret, and offer experimental results showing that high solution quality can be attained quickly. My second contribution is a *payoff elicitation* procedure that can be used to optimize the defender's efforts in assessing payoffs, allowing reduction in the uncertainty of those parameters that most improve decision quality. This is yet another reason to use minimax regret as an alternative robustness criterion—it has been proven to be a very effective driver of elicitation in several domains (Boutilier, Sandholm, & Shields, 2004; Regan & Boutilier, 2009). Finally, I propose and evaluate several payoff elicitation strategies, exploiting minimax regret solutions.

The remainder of the chapter is organized as follows: 1) First, I define minimax regret (MMR) and then introduce several new exact and approximate algorithms for its computation; 2) Second, I explain new preference elicitation methods; and 3) Finally, I provide empirical evaluation.

7.1 Regret-based Solutions

I now introduce some basic game concepts which will be used in this chapter. I then formulate the minimax regret solution for SPACs and finally, discuss several methods for its computation.

7.1.1 Basic Game Concepts

Expected utilities. In SSGs, a *defender* allocates m resources to protect a set of T *targets* from an *attacker* who will attack one of the targets. A defender *mixed* strategy is a vector $\mathbf{x} = (x)_{1 \leq t \leq T}$, with $0 \leq x_t \leq 1$ and $\sum_t x_t \leq m$, where x_t denotes the probability of protecting t. Let $\mathbf{X} = \{\mathbf{x} : 0 \leq x_t \leq 1, \sum_t x_t \leq m\}$ be the set of feasible defender strategies.

If the attacker attacks t, he receives a reward R_t^a if the target is unprotected, and a penalty $P_t^a < R_t^a$ if it is protected. Conversely, the defender receives a penalty P_t^d in the former case

and a reward $R_t^d > P_t^d$ in the latter. Given a defender strategy $\mathbf{x}$ and the attacked target t, the *expected utilities* of the adversary and the defender, respectively, are computed as follows:

$$U_t^a(x_t, R_t^a, P_t^a) = R_t^a(1 - x_t) + P_t^a x_t \tag{7.1}$$

$$U_t^d(x_t, R_t^d, P_t^d) = R_t^d x_t + P_t^d(1 - x_t) \tag{7.2}$$

Attack set. Let $R^a = \{R_t^a\}_t$ and $P^a = \{P_t^a\}_t$ be the the set of attacker rewards/penalties at all targets. The attacker's *attack set* $A(\mathbf{x}, R^a, P^a)$, given defender strategy $\mathbf{x}$, contains the targets that give him the highest utility:

$$A(\mathbf{x}, R^a, P^a) = \{t : U_t^a(x_t, R_t^a, P_t^a) \geq U_{t'}^a(x_{t'}, R_{t'}^a, P_{t'}^a), \forall t'\} \tag{7.3}$$

The defender's aim is to choose a strategy $\mathbf{x}$ that maximizes her own payoff $v(\mathbf{x}, R^a, P^a)$:

$$v(\mathbf{x}, R^a, P^a) = \max_{t \in A(\mathbf{x}, R^a, P^a)} U_t^d(x_t, R_t^d, P_t^d). \tag{7.4}$$

Key target. As in other work in the literature, I assume that the attacker breaks ties in favor of the defender (Von Stengel & Zamir, 2004). Given a defender's strategy $\mathbf{x}$ and an adversary's payoff (R^a, P^a), I call target t a *key target* if t is the attacked target within the attack set, that is:

$$t = \operatorname*{argmax}_{t' \in A(\mathbf{x}, R^a, P^a)} U_{t'}^d(x_{t'}, R_{t'}^d, P_{t'}^d)$$

$$v(\mathbf{x}, R^a, P^a) = U_t^d(x_t, R_t^d, P_t^d)$$

Attacker Payoff Uncertainty. I focus on SPACs (Kiekintveld et al., 2013; Yin et al., 2011) where the defender lacks the data to precisely estimate attacker payoffs. I assume that attacker payoffs are known only to lie within specific intervals: for each t, we have $R_t^a \in I_t^r = [R_{min}^a(t), R_{max}^a(t)]$ and $P_t^a \in I_t^p = [P_{min}^a(t), P_{max}^a(t)]$. Let $\mathbf{I} = \{(I_t^r, I_t^p)\}_t$ denote the set of payoff intervals.

7.1.2 Minimax Regret

Definition 3. *Given uncertainty interval* $\mathbf{I}$*, the* ***max regret*** *of defender strategy* $\mathbf{x} \in \mathbf{X}$ *is:*

$$MR(\mathbf{x}, \mathbf{I}) = \max_{(R^a, P^a) \in \mathbf{I}} \max_{\mathbf{x}' \in \mathbf{X}} \big(v(\mathbf{x}', R^a, P^a) - v(\mathbf{x}, R^a, P^a)\big) \tag{7.5}$$

Essentially, given payoff uncertainty $\mathbf{I}$, max regret evaluates the quality of strategy $\mathbf{x}$, under the worst-case realization of the payoff in $\mathbf{I}$, by measuring the worst-case loss in defender utility of using $\mathbf{x}$ rather than the optimal strategy $\mathbf{x}'$ given that realization.

Example 8. *Table 7.1 shows an example of a 3-target game with uncertainty in the attacker's payoffs. For example, the defender reward and penalty at target* 2 *is* 5 *and* -6 *respectively. On the other hand, the uncertainty intervals of the attacker's reward and penalty at all targets are the same, which in turn are* $[0, 10]$ *and* $[-4, 0]$*. I suppose that the defender has only one security resource and she plays a strategy of* $\{0.5, 0.3, 0.2\}$ *to protect the three targets.*

Targets	*Def. rew.*	*Def. pen.*	*Adv. rew.*	*Adv. pen.*
1	*-6*	*-7*	*[0, 10]*	*[-4,0]*
2	*5*	*-6*	*[0, 10]*	*[-4,0]*
3	*3*	*-5*	*[0, 10]*	*[-4,0]*

Table 7.1: A 3-target, 1-resource SPAC.

Then the payoff instance of the attacker within the uncertainty intervals which leads to a maximum regret for the defender is $\{0, 10, 0\}$ *of rewards and* $\{-4, 0, -4\}$ *of penalties. In this case, the corresponding optimal strategy of the defender w.r.t this payoff instance is* $\{0, 1, 0\}$*—the attacker will attack target* 2 *and as a result, the defender receives a utility of* 5 *to play this strategy. On the other hand, the defender will receive a utility of* -2.7 *for playing the strategy* $\{0.5, 0.3, 0.2\}$*. Thus, the defender receives a utility loss or regret of* $5 - (-2.7) = 7.7$*.*

To guard against this loss in utility, the defender can adopt the strategy that minimizes this max regret, which is defined as follows:

Definition 4. *The* ***minimax regret (MMR)*** *of interval* $\mathbf{I}$ *is:*

$$MMR(\mathbf{I}) = \min_{\mathbf{x} \in \mathbf{X}} MR(\mathbf{x}, \mathbf{I}). \tag{7.6}$$

A ***minimax optimal strategy*** *is any* $\mathbf{x}$ *that minimizes Eq. 7.6. Such a strategy has the strongest optimality guarantee given uncertainty* $\mathbf{I}$*.*

Example 9. *Given the example of a 3-target game in Table 7.1, I will show why MMR might be more valuable than maximin. The optimal defender strategy under maximin is* $[1.0, 0.0, 0.0]$*: it allocates all resources to the most vulnerable target* 1*, simply because it has the lowest reward/penalty, despite the fact that defending target* 1 *provides little benefit. Maximin also leaves*

targets 2 *and* 3 *unprotected, and has a max regret of* 11. *By contrast, MMR diversifies the defender strategy to minimize utility loss over all payoff realizations. The minimax optimal strategy is* $[0.34, 0.44, 0.22]$ *which has max regret* 6.2.

7.1.3 Computing Minimax Regret

I note that MMR can be formulated as follows:

$$\begin{aligned} &\min_{\mathbf{x}\in\mathbf{X}} \theta \\ &\text{s.t. } \theta \geq v(\mathbf{x}', R^a, P^a) - v(\mathbf{x}, R^a, P^a), \forall \mathbf{x}' \in \mathbf{X}, (R^a, P^a) \in \mathbf{I} \end{aligned} \tag{7.7}$$

Unfortunately, since $\mathbf{X}$ and $\mathbf{I}$ are continuous, the set of constraints is infinite; and the problem is non-convex, making MMR computation difficult. One practical approach to optimization with large constraint sets is *constraint sampling* (De Farias & Van Roy, 2004), coupled with *constraint generation* (Boutilier et al., 2006). The key idea is to sample a subset of constraints and gradually expand this set by adding violated constraints to the relaxed problem until convergence to the optimal solution. *MIRAGE (MInimax Regret Algorithm using constraint GEneration)*, see Alg. 2, begins by sampling pairs $(\mathbf{x}, (R^a, P^a))$ of defender strategies and attacker payoffs uniformly from $\mathbf{X}$ and $\mathbf{I}$ to obtain a finite set S of *sampled constraints*. It then solves the corresponding relaxed optimization program Eq. 7.7—using the *bCISM* algorithm—whose optimal solution $(lb, \mathbf{x}^*)$ provides a lower bound (lb) on true MMR. Then constraint generation is applied to determine violated constraints (if any). This uses the *ALARM* algorithm, which computes $MR(\mathbf{x}^*, \mathbf{I})$. This provides an upper bound (ub) on true MMR. If $ub > lb$, ALARM's solution provides us with the maximally violated constraint (see line 5), which is added to S. If $ub = lb$, then $\mathbf{x}^*$ is the minimax optimal strategy and $lb = ub = MMR(\mathbf{I})$.[1]

Example 10. *Figure 7.1 illustrates how the MIRAGE algorithm works for 2-target, 1-resource SPAC. Initially, MIRAGE considers only one payoff sample within the uncertainty intervals which is Payoff 1 in the figure. Then MIRAGE calls bCISM to generate an optimal regret-based strategy* $\mathbf{x^1}$ *for the defender against Payoff 1. Next, MIRAGE calls ALARM to find the new payoff instance Payoff 2 that provides the maximum regret for the defender when the defender plays strategy* $\mathbf{x^1}$.

[1] While constraint generation can be used by itself, generating violated constraints is computationally intensive. Initializing with some randomly sampled constraints offers better performance.

Algorithm 2: Constraint-generation (MIRAGE)

Initialize $S = \phi, ub = \infty, lb = 0$;
Randomly generate $(\mathbf{x}', R^a, P^a)$, $S = S \cup \{\mathbf{x}', (R^a, P^a)\}$;
while $ub > lb$ **do**
    Call bCISM to compute relaxed MMR w.r.t S. Let $\mathbf{x}^*$ be its optimal solution with objective value lb;
    Call ALARM to compute $MR(\mathbf{x}^*, \mathbf{I})$. Let $(\mathbf{x}'^*, R^{a,*}, P^{a,*})$ be its optimal solution with objective value ub;
    $S = S \cup \{\mathbf{x}'^{,*}, R^{a,*}, P^{a,*}\}$;
end
return $(lb, \mathbf{x}^*)$;

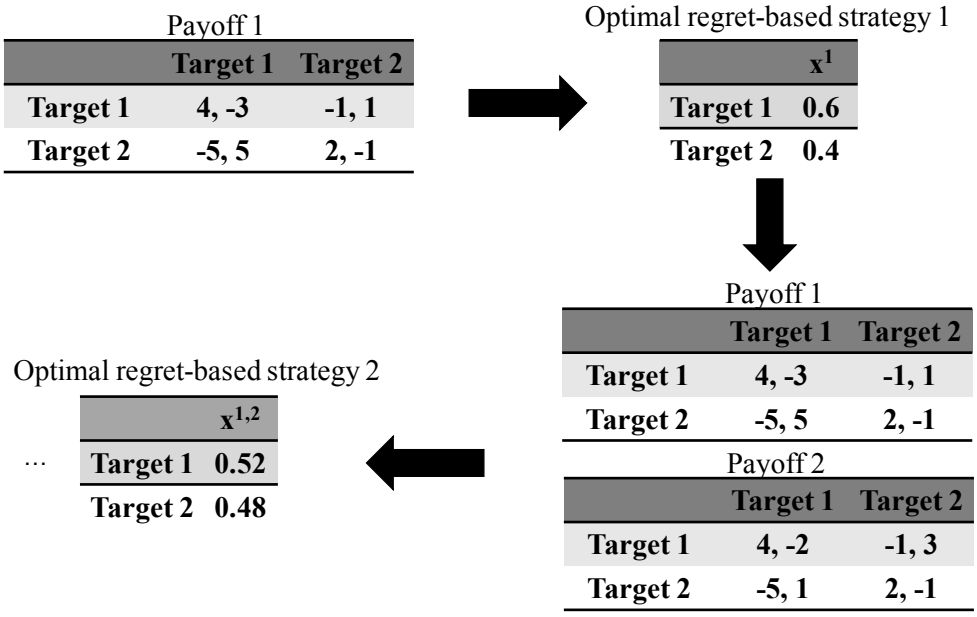


Figure 7.1: Illustration of the MIRAGE algorithm

Given the two payoff instances, in the next iteration, MIRAGE calls bCISM again to generate a new regret-based strategy for the defender, $\mathbf{x^2}$*, that minimizes the maximum regret against these two payoff instances. This process will continue until MIRAGE reaches the final optimal regret-based strategy for the defender.*

7.1.4 Compute Relaxed MMR

The first step, corresponding to line (4) of MIRAGE, solves a relaxed version of Eq. 7.7. This relaxed problem can be formulated as a mixed integer linear program (MILP) as follows:

$$\min_{\mathbf{x},\mathbf{q},\mathbf{r},\theta} \theta \tag{7.8}$$

$$s.t.\ \theta \geq v(\mathbf{x}'^{j}, R^{a,j}, P^{a,j}) - U_t^d(x_t, R_t^d, P_t^d) - (1-q_t^j)M,\ \forall j,t \tag{7.9}$$

$$U_t^a(x_t, R_t^{a,j}, P_t^{a,j}) + (1-q_t^j)M \geq r^j,\ \forall j,t \tag{7.10}$$

$$r^j \geq U_t^a(x_t, R_t^{a,j}, P_t^{a,j}),\ \forall j,t \tag{7.11}$$

$$\sum\nolimits_t x_t \leq m,\ x_t \in [0,1],\ \forall t \tag{7.12}$$

$$\sum\nolimits_t q_t^j = 1, q_t^j \in \{0,1\},\ \forall j,t \tag{7.13}$$

$$r^j \in \mathbf{R},\ \forall j. \tag{7.14}$$

Essentially, this MILP attempts to compute the optimal strategy for the defender that minimizes the maximum regret that the defender receives over all payoff instance within the sample set S. Here the index j ranges over sampled/generated constraints $(\mathbf{x}'^{j}, (R^{a,j}, P^{a,j}))$ in S, index t ranges over targets and M is a large positive constant. The binary variable q_t^j indicates whether t is the key target w.r.t the payoff instance j. In particular, constraints (7.10-7.11) ensure that if t is the key target (i.e., $q_j^t = 1$) for the j^{th} instance of $R^{a,j}$ and $P^{a,j}$, then the attacker's utility $U_t^a(x_t, R_t^{a,j}, P_t^{a,j}) \geq U_{t'}^a(x_{t'}, R_{t'}^{a,j}, P_{t'}^{a,j})\ \ \forall t'$. Finally, constraint (7.9) ensures that θ is the maximum regret for the defender by requiring that if $q_t^j{=}1$, then $\theta \geq v(\mathbf{x}'^{j}, R^{a,j}, P^{a,j}) - U_t^d(x_t, R_t^d, P_t^d)$. I dub this *mCISM (MILP for Computing dIScretized MMR)*. Unfortunately, mCISM becomes quite slow as S grows.

As an alternative, I introduce a *binary-search based algorithm (bCISM)* which searches defender utility space to find the optimal solution in polynomial time. Intuitively, I search for a strategy $\mathbf{x}$ that satisfies constraints (7.9-7.14), where θ is computed using binary search. Specifically, given θ, I compute the defender's *minimum coverage* at each target s.t. $\theta \geq v(\mathbf{x}'^{j}, R^{a,j}, P^{a,j}) - v(\mathbf{x}, R^{a,j}, P^{a,j})$ is satisfied for all $(\mathbf{x}'^{j}, (R^{a,j}, P^{a,j})) \in S$, and then test if $\mathbf{x} \in \mathbf{X}$ (i.e., is the strategy feasible).

Algorithm 3: Compute minimum coverage given θ

1 Initialize $lb_t = 0, x_t^j = +\infty \ \forall j, t$;
2 **while** *true* **do**
3 **for** $j = 1$ *to* $|S|$ **do**
4 **for** $t = 1$ *to* T **do**
5 $c_t^{j,t} = \max\{lb_t, \frac{\theta^j - P_t^d}{R_t^d - P_t^d}\}$;
6 **if** $c_t^{j,t} > 1$ **then** continue;
7 **foreach** $k \neq t$ **do**
8 $c_k^{j,t} = \max\{lb_t, \frac{U_t^a(c_t^j, R_t^{a,j}, P_t^{a,j}) - R_k^{a,j}}{P_k^{a,j} - R_k^{a,j}}\}$;
9 $x_k^j = \min\{x_k^j, c_k^{j,t}\}$;
10 **end**
11 $x_t^j = \min\{x_t^j, c_t^{j,t}\}$;
12 **end**
13 **end**
14 Set $\mathbf{x} = \{x_t : \ x_t = \max_j x_t^j\}_t$;
15 **if** $\mathbf{x} \notin \mathbf{X}$ **then** return false;
16 **else if** $v(\mathbf{x}, R^{a,j}, P^{a,j}) \geq \theta^j \ \forall j$ **then** return $\mathbf{x}$;
17 **else** $lb_t = x_t$;
18 **end**

Let $\theta^j = -\theta + v(\mathbf{x}'^j, R^{a,j}, P^{a,j})$ for the j^{th} constraint in S. I require $\theta^j \leq v(\mathbf{x}, R^{a,j}, P^{a,j})$ for all $j \in S$, with $\mathbf{x}$ is computed by Alg. 3. Alg. 3 iterates through two levels of problem decomposition to find $\mathbf{x}$ (as illustrated in Figure 7.2). First, it finds $c_k^{j,t}$, the minimum defender coverage at each target k for the j^{th} instance such that t is the key target and $\theta^j \leq v(\mathbf{x}, R^{a,j}, P^{a,j})$. I iterate over all possible key targets t to find corresponding $c_k^{j,t}$. Based on these $c_k^{j,t}$, I then compute the minimum defender coverage at target k, denoted x_k^j ($k \leq T$), which satisfies the constraint $\theta^j \leq v(\mathbf{x}^j, R^{a,j}, P^{a,j})$ w.r.t. the j^{th} instance only (lines (4-11)). The resulting $\{x_k^j\}_k$ for all $j \leq |S|$ are then combined to compute $\mathbf{x}$ (line 14), as I elaborate below.

I now explain lines (4-11). Note that if t is the key target of the j^{th} instance, then $v(\mathbf{x}^j, R^{a,j}, P^{a,j}) = U_t^d(x_t^j, R_t^d, P_t^d)$. Here the algorithm goes through all targets t that could potentially be the key target for the j^{th} instance to compute $c_k^{j,t}$. Specifically, if t is the key target, as $U_t^d(c_t^{j,t}, R_t^d, P_t^d) \geq \theta^j$, I have $c_t^{j,t} \geq \max\{lb_t, \frac{\theta^j - P_t^d}{R_t^d - P_t^d}\}$, where lb_t is the lower bound for the defender's coverage at t (line 5). This lower bound is updated in each iteration of the **while** loop. In addition, for any other target k, it follows that $U_k^a(c_k^{j,t}, R_k^{a,j}, P_k^{a,j}) \leq U_t^a(c_t^{j,t}, R_t^{a,j}, P_t^{a,j})$ which means $c_k^{j,t} \geq \max\{lb_k, \frac{U_t^a(c_t^{j,t}, R_t^{a,j}, P_t^{a,j}) - R_k^{a,j}}{P_k^{a,j} - R_k^{a,j}}\}$. Thus, the higher $c_t^{j,t}$,

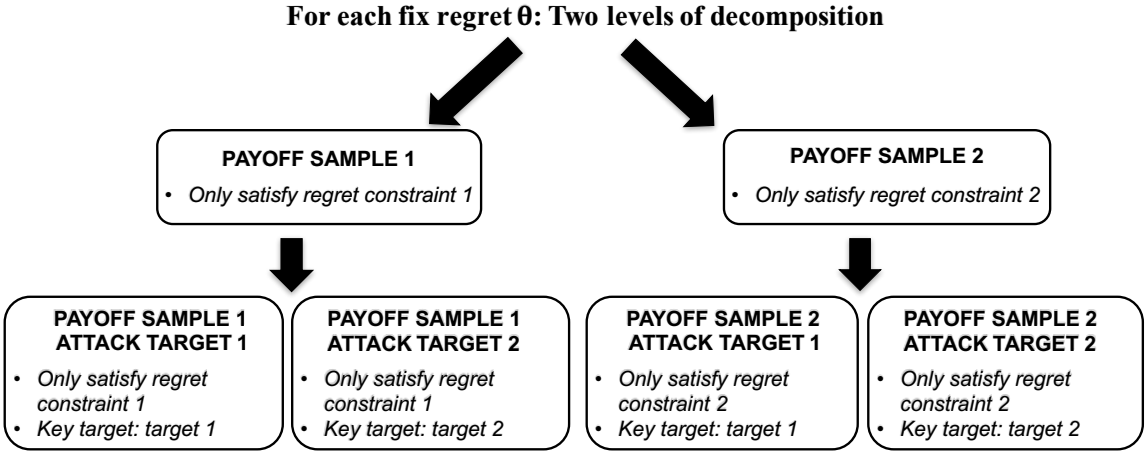


Figure 7.2: Two levels of decomposition in bCISM

the smaller $U_t^a(c_t^{j,t}, R_t^{a,j}, P_t^{a,j})$ and therefore, the higher $c_k^{j,t}$ for all k. The minimum coverage for target t is then $c_t^{j,t} = \max\{lb_t, \frac{\theta^j - P_t^d}{R_t^d - P_t^d}\}$ and for any other target k is $c_k^{j,t} = \max\{lb_k, \frac{U_t^a(c_t^{j,t}, R_t^{a,j}, P_t^{a,j}) - R_k^{a,j}}{P_k^{a,j} - R_k^{a,j}}\}$.

Proposition 3. *Given constraint j, suppose $\mathbf{x}$ is a feasible strategy s.t. $\theta^j \leq v(\mathbf{x}, R^{a,j}, P^{a,j})$ with key target t and lower bound $x_k \geq lb_k, \forall k$. Then $x_k \geq c_k^{j,t}, \forall k$.*

Proof. We always have the defender's coverage at target t satisfying $x_t \geq c_t^{j,t}$ as $\theta^j \leq v(\mathbf{x}, R^{a,j}, P^{a,j}) = U_t^d(x_t, R_t^d, P_t^d)$. Therefore, $U_t^a(x_t, R_t^{a,j}, P_t^{a,j}) \leq U_t^a(c_t^{j,t}, R_t^{a,j}, P_t^{a,j})$. As a result, since t is the key target, we have $U_k^a(x_k, R_k^{a,j}, P_k^{a,j}) \leq U_t^a(x_t, R_t^{a,j}, P_t^{a,j})$ for all k which means that $x_k \geq \max\{lb_k, \frac{U_t^a(x_t, R_t^{a,j}, P_t^{a,j}) - R_k^{a,j}}{P_k^{a,j} - R_k^{a,j}}\} \geq c_k^{j,t}$. □

Proposition 4. *For any constraint j, if $c_k^{j,t} > c_k^{j,t'}$ for some target k, then $c_i^{j,t} \geq c_i^{j,t'}$ for* all *targets i.*

Proof. Note that $c_i^{j,t} = \max\{lb_i, \frac{U_t^a(c_t^{j,t}, R_t^{a,j}, P_t^{a,j}) - R_i^{a,j}}{P_i^{a,j} - R_i^{a,j}}\}$ for all target i (including the key target t). Thus, $c_k^{j,t} > c_k^{j,t'}$ for some target k is equivalent to $U_t^a(c_t^{j,t}, R_t^{a,j}, P_t^{a,j}) < U_t^a(c_{t'}^{j,t'}, R_{t'}^{a,j}, P_{t'}^{a,j})$. As a result, I obtain $c_i^{j,t} \geq c_i^{j,t'}$ for all target i. □

Prop. 3 implies that if $x_k < c_k^{j,t}$ for some k, $\mathbf{x}$ is not feasible given that t is the key target and $\theta^j \leq v(\mathbf{x}, R^{a,j}, P^{a,j})$. In other words, $c_k^{j,t}$ is a lower bound on all x_k satisfying this condition. Prop. 4 implies that for any pair of key targets t and t', $c_k^{j,t} \geq c_k^{j,t'}$ for all targets k, or vice versa. Taken together, the minimum defender coverages are $x_k^j = \min_t c_k^{j,t}, \ \forall k$. As strategy $\mathbf{x}$ must satisfy $\theta^j \leq v(\mathbf{x}, R^{a,j}, P^{a,j})$ for all $j \in S$, $x_t \geq x_t^j, \forall t$. Therefore, $x_t \geq \max_j x_t^j, \forall t$ (line 14).

Since this minimum coverage x_t may be infeasible (i.e., $\mathbf{x} \notin \mathbf{X}$), I test feasibility (line (15)). Furthermore, it may not be feasible w.r.t. θ as it may now violate the constraints for the chosen key targets in S. Thus, I check (line (16)) for violations that give an objective value less than θ. If so, I update the lower bound on x_t (line (17)). I repeat until the constraint $\mathbf{x} \in \mathbf{X}$ is violated or a feasible solution is found. bCISM thus determines if some feasible $\mathbf{x}$ satisfies $\theta \geq v(\mathbf{x}', R^a, P^a) - v(\mathbf{x}, R^a, P^a)$ for all $(\mathbf{x}', (R^a, P^a)) \in S$. bCISM is guaranteed to provide a δ-optimal solution to MMR, where δ is the binary-search termination threshold.

Example 11. *Figure 7.3 shows an example of a 2-target game with two instances of the attacker's payoffs. Assume that the defender has one resource, the optimal strategies for the defender against these two payoff instances are:* $\{\frac{3}{8}, \frac{5}{8}\}$ *and* $\{\frac{1}{2}, \frac{1}{2}\}$ *respectively. The corresponding optimal utilities for the defender is* 2.25 *and* 1.0 *w.r.t the payoff instances* 1 *and* 2 *respectively.*

Targets	*Att. rew.*	*Att. pen.*
1	*2*	*-2*
2	*3*	*-1*

(a) Attack payoff instance 1

Targets	*Att. rew.*	*Att. pen.*
1	*1*	*-3*
2	*2*	*-4*

(b) Attacker payoff instance 2

Targets	*Def. rew.*	*Def. pen.*
1	*3*	*-1*
2	*6*	*-4*

(c) Defender's payoffs

Figure 7.3: A 2-target, 1-resource game with 2 attacker payoff instances.

Let's consider a value of $\theta = 1$. *Since, we aim at determining whether the defender's maximum regret is at most* $\theta = 1$*, then the utilties of the defender w.r.t these two payoff instances must be no less than* $\theta^1 = 1.25$ *and* $\theta^2 = 0.0$ *respectively. I analyze these two conditions separately:*
W.r.t payoff instance 1. *Considering two cases of key target:*

- *If target* 1 *is the key target, this means that the attacker will attack target* 1*. In other words, the defender receives the expected utility at target* 1 *and the attacker's expected utility at*

target 1 *is no less than at target* 2*. Since the defender's utility is no less than* $\theta^1 = 1.25$*, we obtain the minimum coverages* $\{c_1^{1,1}, c_2^{1,1}\}$*:*

$$3 \times c_1^{1,1} + (-1) \times (1 - c_1^{1,1}) \geq 1.25 \implies c_1^{1,1} = \frac{5}{8}$$
$$2 \times (1 - c_1^{1,1}) + (-2) \times c_1^{1,1} \geq 3 \times (1 - c_2^{1,1}) + (-1) \times c_2^{1,1} \implies c_2^{1,1} = \frac{7}{8}$$

- *If target* 2 *is the key target, this mean that the attacker will attack target* 2*. Similarly, I obtain the minimum coverages* $\{c_1^{1,2}, c_2^{1,2}\}$*:*

$$6 \times c_2^{1,1} + (-4) \times (1 - c_2^{1,1}) \geq 1.25 \implies c_2^{1,1} = 0.525$$
$$2 \times (1 - c_1^{1,1}) + (-2) \times c_1^{1,1} \leq 3 \times (1 - c_2^{1,1}) + (-1) \times c_2^{1,1} \implies c_1^{1,1} = 0.275$$

By combining these two cases of attack target, I obtain the minimum coverages of the defender, $\{x_1^1, x_2^1\}$*, which satisfies that the defender's utility is no less than* $\theta^1 = 1.25$*:* $x_1^1 = \min\{c_1^{1,1}, c_1^{1,2}\} = 0.525$ *and* $x_2^1 = \min\{c_2^{1,1}, c_2^{1,2}\} = 0.275$*.*

W.r.t payoff instance 2. *Similarly, I obtain the minimum coverages of the defender,* $\{x_1^2, x_2^2\}$*, which satisfies that the defender's utility is no less than* $\theta^2 = 0.0$*:* $x_1^2 = 0.25$ *and* $x_2^2 = \frac{1}{3}$*.*

Finally, since the defender's strategy must satisfy both utillity conditions w.r.t the two attacker payoff instance, the defender's coverages must satisify the following lower bound conditions: $x_1 \geq \max\{x_1^1, x_1^2\} = 0.525$ *and* $x_2 \geq \max\{x_2^1, x_2^2\} = \frac{1}{3}$*. To that end, I verify if these lower bounds will satisfy both utility conditions (which means that the defender can obtain the regret which is less than* $\theta = 1$*). Otherwise, I will continue the above steps but now with additional lower bound constraints that the defender's coverages must be no less than* $\{0.525, \frac{1}{3}\}$*.*

7.1.5 Computing Max Regret

The second sub-problem of MIRAGE is computation of $MR(\mathbf{x}, \mathbf{I})$. This is accomplished with *ALARM (Approximate Linearization Algorithm for Reckoning Max regret)*. Given Eq. 7.5, $MR(\mathbf{x}, \mathbf{I})$ requires computing the strategy $\mathbf{x}'$ and attacker payoff (R^a, P^a) that maximize the loss of $\mathbf{x}$. However, value of $v(\mathbf{x}', R^a, P^a) = \max_{t \in A(\mathbf{x}', R^a, P^a)} U_t^d(x_t', R_t^d, P_t^d)$ depends on the attack set $A(\mathbf{x}', R^a, P^a)$, which in turn is determined by the attacker's utility at each target t—$U_t^a(x_t', R_t^a, P_t^a) = R_t^a(1 - x_t') + P_t^a x_t'$. This is a non-convex function of variables x_t' and (R_t^a, P_t^a),

making max regret a non-convex optimization problem. To speed up computation, I now develop a linearization.

Both $v(\mathbf{x}', R^a, P^a)$ and $v(\mathbf{x}, R^a, P^a)$ are dictated by the key targets, which depend on $(R^a, P^a) \in \mathbf{I}$ and $\mathbf{x}' \in \mathbf{X}$. I partition $(\mathbf{I}, \mathbf{X})$ into T^2 subsets such that each pair of targets t, t' are the key targets within a particular subset. I then search over all pairs of possible key targets (t', t) to compute max regret. Specifically, given key targets (t', t), max regret can be reformulated as follows:

$$\max_{(R^a,P^a)\in\mathbf{I},\mathbf{x}'\in\mathbf{X}} U^d_{t'}(x'_{t'}, R^d_{t'}, P^d_{t'}) - U^d_t(x_t, R^d_t, P^d_t) \tag{7.15}$$

$$\text{s.t. } U^a_{t'}(x'_{t'}, R^a_{t'}, P^a_{t'}) \geq U^a_k(x'_k, R^a_k, P^a_k), \forall k \neq t, t' \tag{7.16}$$

$$U^a_t(x_t, R^a_t, P^a_t) \geq U^a_k(x_k, R^a_k, P^a_k), k \in N(t) \setminus \{t, t'\} \tag{7.17}$$

$$U^a_t(x_t, R^a_t, P^a_t) \geq U^a_k(x_k, R^a_k, P^a_k) + \epsilon, k \notin N(t) \cup \{t, t'\} \tag{7.18}$$

$$U^a_{t'}(x'_{t'}, R^a_{t'}, P^a_{t'}) \geq U^a_t(x'_t, R^a_t, P^a_t) \tag{7.19}$$

$$U^a_t(x_t, R^a_t, P^a_t) \geq U^a_{t'}(x_{t'}, R^a_{t'}, P^a_{t'}) \text{ if } t' \in N(t) \tag{7.20}$$

$$U^a_t(x_t, R^a_t, P^a_t) \geq U^a_{t'}(x_{t'}, R^a_{t'}, P^a_{t'}) + \epsilon \text{ if } t' \notin N(t) \tag{7.21}$$

where $N(t) = \{k : U^d_t(x_t, R^d_t, P^d_t) \geq U^d_k(x_k, R^d_k, P^d_k)\}$ is the set of targets at which the defender utility is lower than the utility at t, and ϵ is a small positive constant. I separate constraints for key and non-key targets for expository purposes. Constraints (7.17, 7.18, 7.20, 7.21) ensure that t is the key target w.r.t. $\mathbf{x}$, while ϵ ensures the strict inequality $U^a_t(x_t, R^a_t, P^a_t) > U^a_k(x_k, R^a_k, P^a_k)$ when $U^d_t(x_t, R^d_t, P^d_t) < U^d_k(x_k, R^d_k, P^d_k)$ (or when $k \notin N(t)$); otherwise, the attacker will attack k instead of t (by tie-breaking). I do allow violation of the tie-breaking constraint for the key target t' w.r.t. $\mathbf{x}'$ for each sub-problem (7.15-7.21). Searching over all possible key targets t' guarantees the final solution will satisfy the tie-breaking assumption.

Note that (R^a_k, P^a_k) and $\mathbf{x}'$ are involved in computing $U^a_k(x'_k, R^a_k, P^a_k)$ which makes this utility function non-convex (constraints (7.16–7.19)). I could solve the MR problem (Eq. 7.15) using existing commercial solvers for non-convex optimization (e.g., Knitro): (a) solve a non-convex program for every combination of key targets (Multi-NLP); or (b) formulate it as a mixed integer non-linear program (MINLP, see Appendix). However, these perform poorly, as I show below.

Hence I use binary search to *linearize* these non-convex constraints to approximate MR (noting that they can be linearized if $x'_{t'}$ and x'_t are known).

Problem 1. ***(ALARM binary search decision problem)*** *Given a value θ, and t', t as key targets: Are there $(R^a, P^a) \in \mathbf{I}$ and $\mathbf{x}' \in \mathbf{X}$ satisfying (7.16–7.21), such that $U^d_{t'}(x'_{t'}, R^d_{t'}, P^d_{t'}) - U^d_t(x_t, R^d_t, P^d_t) \geq \theta$?*

Proposition 5. *If Problem 1 has a feasible solution, then the following solution is feasible: $R^a_k = R^a_{min}(k)$ and $P^a_k = P^a_{min}(k)$, $\forall k \neq t, t'$; and $x'_{t'} = x'_{min}(t') = \max\{0, \frac{\theta + U^d_t(x_t, R^d_t, P^d_t) - P^d_{t'}}{R^d_{t'} - P^d_{t'}}\}$.*

Proof. I observe that (R^a_k, P^a_k) with $k \neq t, t'$ only involve in the RHS of constraints (7.16-7.18). Therefore, if there exists a feasible solution of the decision problem, then for all $k \neq t, t'$, $(R^a_{min}(k), P^a_{min}(k))$ is also a feasible value of (R^a_k, P^a_k) since $U^a_k(x'_k, R^a_k, P^a_k)$ and $U^a_k(x_k, R^a_k, P^a_k)$ (the RHS of the constraints) are minimized when $R^a_k = R^a_{min}(k)$ and $P^a_k = P^a_{min}(k)$ for all $k \neq t, t'$. As a result, constraints (7.16-7.18) are satisfied given $R^a_k = R^a_{min}(k)$ and $P^a_k = P^a_{min}(k)$ for all $k \neq t, t'$.

Furthermore, it follows that $U^d_{t'}(x'_{t'}, R^d_{t'}, P^d_{t'}) - U^d_t(x_t, R^d_t, P^d_t) \geq \theta \iff x'_{t'} \geq x'_{min}(t')$ where $x'_{min}(t') = \max\{0, \frac{\theta + U^d_t(x_t, R^d_t, P^d_t) - P^d_{t'}}{R^d_{t'} - P^d_{t'}}\}$. Therefore, I have $U^a_{t'}(x'_{t'}, R^a_{t'}, P^a_{t'}) \leq U^a_{t'}(x'_{min}(t'), R^a_{t'}, P^a_{t'})$ for all $\mathbf{x}'$. As a result, if constraints (7.16, 7.19) are satisfied with any $x'_{t'}$, then the constraints are also satisfied with $x'_{t'} = x'_{min}(t')$. □

Thus, by replacing (R^a_k, P^a_k) with $(R^a_{min}(k), P^a_{min}(k))$, for all $k \neq t, t'$ and $x'_{t'}$ with $x'_{min}(t')$, We are left with one non-convex constraint (7.19). I circumvent this non-convex constraint by converting Problem 1 into the following optimization with a non-convex objective:

$$\max_{\{x'_k\}_{k \neq t'}, R^a_t, P^a_t, R^a_{t'}, P^a_{t'}} U^a_{t'}(x'_{t'}, R^a_{t'}, P^a_{t'}) - U^a_t(x'_t, R^a_t, P^a_t) \tag{7.22}$$

$$\text{s.t. updated (7.16–7.18, 7.20–7.21)} \tag{7.23}$$

Intuitively, constraint (7.19) is translated into the objective (7.22), maintaining other constraints with their updates $(R^a_{min}(k), P^a_{min}(k))$ and $x'_{min}(t')$.

Proposition 6. *If the optimum of (7.22) is no less than zero, Problem 1 has a feasible solution; otherwise it is infeasible.*

Algorithm 4: Local search

Initialize x'_t, $\delta = \inf$;
Compute $x'_{k\neq t,t'}$ which minimize $\max_{k\neq t,t'} U^a_k(x'_k, R^a_k, P^a_k)$ using ORIGAMI;
while $\delta > 0$ **do**
    Given $x'_{k\neq t'}$, solve (7.22-7.23) to obtain obj^* and $(R^a_t, P^a_t, R^a_{t'}, P^a_{t'})^*$;
    Given $(R^a_t, P^a_t, R^a_{t'}, P^a_{t'})^*$, solve (7.22-7.23) to obtain obj^{**} and $x'_{k\neq t'}$;
    $\delta = obj^{**} - obj^*$;
end
return $(obj^{**}, (R^a_t, P^a_t, R^a_{t'}, P^a_{t'})^*, x'_{k\neq t'}$;

Proof. Because the optimal solution of (7.22-7.23) is no less than zero, it satisfies constraint (7.19). As this optimal solution also satisfies constraints (7.16-7.18, 7.20-7.21), it thus is a feasible solution of the decision problem 5. □

As the objective (7.22) remains non-convex, and direct non-convex methods are inefficient, I apply the following linearization:

Proposition 7. *If x'_t is fixed, then the $\{x'_k\}_{k\neq t,t'}$ which minimizes $\max_{k\neq t,t'} U^a_k(x'_k, R^a_k, P^a_k)$ are optimal for (7.22).*

Proof. I observe that x'_k for all $k \neq t', t$ only involve in the RHS of constraint (7.16). Denote by $\{x'^*_k\}_{k\neq t',t} = \underset{\{x'_k\}_{k\neq t',t}}{\operatorname{argmin}} \max_{k\neq t,t'} U^a_k(x'_k, R^a_k, P^a_k)$. As x'_t is fixed, for all feasible solutions $\{x'_k\}_{k\neq t',t}, R^a_t, P^a_t, R^a_{t'}, P^a_{t'}$ of (7.22-7.23), I also have $\{x'^*_k\}_{k\neq t',t}, R^a_t, P^a_t, R^a_{t'}, P^a_{t'}$ is a feasible solution. Because the objective function depends on only $R^a_t, P^a_t, R^a_{t'}, P^a_{t'}$, thus $\{x'^*_k\}_{k\neq t',t}$ is optimal. □

Recall that my variables are $R^a_t, P^a_t, R^a_{t'}, P^a_{t'}$ and x'_k $(k \neq t')$. Given a value of x'_t, by Prop. 7 I can apply ORIGAMI (Kiekintveld et al., 2009) to compute x'_k for all $k \neq t, t'$ to minimize $\max_{k\neq t,t'} U^a_k(x'_k, R^a_k, P^a_k)$. The remaining variables are $R^a_t, P^a_t, R^a_{t'}, P^a_{t'}$. Starting with an initial value x'_t and the corresponding x'_k for all $k \neq t, t'$ computed by ORIGAMI, my local search, see Alg. 4, gradually updates $(R^a_t, P^a_t, R^a_{t'}, P^a_{t'})$ and $x'_{k\neq t'}$ to find a local optimum of Eq. (7.22). In Alg. 4, obj^* and obj^{**} are the optimal objective values of Eq. (7.22) given $x'_{k\neq t'}$ and $(R^a_t, P^a_t, R^a_{t'}, P^a_{t'})^*$, respectively. By fixing either $x'_{k\neq t'}$ or $(R^a_t, P^a_t, R^a_{t'}, P^a_{t'})$, Eq. (7.22) becomes a linear program, allowing one to readily obtain a locally optimal solution. my experiments show that randomly initializing x'_t even just five times typically suffices to find a *global* optimum.

7.2 Payoff Elicitation

Typically, defenders employ the services of expert risk analysts to assess the attacker payoffs at specific targets (Shieh et al., 2012). While my MMR methods offer robust decisions in the face of payoff uncertainty, the resulting max regret level may be too large in certain cases. Thus I develop an interactive process whereby the defender can reduce her uncertainty w.r.t. attacker payoffs by querying the expert, at some cost, for additional information about these payoffs. Note however that reducing uncertainty for its own sake often fails to improve max regret (Boutilier et al., 2006; Braziunas & Boutilier, 2010); attention must be focused on those payoffs that actually influence *decisions*. I develop preference elicitation strategies driven by MMR that focus on "relevant" uncertainty. Generally, queries and payoff assessment continue until MMR reaches an acceptable level or some budget limit is met.

I consider *bound queries*, widely used in preference elicitation (Boutilier et al., 2006; French, 1986), in which an expert is asked whether the attacker reward/penalty at some target t is greater than some value p. For example, if the reward interval at target t is $[0, 10]$, I can ask if the reward is greater than 5. A positive (negative) response increases the lower bound to 5 (decreases the upper bound to 5). I assume each query q_t is associated with a single target t, and queries both the reward R_t^a and penalty P_t^a terms at the midpoints of their corresponding intervals I_t^r and I_t^p (since assessment of a target generally provides information about both). Thus each query has fmy possible responses. Query costs may vary with the target.

Since reducing uncertainty at different targets impacts max regret differently, and I may have a fixed budget, queries must be chosen effectively. Alg. 5 outlines my elicitation procedure. I now describe three heuristic *query selection strategies*.

Myopic strategy. I compute the average regret reduction over the fmy possible responses to each query (target) and query the target with the greatest average reduction.

Approximate strategy. The Myopic strategy may be computationally inefficient as it relies on exact computation of MMR for all $4T$ query responses. This strategy uses bCISM to approximate MMR, but otherwise mimics Myopic.

Optimistic/pessimistic strategy. To maximize regret, attacker payoffs are set within uncertainty intervals to induce the attacker to select targets that cause greatest defender loss: let the MMR solution set attacker payoffs to be $(\overline{R}^a, \overline{P}^a)$. If a query at target t obtains a response that makes the

Algorithm 5: Preference elicitation

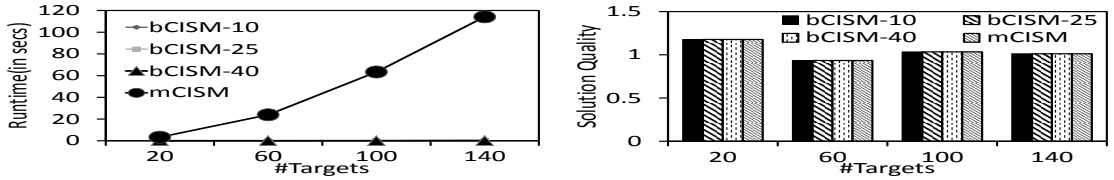
```
Initialize totalcost = 0, regret = ∞;
while totalcost < budget or regret > thres do
    Find the best target t* using an elicitation strategy;
    Ask a bound query regarding payoff of target t*;
    Update uncertainty interval I_new;
    Update totalcost, recompute MMR;
end
```

attacker's (say) reward $\overline{R}_t^a$ infeasible (by responding with the half interval of I_t^T *not* containing $\overline{R}_t^a$) max regret may be reduced significantly. my optimistic heuristic evaluates queries by assuming optimistically that each query response will rule out the current "regret-inducing" payoffs at that target, and selects the query that, assuming this response, offers the greatest MMR reduction. my pessimistic strategy is analogous. I combine the Optimistic and Pessimistic strategies with the Approximate strategy by replacing the exact MMR computation with the approximation bCISM, thereby increasing their computational efficiency. I call these strategies **Opt-Approx** and **Pess-Approx**.

7.3 Experimental Evaluation

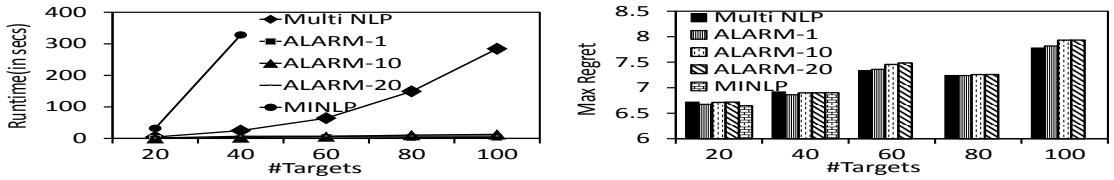
I evaluate the runtime and solution quality of my algorithms on games generated using GAMUT.[2] All experiments Ire run on a 2.83GHz Intel processor with 4GB of RAM, using CPLEX 12.3 for LP/MILPs and KNITRO 8.0.0.z for nonlinear optimization. I set the covariance value $r \in [0.0, 1.0]$ with step size $\lambda = 0.2$ to control the correlation of attacker and defender rewards. Upper and lower bounds for payoff intervals are generated randomly from $[-14, -1]$ for penalties and $[1, 14]$ for rewards, with the difference betIen the upper and lower bound (i.e., interval size) exactly 2 (this gives payoff uncertainty of roughly 30%). All results are averaged over 120 instances (20 games per covariance value) and use eight defender resources unless otherwise specified. All comparison results with my algorithms are statistically significant under bootstrap-t ($\alpha = 0.05$).

[2] See http://gamut.stanford.edu/.



(a) Runtime (b) Solution quality

Figure 7.4: Evaluating discretized MMR algorithms



(a) Runtime (b) Solution quality

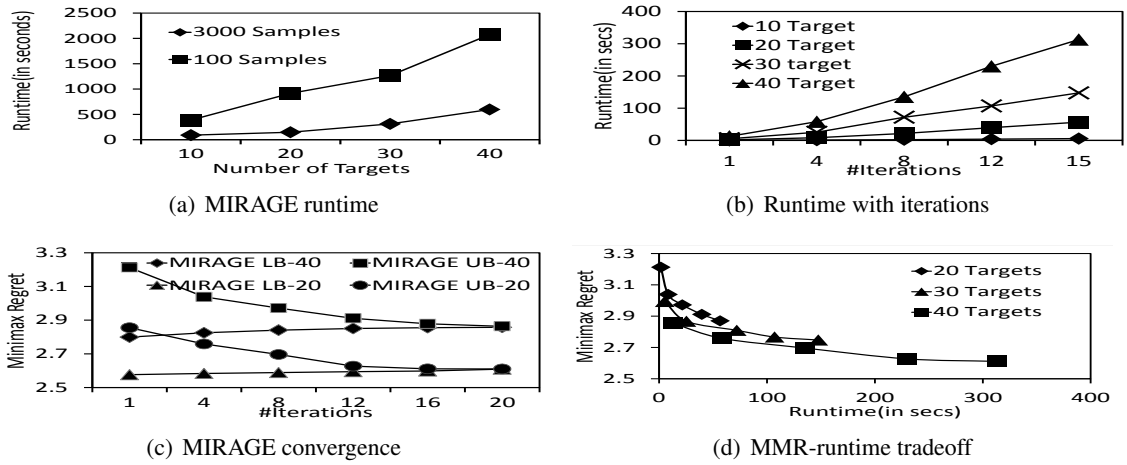
Figure 7.5: Evaluating algorithms for computing MR

7.3.1 Evaluating MMR Algorithms

I first evaluate my algorithms for discretized MMR. Fig. 7.4(a) shows runtimes of bCISM with 10, 25 and 40 initial samples from S, and mCISM with 10 samples. bCISM is roughly 68 times faster than mCISM (60 targets), and is rather insensitive to the number of initial samples. Fig. 7.4(b) plots solution quality; bCISM's solutions are within 0.01% of the mCISM's optima. Thus, the computational efficiency of bCISM comes at low cost in terms of solution quality.

I next evaluate my three MR algorithms, ALARM (with 1, 10 and 20 samples of x'_t), MINLP, and Multi-NLP. Fig. 7.5(a) shows that MINLP and Multi-NLP's runtimes increase exponentially with the number of targets. All instances of ALARM run approximately 100 (resp., 30) times faster than MINLP (resp., Multi-NLP) with 100 targets. Fig. 7.5(b) plots their relative solution quality and shows that ALARM, despite its exponential speedup, has at most a 1% (avg.) loss in solution quality vs. MINLP and Multi-NLP.

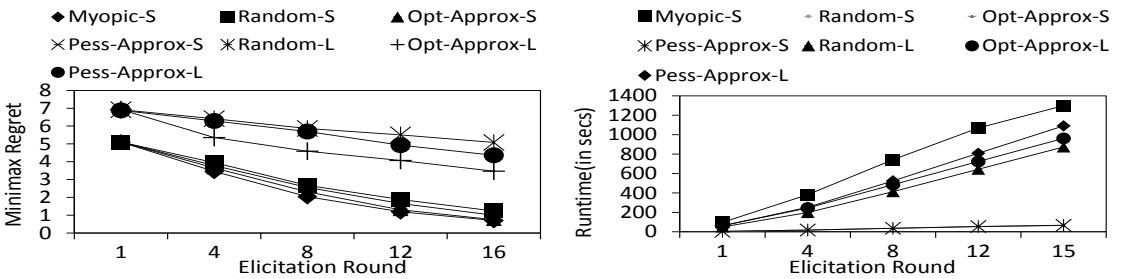
Since bCISM and ALARM handily outperform the other algorithms, I now evaluate MIRAGE only using bCISM and ALARM as its relaxed-MMR and MR subroutines. Fig. 7.6(a) plots MIRAGE runtimes, as number of targets varies, with either 100 and 3000 samples (constraints) added at each iteration. With 50 targets (not shown), MIRAGE takes about 10mins., and converges after 15 iterations. With 100 samples, MIRAGE takes longer to converge (about 6 times slower than with 3000); and runtime increases roughly by a factor of 14 with every 5 targets. Fig. 7.6(b) shows MIRAGE's runtime with the number of iterations (10 to 40 targets),



(a) MIRAGE runtime (b) Runtime with iterations

(c) MIRAGE convergence (d) MMR-runtime tradeoff

Figure 7.6: Evaluating MIRAGE properties



(a) Solution quality (b) Runtime

Figure 7.7: Evaluating payoff elicitation strategies

while Fig. 7.6(c) shows progress of its MMR upper and lower bounds (20 and 40 targets). This demonstrates a promising anytime profile, allowing early termination with high quality solutions. For instance, Fig. 7.6(c) shows, with 20 targets, that 9 iterations offers a solution within 5% of minimax optimality. Fig. 7.6(d) shows the tradeoff betIen runtime and (upper bound on) MMR (20 and 40 targets) up to convergence, confirming the positive anytime profile.

7.3.2 Evaluating Payoff Elicitation Strategies

I analyze the performance of the three elicitation strategies described above—Myopic, Opt-Approx and Pess-Approx—as Ill as a Random strategy (with target queries chosen at random, but eliciting at midpoints as above). I test on small problems with five targets (Random-S, etc.) and large problems with 20 targets (Random-L, etc.). Fig. 7.7(a) shows how MMR decreases as I ask queries using the different strategies. I see that Myopic is the most effective strategy, folloId

by Opt-Approx, Pess-Approx, and Random, which performs worst. Fig. 7.7(b) plots cumulative runtime of the elicitation process using my different strategies. Fig. 7.7(b) shows that Myopic is about 20 times slower than the other strategies. Opt-Approx is not much slower (1.08 times) than Random, while Pess-Approx is 1.24 times slower than Random. I use five-target games primarily to compare Myopic to the other methods; with 20 targets, Myopic is intractable and cannot be evaluated. With five targets, the strategies (unsurprisingly) do not vary much in elicitation performance. However, with 20 targets, Opt-Approx reduces MMR significantly faster than Random or Pess-Approx. Given its computational effectiveness, it seems to be a reasonable choice for payoff elicitation. However, all strategies point to tradeoffs that, depending on the task at hand and available budget, may make any of them viable.

7.4 Summary

Despite significant applications of SSGs for protecting major critical infrastructure, research on *robustness* in SSGs has, to date, focused only on one concept, maximin over interval uncertainty of payoffs. In this chapter, I have proposed the use of MMR as a decision criterion for payoff-uncertain SSGs and presents efficient algorithms for computing MMR for such games. Furthermore, I have addressed, for the first time, the challenge of preference elicitation in SSGs, providing novel regret-based solution strategies. Experimental results validate the effectiveness of my approaches w.r.t. both computational and informational efficiency.

Chapter 8

Regret-based Solutions for Wildlife Protection

Motivated by the effectiveness of regret-based solutions for security games provided in Chapter 7, this chapter focuses on developing new regret-based solutions for addressing payoff uncertainty in wildlife protection. In particular, the regret-based solutions in Chapter 7 are mainly designed for addressing uncertainty in the attacker's payoffs in standard SSGs, assuming that the attacker is perfectly rational. In this chapter, I introduce new regret-based solutions which can handle uncertainty in both players' payoffs in different game scenarios in which the attacker is either boundedly rational or perfectly rational. Furthermore, I develop new elicitation strategies to strategically reduce uncertainty, given available elicitation resources in wildlife domains — mobile sensors such as Unmanned Aerial Vehicles (UAV).

8.1 Overview

Prior work on wildlife protection are founded on the promise of an abundance of adversary data (about where the adversaries attacked in the past) that can be used to accurately learn adversary behavior models which capture their bounded rationality (Yang et al., 2014; Fang et al., 2015). Furthermore, game-theoretic research on wildlife protection assumes that available domain data such as animal density is sufficient to help determine payoff values precisely. However, there remain four key shortcomings in wildlife protection related to these assumptions about data. First, despite proposing different adversary behavioral models (e.g., Quantal Response (Yang et al., 2012)), research on wildlife protection has yet to evaluate these models on any real-world data. Second, the amount of real-world data available is not always present in abundance, introducing

different types of uncertainties in wildlife protection. In particular, in some wildlife protection domains, there is a significant need to handle uncertainty in both the defender and the adversary's payoffs since information on key domain features, e.g., animal density, terrain, etc. that contribute to the payoffs is not precisely known. Third, in some wildlife protection domains, we may even lack sufficient attack data to learn an adversary behavior model, and simultaneously must handle the aforementioned payoff uncertainty. Finally, defenders have access to mobile sensors such as UAVs to elicit information over multiple targets at once to reduce payoff uncertainty, no efficient technique has been provided to exploit such sensors for payoff elicitation.

In this chapter, I address these challenges by proposing four key contributions. As my first contribution, I provide the first results demonstrating the usefulness of behavioral models in SSGs using real-world data from a wildlife park. To address the second limitation of uncertainty over payoff values, my second contribution is ARROW (i.e., **A**lgorithm for **R**educing **R**egret to **O**ppose **W**ildlife crime), a novel security game algorithm that can solve the *behavioral minimax regret problem*. ARROW is the first algorithm to compute MMR in the presence of an adversary behavioral model; it is also the first to handle payoff uncertainty in both players' payoffs in SSGs. However, jointly handling of adversary bounded rationality and payoff uncertainty creates the challenge of solving a non-convex optimization problem; ARROW provides an efficient solution to this problem. (Note that I primarily assume a zero-sum game as done in some prior wildlife protection research; however as discussed my key techniques generalize to non-zero sum games as well.) My third contribution addresses situations where I do not even have data to learn a behavior model. Specifically, I propose ARROW-Perfect, a novel MMR-based algorithm to handle uncertainty in *both* players' payoffs, assuming a perfectly rational adversary without any requirement of data for learning. ARROW-Perfect exploits the adversary's perfect rationality as well as extreme points of payoff uncertainty sets to gain significant additional efficiency over ARROW. Fourth, I present two new elicitation heuristics which select *multiple* targets at a time for reducing payoff uncertainty, leveraging the multi-target-elicitation capability of sensors (e.g., UAVs) available in green security domains. Lastly, I conduct extensive experiments, including evaluations of ARROW based on data from a wildlife park.

The outline of the chapter as follows: 1) First, I present the results of behavioral modeling validation based on real-world wildlife/poaching data; 2) Second, I introduce the concept of

behavioral minimax regret and describe the ARROW algorithm to solve it; 3) Third, I provide the new regret-based algorithm, ARROW-Perfect; 4) Fourth, I present new elicitation heuristics; and 5) Finally, I provide experimental results.

8.2 Behavioral Modeling Validation

My first contribution addresses the first limitation of previous work mentioned in Section 8.1: understanding the extent to which existing behavior models capture real-world behavior data from green security domains. I used a real-world patrol and poaching dataset from Uganda Wildlife Authority supported by Wildlife Conservation Society. This dataset was collected from 1-year patrols in the Queen Elizabeth national park.[1]

8.2.1 Dataset Description

My dataset had different types of observations (poacher sighting, animal sighting, etc.) with $40,611$ observations in total recorded by rangers at various locations in the park. The latitude and longitude of the location corresponding to each observation was recorded using a GPS device, thus providing reliable data. Each observation has a feature that specified the total count of the category of observation recorded, for example, number and type of animals sighted or poaching attacks identified, at a particular location. The date and time for a particular patrol was also present in the dataset. I discretized the park area into 2423 grid cells, with each grid cell corresponding to a $1km \times 1km$ area within the park. After the discretization, each observation fell within one of the 2423 target cells and I therefore aggregated the animal densities and the number of poaching attacks within each target cell. I considered attack data from the year 2012 in my analysis, which has 2352 attacks in total.

Gaussian smoothing of animal densities: Animal density at each target is computed based on the patrols conducted by the rangers and are thus observations at a particular instant of time. Animal density also has a spatial component, meaning that it is unlikely to change abruptly betIen grid cells. Therefore, to account for movement of animals over a few kilometers in general, I do a blurring of the current recording of animal densities over the cells. To obtain the spatial spread

[1]This is the preliminary work on modeling poachers' behaviors. Further study on building more complex behavioral models would be a new interesting research topic for future work.

based on recordings of animal sightings, I use Gaussian smoothing; more specifically I use a Gaussian Kernel of size 5×5 with $\sigma = 2.5$ to smoothen out the animal densities over all the grid cells.

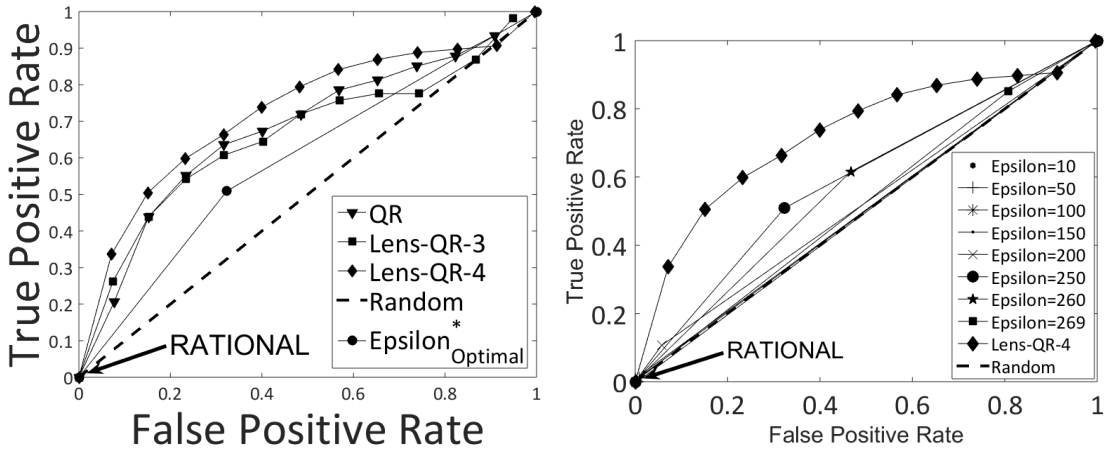
Distance as a feature: In addition to animal density, the poachers' payoffs should take into account the distance (or effort) the poacher takes in reaching the grid cell. Therefore, I also use distance as a feature of my LensQR models. Here, the subjective utility function (Equation 4.2) is extended to include the distance feature: $\hat{\mathrm{U}}^{\mathrm{a}}_{\mathrm{t}}(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}) = w_1 x_t + w_2 R^a_t + w_3 P^a_t + w_4 \Phi_t$ where Φ_t is the distance from the attacker current position to target t. For calculating distance, I took a set of 10 entry points based on geographical considerations. The distance to each target location is computed as the minimum over the distances to this target from the 10 entry points.

8.2.2 Learning Results

I compare the performance of 13 behavioral models[2] as follows (Figure 8.1): (i) LensQR-3, which corresponds to LensQR with three features (coverage probability, poacher reward which is considered to be same as the animal density and poacher penalty which is kept uniform over all targets); (ii) LensQR-4, which corresponds to LensQR with four features (coverage probability, animal density, poacher penalty and distance to the target location); (iii) QR; (iv) eight versions of the ϵ-optimal model, a bounded rationality model (Pita et al., 2009) where the adversary chooses to attack any one of the targets with an utility value which is within ϵ of the optimal target's utility, with equal probability; (v) a random adversary model; and (vi) a perfectly rational model.

From the 2352 total attacks in my dataset, I randomly sampled (10 times) 20% of the attack data for testing and trained the three models: LensQR-3, LensQR-4 and QR on the remaining 80% data. For each train-test split, I trained my behavioral models to learn their parameters, which are used to get probabilities of attack on each grid cell in the test set. Thus, for each grid cell, I get the actual label (whether the target was attacked or not) along with my predicted probability of attack on the cell. Using these labels and the predicted probabilities, I plotted a Receiver Operating Characteristic (ROC) curve (in Figure 8.1) to analyze the performance of the various models.

[2]Models involving cognitive hierarchies (Wright & Leyton-Brown, 2014) are not applicable in Stackelberg games given that attacker plays knowing the defender's actual strategy.



(a) All Models

(b) ϵ-optimal (various values of ϵ) vs LensQR-4

Figure 8.1: ROC plots on Uganda dataset

The result shows that the perfectly rational model, that deterministically classifies which target gets attacked (unlike LensQR/QR which give probabilities of attack on all targets), achieves an extremely poor prediction accuracy. I also observe that the ϵ^*-optimal model performs worse than QR and LensQR models (Figure 8.1(a)). Here, by ϵ^*-optimal model, I mean the model corresponding to the ϵ that generates the best prediction (Figure 8.1(b)). In my case, the best value of ϵ is 250. For the ϵ-optimal model, no matter what ϵ I choose, the curves from the ϵ-optimal method never gets above the LensQR-4 curve, demonstrating that LensQR-4 is a better model than ϵ-optimal. Furthermore, LensQR-4 (Area Under the Curve (AUC) = 0.73) performs better than both QR (AUC = 0.67) and LensQR-3 (AUC = 0.67), thus highlighting the importance of distance as a feature in the adversary's utility. Thus, LensQR-4 provides the highest prediction accuracy and thus will be my model of choice in the rest of the paper.

In summary, comparing many different models shows for the first time support for LensQR from real-world data in the context of GSGs. The LensQR-4 model convincingly beats QR, $\epsilon-$optimal, perfect-rationality and the random model, thus showing the validity of using LensQR in predicting adversary behaviors in GSGs.

8.3 Behavioral Minimax Regret ($\mathrm{MMR_b}$)

While we can learn a behavioral model from real-world data, we may not always have access to data to precisely compute animal density. For example, given limited numbers of rangers,

they may have patrolled and collected wildlife data from only a small portion of a national park, and thus payoffs in other areas of the park may remain uncertain. Also, due to the dynamic changes (e.g., animal migration), players' payoffs may become uncertain in the next season. Hence, this chapter introduces my new MMR-based robust algorithm, ARROW, to handle payoff uncertainty in wildlife protection, taking into account adversary behavioral models. Here, I primarily focus on zero-sum games. In addition, I use a model inspired by LensQR-4 as the adversary's behavioral model, given its high prediction accuracy presented in Section 8.2. More specifically, the subjective utility function in Equation (4.2) is extended to: $\hat{\mathrm{U}}^{\mathrm{a}}_{\mathrm{t}}(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}) = w_1 x_t + w_2 R^a_t + w_3 P^a_t + w_4 \Phi_t$ where Φ_t is some other feature (e.g., distance) of target t. In fact, my methods generalize to non-zero-sum games with a general class of QR. I now formulate $\mathrm{MMR_b}$ with uncertain payoffs for both players in zero-sum SSG with a boundedly rational attacker.

Definition 5. *Given* $(\mathbf{R^a}, \mathbf{P^a})$*, the defender's* ***behavioral regret*** *is the loss in her utility for playing a strategy* $\mathbf{x}$ *instead of the optimal strategy, which is represented as follows:*

$$\mathrm{R_b}(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}) = \max_{\mathbf{x'} \in \mathbf{X}} \mathrm{F}(\mathbf{x'}, \mathbf{R^a}, \mathbf{P^a}) - \mathrm{F}(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}) \tag{8.1}$$

$$\text{where } \mathrm{F}(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}) = \sum\nolimits_t \hat{\mathrm{q}}_{\mathrm{t}}(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}) \mathrm{U}^{\mathrm{d}}_{\mathrm{t}}(\mathbf{x}, \mathbf{R^d}, \mathbf{P^d}) \tag{8.2}$$

Behavioral regret measures the distance in terms of utility loss from the defender strategy $\mathbf{x}$ to the optimal strategy given the attacker payoffs. Here, $\mathrm{F}(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a})$ is the defender's utility (which is non-convex fractional in $\mathbf{x}$) for playing $\mathbf{x}$ where the attacker payoffs, whose response follows LensQR, are $(\mathbf{R^a}, \mathbf{P^a})$. The defender's payoffs in zero-sum games are $\mathbf{R^d} = -\mathbf{P^a}$ and $\mathbf{P^d} = -\mathbf{R^a}$. In addition, the attacking probability, $\hat{\mathrm{q}}_{\mathrm{t}}(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a})$, is given by Equation 4.2. When the payoffs are uncertain, if the defender plays a strategy $\mathbf{x}$, she receives different behavioral regrets w.r.t to different payoff instances within the uncertainty intervals. Thus, she could receive a **behavioral max regret** which is defined as follows:

Definition 6. *Given payoff intervals* $\mathbf{I}$*, the* ***behavioral max regret*** *for the defender to play a strategy* $\mathbf{x}$ *is the maximum behavioral regret over all payoff instances:*

$$\mathrm{MR_b}(\mathbf{x}, \mathbf{I}) = \max_{(\mathbf{R^a}, \mathbf{P^a}) \in \mathbf{I}} \mathrm{R_b}(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}) \tag{8.3}$$

Definition 7. *Given payoff intervals* $\mathbf{I}$*, the* ***behavioral minimax regret*** *problem attempts to find the defender optimal strategy that minimizes the* $\mathrm{MR_b}$ *she receives:*

$$\mathrm{MMR_b}(\mathbf{I}) = \min_{\mathbf{x} \in \mathbf{X}} \mathrm{MR_b}(\mathbf{x}, \mathbf{I}) \tag{8.4}$$

Intuitively, behavorial minimax regret ensures that the defender's strategy minimizes the loss in the solution quality over the uncertainty of all possible payoff realizations.

Example 12. *In the 2-target zero-sum game as shown in Table 8.1, each target is associated with uncertainty intervals of the attacker's reward and penalty. For example, if the adversary successfully attacks Target 1, he obtains a reward which belongs to the interval* $[2, 3]$*. Otherwise, he receives a penalty which lies within the interval* $[-2, 0]$*. The attacker's response, assumed to follow LensQR, is defined by the parameters* $(w_1 = -10.0, w_2 = 2.0, w_3 = 0.2, w_4 = 0.0)$*.*

Table 8.1: A 2-target, 1-resource game.

Targets	*Attacker reward.*	*Attacker penalty.*
1	*[2, 3]*	*[-2, 0]*
2	*[5, 7]*	*[-10, -9]*

Then the defender's optimal mixed strategy generated by behavioral MMR (Equation 8.4) corresponding to this LensQR model is $\mathbf{x} = \{0.35, 0.65\}$*. The attacker payoff values which give the defender the maximum regret w.r.t this behavioral MMR strategy are* $(3.0, 0.0)$ *and* $(5.0, -10.0)$ *at Target 1 and 2 respectively. In particular, the defender obtains an expected utility of* -0.14 *for playing* $\mathbf{x}$ *against this payoff instance. On the other hand, she would receive a utility of* 2.06 *if playing the optimal strategy* $\mathbf{x}' = \{0.48, 0.52\}$ *against this payoff instance. As a result, the defender gets a maximum regret of 2.20.*

8.4 ARROW Algorithm: Boundedly Rational Attacker

Algorithm 6 presents the outline of ARROW to solve the $\mathrm{MMR_b}$ problem in Equation 8.4. Essentially, ARROW's two novelties — addressing uncertainty in both players' payoffs and a boundedly rational attacker — lead to two new computational challenges: 1) uncertainty in defender payoffs makes the defender's expected utility at every target t non-convex in $\mathbf{x}$ and $(\mathbf{R^d}, \mathbf{P^d})$

Algorithm 6: ARROW Outline

1 Initialize $S = \phi, ub = \infty, lb = 0$;
2 Randomly generate sample $(\mathbf{x}', \mathbf{R^a}, \mathbf{P^a})$, $S = S \cup \{\mathbf{x}', (\mathbf{R^a}, \mathbf{P^a})\}$;
3 **while** $ub > lb$ **do**
4 Call R.ARROW to compute relaxed $\mathrm{MMR_b}$ w.r.t S. Let $\mathbf{x}^*$ be its optimal solution with objective value lb;
5 Call M.ARROW to compute $\mathrm{MR_b}(\mathbf{x}^*, \mathbf{I})$. Let the optimal solution be $(\mathbf{x}'^{,*}, \mathbf{R^{a,*}}, \mathbf{P^{a,*}})$ with objective value ub;
6 $S = S \cup \{\mathbf{x}'^{,*}, \mathbf{R^{a,*}}, \mathbf{P^{a,*}}\}$;
7 **end**
8 return $(lb, \mathbf{x}^*)$;

(Equation 2.2); and 2) the LensQR model is in the form of a logit function which is non-convex. These two non-convex functions are combined when calculating the defender's utility (Equation 8.2) — which is then used in computing $\mathrm{MMR_b}$ (Equation 8.4), making it computationally expensive. Overall, $\mathrm{MMR_b}$ can be reformulated as minimizing the max regret r such that r is no less than the behavioral regrets over all payoff instances within the intervals:

$$\min_{\mathbf{x}\in\mathbf{X}, r\in\mathbb{R}} r \tag{8.5}$$
$$\text{s.t. } r \geq \mathrm{F}(\mathbf{x}', \mathbf{R^a}, \mathbf{P^a}) - \mathrm{F}(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}), \forall (\mathbf{R^a}, \mathbf{P^a}) \in \mathbf{I}, \mathbf{x}' \in \mathbf{X}$$

In (8.5), the set of (non-convex) constraints is infinite since $\mathbf{X}$ and $\mathbf{I}$ are continuous. One practical approach to optimization with large constraint sets is *constraint sampling* (De Farias & Van Roy, 2004), coupled with *constraint generation* (Boutilier et al., 2006). Following this approach, ARROW samples a subset of constraints in Problem (8.5) and gradually expands this set by adding violated constraints to the relaxed problem until convergence to the optimal $\mathrm{MMR_b}$ solution.

Specifically, ARROW begins by sampling pairs $(\mathbf{R^a}, \mathbf{P^a})$ of the adversary payoffs uniformly from $\mathbf{I}$. The corresponding optimal strategies for the defender given these payoff samples, denoted $\mathbf{x}'$, are then computed using the PASAQ algorithm (Yang et al., 2012) to obtain a finite set S of sampled constraints (Line 2). These sampled constraints are then used to solve the corresponding *relaxed* $\mathrm{MMR_b}$ program (line 4) using the R.ARROW algorithm (described in Section 8.4.1) — I call this problem *relaxed* $\mathrm{MMR_b}$ as it only has samples of constraints in (8.5). I thus obtain the optimal solution $(lb, \mathbf{x}^*)$ which provides a loIr bound (lb) on the true $\mathrm{MMR_b}$. Then constraint generation is applied to determine violated constraints (if any). This uses the M.ARROW algorithm (described in Section 8.4.2) which computes $\mathrm{MR_b}(\mathbf{x}^*, \mathbf{I})$ — the optimal

regret of $\mathbf{x}^*$ which is an upper bound (ub) on the true $\mathrm{MMR_b}$. If $ub > lb$, the optimal solution of M.ARROW, $\{\mathbf{x}'^{,*}, \mathbf{R}^{\mathbf{a},*}, \mathbf{P}^{\mathbf{a},*}\}$, provides the maximally violated constraint (line 5), which is added to S. Otherwise, $\mathbf{x}^*$ is the minimax optimal strategy and $lb = ub = \mathrm{MMR_b}(\mathbf{I})$.

8.4.1 R.ARROW: Compute Relaxed $\mathrm{MMR_b}$

The first step of ARROW is to solve the relaxed $\mathrm{MMR_b}$ problem using R.ARROW. This relaxed $\mathrm{MMR_b}$ problem is non-convex. Thus, R.ARROW presents two key ideas for efficiency: 1) binary search (which iteratively searches the defender's utility space to find the optimal solution) to remove the fractional terms (i.e., the attacking probabilities in Equation 4.2) in relaxed $\mathrm{MMR_b}$; and 2) it then applies piecewise-linear approximation to linearize the non-convex terms of the resulting decision problem at each binary search step (as explained below). Overall, relaxed $\mathrm{MMR_b}$ can be represented as follows:

$$\begin{aligned} &\min_{\mathbf{x}\in\mathbf{X}, r\in\mathbb{R}} r \\ &\text{s.t. } r \geq \mathrm{F}(\mathbf{x}'^{,\mathbf{k}}, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}}) - \mathrm{F}(\mathbf{x}, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}}), \forall k = 1 \dots K \end{aligned} \tag{8.6}$$

where $(\mathbf{x}'^{,\mathbf{k}}, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}})$ is the k^{th} sample in S (i.e., the payoff sample set as described in Algorithm 6) where $k = 1 \dots K$ and K is the total number of samples in S. In addition, r is the defender's max regret for playing $\mathbf{x}$ against sample set S. Finally, $\mathrm{F}(\mathbf{x}'^{,\mathbf{k}}, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}})$ is the defender's optimal utility for every sample of attacker payoffs $(\mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}})$ where $\mathbf{x}'^{,\mathbf{k}}$ is the corresponding defender's optimal strategy (which can be obtained via PASAQ (Yang et al., 2012)). The term $\mathrm{F}(\mathbf{x}, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}})$, which is included in relaxed $\mathrm{MMR_b}$'s constraints, is non-convex and fractional in $\mathbf{x}$ (Equation 8.2), making (8.6) non-convex and fractional. I now detail the two key ideas of R.ARROW.

Binary search. In each binary search step, given a value of r, R.ARROW tries to solve the decision problem (**P1**) that determines if there exists a defender strategy $\mathbf{x}$ such that the defender's regret for playing $\mathbf{x}$ against any payoff sample in S is no greater than r.

(P1) : $\exists \mathbf{x}$ s.t. $r \geq \mathrm{F}(\mathbf{x}'^{,\mathbf{k}}, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}}) - \mathrm{F}(\mathbf{x}, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}}), \forall k = 1 \dots K$?

I present the following Proposition 8 showing that (**P1**) can be converted into the *non-fractional* optimization problem (**P2**) (as shown below) of which the optimal solution is used to determine the feasibility of (**P1**):

$$\textbf{(P2):} \min_{\mathbf{x}\in\mathbf{X}, v\in\mathbb{R}} v$$
$$\text{s.t. } v \geq \sum\nolimits_t \left[\mathrm{F}(\mathbf{x}^{\prime,\mathbf{k}}, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}}) - r - \mathrm{U}_{\mathrm{t}}^{\mathrm{d,k}}(\mathbf{x})\right] e^{\hat{\mathrm{U}}_{\mathrm{t}}^{\mathrm{a}}(\mathbf{x},\mathbf{R}^{\mathbf{a},\mathbf{k}},\mathbf{P}^{\mathbf{a},\mathbf{k}})}, \forall k = 1 \dots K$$

where $\mathrm{U}_{\mathrm{t}}^{\mathrm{d,k}}(\mathbf{x}) = -\left[x_t P_t^{a,k} + (1 - x_t) R_t^{a,k}\right]$ is the defender's expected utility at target t given $\mathbf{x}$ and the k^{th} payoff sample.

Proposition 8. *Suppose that $(v^*, \mathbf{x}^*)$ is the optimal solution of (**P2**). If $v^* \leq 0$, then $\mathbf{x}^*$ is a feasible solution of the decision problem (**P1**). Otherwise, (**P1**) is infeasible.*

Proof. If $v^* \leq 0$, the RHS of all constraints in (**P2**) must be no greater than zero. By dividing these RHSs by the term $\sum_t e^{\hat{\mathrm{U}}_{\mathrm{t}}^{\mathrm{a}}(\mathbf{x}^*,\mathbf{R}^{\mathbf{a}},\mathbf{P}^{\mathbf{a}})}$, we obtain the following inequalities (Equation 8.7), implying that the regret for playing $\mathbf{x}^*$ against any payoff sample is no greater than the regret value r, indicating that $\mathbf{x}^*$ is a feasible solution of (**P1**).

$$r \geq \mathrm{F}\left(\mathbf{x}^{\prime,\mathbf{k}}, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}}\right) - \mathrm{F}\left(\mathbf{x}^*, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}}\right), \forall k. \tag{8.7}$$

Conversely, if $v^* > 0$, then for every $\mathbf{x} \in \mathbf{X}$, there exists $k \in \{1, \dots, K\}$ such that the RHS of the k^{th} constraint in (**P2**) is greater than zero (otherwise we can always find a feasible solution $v \leq 0 < v^*$ which is a contradiction to our assumption that v^* is optimal). As a result, by dividing this RHS by $\sum_t e^{\hat{\mathrm{U}}_{\mathrm{t}}^{\mathrm{a}}(\mathbf{x},\mathbf{R}^{\mathbf{a}},\mathbf{P}^{\mathbf{a}})}$, we still obtain a positive value, meaning that for all strategies $\mathbf{x} \in \mathbf{X}$, there exists $k \in \{1, \dots, K\}$ such that $r < \mathrm{F}\left(\mathbf{x}^{\prime,\mathbf{k}}, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}}\right) - \mathrm{F}\left(\mathbf{x}, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}}\right)$. Therefore, (**P1**) is infeasible. □

Given that the decision problem **(P1)** is now converted into the optimization problem **(P2)**, as the next step, I attempt to solve **(P2)** using piecewise linear approximation.

Piecewise linear approximation. Although (**P2**) is non-fractional, its constraints are non-convex. I use a piecewise linear approximation for the RHS of the constraints in (**P2**) which is in the form of $\sum_t f_t^k(x_t)$ where the term $f_t^k(x_t)$ is a non-convex function of x_t (recall that x_t is the defender's coverage probability at target t). The feasible region of the defender's coverage x_t for all t, $[0, 1]$, is then divided into M equal segments $\left\{\left[0, \frac{1}{M}\right], \left[\frac{1}{M}, \frac{2}{M}\right], \dots, \left[\frac{M-1}{M}, 1\right]\right\}$ where

M is given. The values of $f_t^k(x_t)$ are then approximated by using the segments connecting pairs of consecutive points $\left(\frac{i-1}{M}, f_t^k\left(\frac{i-1}{M}\right)\right)$ and $\left(\frac{i}{M}, f_t^k\left(\frac{i}{M}\right)\right)$ for $i = 1 \dots M$ as follows:

$$f_t^k(x_t) \approx f_t^k(0) + \sum_{i=1}^{M} \alpha_{t,i}^k x_{t,i} \tag{8.8}$$

where $\alpha_{t,i}^k$ is the slope of the i^{th} segment which can be determined based on the two extreme points of the segment. Also, $x_{t,i}$ refers to the portion of the defender's coverage at target t belonging to the i^{th} segment, i.e., $x_t = \sum_i x_{t,i}$.

Example 13. *When the number of segments $M = 5$, it means that I divide $[0, 1]$ into 5 segments $\left\{\left[0, \frac{1}{5}\right], \left[\frac{1}{5}, \frac{2}{5}\right], \left[\frac{2}{5}, \frac{3}{5}\right], \left[\frac{3}{5}, \frac{4}{5}\right], \left[\frac{4}{5}, 1\right]\right\}$. Suppose that the defender's coverage at target t is $x_t = 0.3$, since $\frac{1}{5} < x_t < \frac{2}{5}$, I obtain the portions of x_t that belongs to each segment is $x_{t,1} = \frac{1}{5}$, $x_{t,2} = 0.1$, and $x_{t,3} = x_{t,4} = x_{t,5} = 0$ respectively. Then each non-linear term $f_t^k(x_t)$ is approximated as $f_t^k(x_t) \approx f_t^k(0) + \frac{1}{5}\alpha_{t,1}^k + 0.1\alpha_{t,2}^k$ where the slopes of the 1^{st} and 2^{nd} segments are $\alpha_{t,1}^k = 5\left[f_t^k\left(\frac{1}{5}\right) - f_t^k(0)\right]$ and $\alpha_{t,2}^k = 5\left[f_t^k\left(\frac{2}{5}\right) - f_t^k\left(\frac{1}{5}\right)\right]$ respectively.*

By using the approximations of $f_t^k(x_t)$ for all k and t, I can reformulate (**P2**) as the MILP **(P2')** which can be solved by the solver CPLEX:

$$\textbf{(P2')}: \min_{x_{t,i}, z_{t,i}, v} v \tag{8.9}$$

$$\text{s.t. } v \geq \sum_t f_t^k(0) + \sum_t \sum_i \alpha_{t,i}^k x_{t,i}, \forall k = 1 \dots K \tag{8.10}$$

$$\sum_{t,i} x_{t,i} \leq R, 0 \leq x_{t,i} \leq \frac{1}{M}, \forall t = 1 \dots T, i = 1 \dots M \tag{8.11}$$

$$z_{t,i} \frac{1}{M} \leq x_{t,i}, \forall t = 1 \dots T, i = 1 \dots M - 1 \tag{8.12}$$

$$x_{t,i+1} \leq z_{t,i}, \forall t = 1 \dots T, i = 1 \dots M - 1 \tag{8.13}$$

$$z_{t,i} \in \{0, 1\}, \forall t = 1 \dots T, i = 1 \dots M - 1 \tag{8.14}$$

where $z_{t,i}$ is an auxiliary integer variable which ensures that the portions of x_t satisfies $x_{t,i} = \frac{1}{M}$ if $x_t \geq \frac{i}{M}$ ($z_{t,i} = 1$) or $x_{t,i+1} = 0$ if $x_t < \frac{i}{M}$ ($z_{t,i} = 0$) (constraints (8.11 – 8.14)). Constraints (8.10) are piecewise linear approximations of constraints in **(P2)**. In addition, constraint (8.11) guarantees that the resource allocation condition, $\sum_t x_t \leq R$, holds true and the piecewise segments $0 \leq x_{t,i} \leq \frac{1}{M}$.

Finally, I provide Theorem 3 showing that R.ARROW guarantees a solution bound on computing relaxed $\mathrm{MMR_b}$.

Theorem 3. *R.ARROW provides an $O\left(\epsilon + \frac{1}{M}\right)$-optimal solution of relaxed $\mathrm{MMR_b}$ where ϵ is the tolerance of binary search and M is the number of piecewise segments.*

We apply the following lemmas for proving Theorem 3:

Lemma 1. *Denote by $\overline{\mathrm{F}}\left(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}\right)$ the piecewise linear approximation of the defender's utility $\mathrm{F}\left(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}\right)$, then the difference $\left|\mathrm{F}\left(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}\right) - \overline{\mathrm{F}}\left(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}\right)\right|$ is bounded by $O\left(\frac{1}{M}\right)$ where M is the number of piecewise segments.*

Proof. Overall, the defender's utility $\mathrm{F}\left(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}\right)$ has the form $\mathrm{F}\left(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}\right) = \frac{\mathrm{N}(\mathbf{x})}{\mathrm{D}(\mathbf{x})}$ where $\mathrm{N}(\mathbf{x}) = \sum_t e^{\hat{\mathrm{U}}^{\mathrm{a}}_{\mathrm{t}}(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a})} \mathrm{U}^{\mathrm{d}}_{\mathrm{t}}(\mathbf{x}, \mathbf{R^d}, \mathbf{P^d})$ and $\mathrm{D}(\mathbf{x}) = \sum_t e^{\hat{\mathrm{U}}^{\mathrm{a}}_{\mathrm{t}}(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a})}$ where $\mathbf{R^d} = -\mathbf{P^a}$ and $\mathbf{P^d} = -\mathbf{R^a}$ in zero-sum games. Similarly, the approximate utility $\overline{\mathrm{F}}\left(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}\right)$ has the form $\overline{\mathrm{F}}\left(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}\right) = \frac{\overline{\mathrm{N}}(\mathbf{x})}{\overline{\mathrm{D}}(\mathbf{x})}$ where $\overline{\mathrm{N}}(\mathbf{x})$ and $\overline{\mathrm{D}}(\mathbf{x})$ are piecewise linear approximations of the enumerator $\mathrm{N}(\mathbf{x})$ and the denominator $\mathrm{D}(\mathbf{x})$ respectively. We obtain the following inequality on the approximation error:

$$\begin{aligned}
&\left|\mathrm{F}\left(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}\right) - \overline{\mathrm{F}}\left(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}\right)\right| = \left|\frac{\mathrm{N}(\mathbf{x})}{\mathrm{D}(\mathbf{x})} - \frac{\overline{\mathrm{N}}(\mathbf{x})}{\overline{\mathrm{D}}(\mathbf{x})}\right| \\
&= \left|\frac{\mathrm{N}(\mathbf{x}) - \overline{\mathrm{N}}(\mathbf{x})}{\mathrm{D}(\mathbf{x})} - \overline{\mathrm{N}}(\mathbf{x})\left[\frac{1}{\overline{\mathrm{D}}(\mathbf{x})} - \frac{1}{\mathrm{D}(\mathbf{x})}\right]\right| \\
&\leq \left|\mathrm{N}(\mathbf{x}) - \overline{\mathrm{N}}(\mathbf{x})\right| \frac{1}{|\mathrm{D}(\mathbf{x})|} + \left|\mathrm{D}(\mathbf{x}) - \overline{\mathrm{D}}(\mathbf{x})\right| \frac{\left|\overline{\mathrm{N}}(\mathbf{x})\right|}{|\mathrm{D}(\mathbf{x})|\left|\overline{\mathrm{D}}(\mathbf{x})\right|}.
\end{aligned} \tag{8.15}$$

As $\mathrm{N}(\mathbf{x})$ and $\mathrm{D}(\mathbf{x})$ are continuous functions over the compact strategy set $\mathbf{X}$, the two terms $\frac{1}{|\mathrm{D}(\mathbf{x})|}$ and $\frac{\left|\overline{\mathrm{N}}(\mathbf{x})\right|}{|\mathrm{D}(\mathbf{x})|\left|\overline{\mathrm{D}}(\mathbf{x})\right|}$ are bounded. Thus, there exist constants $C^1, C^2 \geq 0$ such that the following inequality holds true:

$$\left|\mathrm{F}\left(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}\right) - \overline{\mathrm{F}}\left(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}\right)\right| \leq C^1\left|\mathrm{N}(\mathbf{x}) - \overline{\mathrm{N}}(\mathbf{x})\right| + C^2\left|\mathrm{D}(\mathbf{x}) - \overline{\mathrm{D}}(\mathbf{x})\right| \tag{8.16}$$

On the other hand, the error for piecewise linearly approximating $\mathrm{D}(\mathbf{x})$ satisfies:

$$\left|\mathrm{D}(\mathbf{x}) - \overline{\mathrm{D}}(\mathbf{x})\right| = \left|\sum\nolimits_t \mathrm{L_t}(x_t) - \overline{\mathrm{L}}_{\mathrm{t}}(x_t)\right| \leq \sum\nolimits_t \left|\mathrm{L_t}(x_t) - \overline{\mathrm{L}}_{\mathrm{t}}(x_t)\right| \tag{8.17}$$

where $\mathrm{L_t}(x_t) = e^{\hat{\mathrm{U}}^{\mathrm{a}}_{\mathrm{t}}(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a})}$. In addition, suppose that $x_t \in \left[\frac{i-1}{M}, \frac{i}{M}\right]$ for some $1 \leq i \leq M$, according to the approximation, the following condition holds:

$$\min\left\{\mathrm{L_t}\left(\frac{i-1}{M}\right), \mathrm{L_t}\left(\frac{i}{M}\right)\right\} \leq \overline{\mathrm{L}}_{\mathrm{t}}(x_t) \leq \max\left\{\mathrm{L_t}\left(\frac{i-1}{M}\right), \mathrm{L_t}\left(\frac{i}{M}\right)\right\} \tag{8.18}$$

According to (8.18) and since $\mathrm{L_t}(x_t)$ is a continuous function, there exists $\hat{x}_t \in \left[\frac{i-1}{M}, \frac{i}{M}\right]$ such that the value of $\mathrm{L_t}(x_t)$ at $\hat{x}_t$ is equal to the value of its approximation at x_t, i.e., $\mathrm{L_t}(\hat{x}_t) = \overline{\mathrm{L}}_\mathrm{t}(x_t)$. As a result, we obtain $\left|\mathrm{L_t}(x_t) - \overline{\mathrm{L}}_\mathrm{t}(x_t)\right| = \left|\mathrm{L_t}(x_t) - \mathrm{L_t}(\hat{x}_t)\right|$. On the other hand, according to the Lagrange mean value theorem, there exists $a \in [\min\{x_t, \hat{x}_t\}, \max\{x_t, \hat{x}_t\}]$ such that the derivative at a satisfies the following equality:

$$\mathrm{L}'_\mathrm{t}(a) = \frac{\mathrm{L_t}(x_t) - \mathrm{L_t}(\hat{x}_t)}{x_t - \hat{x}_t} \tag{8.19}$$

Thus, we obtain the following inequality:

$$\left|\mathrm{L_t}(x_t) - \overline{\mathrm{L}}_\mathrm{t}(x_t)\right| = \left|\mathrm{L_t}(x_t) - \mathrm{L_t}(\hat{x}_t)\right| = \left|\mathrm{L}'_\mathrm{t}(a)\right| \left|x_t - \hat{x}_t\right| \leq \frac{1}{M} \max_{x_t \in [0,1]} \left|\mathrm{L}'_\mathrm{t}(x_t)\right| \tag{8.20}$$

As a result, denote by $C_t = \max_{x_t \in [0,1]} \left|\mathrm{L}'_\mathrm{t}(x_t)\right|$ the maximum derivative value of $\mathrm{L_t}(x_t)$ over the range $[0, 1]$, by combining (8.17) and (8.20), we obtain an upper bound on the error for approximating $\mathrm{D}(\mathbf{x})$ as the follows:

$$\left|\mathrm{D}(\mathbf{x}) - \overline{\mathrm{D}}(\mathbf{x})\right| \leq \frac{1}{M} \sum\nolimits_t C_t = O\left(\frac{1}{M}\right) \tag{8.21}$$

Similarly, we also have $\left|\mathrm{N}(\mathbf{x}) - \overline{\mathrm{N}}(\mathbf{x})\right|$ to be bounded by $O\left(\frac{1}{M}\right)$. Finally, according to (8.16) and the error bounds $O\left(\frac{1}{M}\right)$ for two terms $\left|\mathrm{L_t}(x_t) - \overline{\mathrm{L}}_\mathrm{t}(x_t)\right|$ and $\left|\mathrm{N}(\mathbf{x}) - \overline{\mathrm{N}}(\mathbf{x})\right|$, the error bound, $\left|\mathrm{F}\left(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}\right) - \overline{\mathrm{F}}\left(\mathbf{x}, \mathbf{R^a}, \mathbf{P^a}\right)\right|$, is $O\left(\frac{1}{M}\right)$. ■ □

Lemma 2. *Let lb and ub be the final lower and upper bounds of the binary search in R.ARROW, i.e., $0 \leq ub - lb \leq \epsilon$ and $\overline{\mathbf{x}}^*$ be the final solution of R.ARROW, we denote by $r\left(\overline{\mathbf{x}}^*\right)$ the defender's max regret for playing $\overline{\mathbf{x}}^*$ against the payoff sample set S. Then $r\left(\overline{\mathbf{x}}^*\right)$ has an upper bound of $ub + O\left(\frac{1}{M}\right)$.*

Proof. Denote by $\overline{r}\left(\overline{\mathbf{x}}^*\right)$ the piecewise approximate max regret obtained by R.ARROW, then we have $lb \leq \overline{r}\left(\overline{\mathbf{x}}^*\right) \leq ub$. Furthermore, the regret constraint must hold: $\overline{r}\left(\overline{\mathbf{x}}^*\right) = \max_k \mathrm{F}\left(\mathbf{x}'^{,\mathbf{k}}, \mathbf{R^{a,k}}, \mathbf{P^{a,k}}\right) - \overline{\mathrm{F}}\left(\overline{\mathbf{x}}^*, \mathbf{R^{a,k}}, \mathbf{P^{a,k}}\right)$.[3]

[3]For simplification, we assume the optimal utility of the defender against the k^{th} payoff sample, $\mathrm{F}\left(\mathbf{x}'^{,\mathbf{k}}, \mathbf{R^{a,k}}, \mathbf{P^{a,k}}\right)$, is exactly computed. In fact, it is straightforward to extend this lemma to the case when this optimal utility is approximated based on the PASAQ algorithm.

On the other hand, since $r\left(\overline{\mathbf{x}}^*\right)$ is the defender's max regret for playing $\overline{\mathbf{x}}^*$ against the payoff sample set S, then $r\left(\overline{\mathbf{x}}^*\right) = \max_k \mathrm{F}\left(\mathbf{x}'^{,\mathbf{k}}, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}}\right) - \mathrm{F}\left(\overline{\mathbf{x}}^*, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}}\right)$. As a result, we obtain the following inequality:

$$\left|r\left(\overline{\mathbf{x}}^*\right) - \overline{r}\left(\overline{\mathbf{x}}^*\right)\right| \leq \max_k \left|\mathrm{F}\left(\overline{\mathbf{x}}^*, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}}\right) - \overline{\mathrm{F}}\left(\overline{\mathbf{x}}^*, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}}\right)\right| \tag{8.22}$$

According to Lemma 1, the RHS of this inequality is bounded by $O\left(\frac{1}{M}\right)$. Since $\overline{r}\left(\overline{\mathbf{x}}^*\right) \leq ub$, then $r\left(\overline{\mathbf{x}}^*\right)$ has an upper bound of $ub + O\left(\frac{1}{M}\right)$. ■ □

Lemma 3. *Denote by $\mathbf{x}^*$ the optimal solution of the relaxed MMR problem and $r\left(\mathbf{x}^*\right)$ the corresponding max regret, then $r\left(\mathbf{x}^*\right)$ has a lower bound of $lb + O\left(\frac{1}{M}\right)$ where lb is the final lower bound of the binary search in R.ARROW.*

Proof. Similar to the proof of Lemma 2, denote by $\overline{r}\left(\mathbf{x}^*\right)$ the piecewise approximate max regret of the defender for playing the optimal strategy $\mathbf{x}^*$, we obtain the following inequality:

$$\left|r\left(\mathbf{x}^*\right) - \overline{r}\left(\mathbf{x}^*\right)\right| \leq \max_k \left|\mathrm{F}\left(\mathbf{x}^*, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}}\right) - \overline{\mathrm{F}}\left(\mathbf{x}^*, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}}\right)\right| \tag{8.23}$$

According to Lemma 1, the RHS of this inequality is bounded by $O\left(\frac{1}{M}\right)$. Since $\overline{r}\left(\mathbf{x}^*\right) \geq \overline{r}\left(\overline{\mathbf{x}}^*\right) \geq lb$, then $r\left(\mathbf{x}^*\right)$ has a lower bound of $lb + O\left(\frac{1}{M}\right)$. ■ □

Based on Lemmas 2 and 3, we now provide the proof for Theorem 3. According to Lemma 2, we obtain the upper bound of $ub + O\left(\frac{1}{M}\right)$ on the defender's max regret for playing the R.ARROW strategy $\overline{\mathbf{x}}^*$. Furthermore, based on Lemma 3, the defender's max regret for playing the optimal relaxed $\mathrm{MMR_b}$ strategy has a lower bound of $lb + O\left(\frac{1}{M}\right)$. Since lb and ub are the final lower and upper bounds of R.ARROW's binary search, i.e., $0 \leq ub - lb \leq \epsilon$, it means that R.ARROW provides an $O\left(\epsilon + \frac{1}{M}\right)$-optimal solution for computing the relaxed $\mathrm{MMR_b}$ problem.

8.4.2 M.ARROW: Compute $\mathrm{MR_b}$

Given the optimal solution $\mathbf{x}^*$ returned by R.ARROW, the second step of ARROW is to compute $\mathrm{MR_b}$ of $\mathbf{x}^*$ using M.ARROW (line 5 in Algorithm 6). The problem of computing $\mathrm{MR_b}$ can be represented as the following non-convex maximization problem:

$$\max_{\mathbf{x}' \in \mathbf{X}, (\mathbf{R}^{\mathbf{a}}, \mathbf{P}^{\mathbf{a}}) \in \mathbf{I}} \mathrm{F}(\mathbf{x}', \mathbf{R}^{\mathbf{a}}, \mathbf{P}^{\mathbf{a}}) - \mathrm{F}(\mathbf{x}^*, \mathbf{R}^{\mathbf{a}}, \mathbf{P}^{\mathbf{a}}) \tag{8.24}$$

Overall, it is difficult to apply the same techniques used in R.ARROW for M.ARROW since it is a subtraction of two non-convex fractional functions, $\mathrm{F}(\mathbf{x}', \mathbf{R^a}, \mathbf{P^a})$ and $\mathrm{F}(\mathbf{x}^*, \mathbf{R^a}, \mathbf{P^a})$. Therefore, I use local search with multiple starting points which allows us to reach different local optima.

8.5 ARROW-Perfect Algorithm: Perfectly Rational Attacker

While ARROW incorporates an adversary behavioral model, it may not be applicable for green security domains where there may be a further paucity of data in which not only payoffs are uncertain but also parameters of the behavioral model are difficult to learn accurately. Therefore, I introduce a novel MMR-based algorithm, ARROW-Perfect, to handle uncertainty in both players' payoffs assuming a perfectly rational attacker. In general, ARROW-Perfect follows the same *constraint sampling* and *constraint generation* methodology as ARROW. Yet, by leveraging the property that the attacker's optimal response is a pure strategy (given a perfectly rational attacker) and the game is zero-sum, I obtain the *exact optimal solutions* for computing both relaxed MMR and max regret in *polynomial time* (while I cannot provide such guarantees for a boundedly rational attacker). In this case, I call the new algorithms for computing relaxed MMR and max regret: R.ARROW-Perfect and M.ARROW-Perfect respectively.

8.5.1 R.ARROW-Perfect: Compute Relaxed MMR

In zero-sum games, when the attacker is perfectly rational, the defender's utility for playing a strategy $\mathbf{x}$ w.r.t the payoff sample $(\mathbf{R^{a,k}}, \mathbf{P^{a,k}})$ is equal to $\mathrm{F}\big(\mathbf{x}, \mathbf{R^{a,k}}, \mathbf{P^{a,k}}\big) = -\mathrm{U_t^a}\big(\mathbf{x}, \mathbf{R^{a,k}}, \mathbf{P^{a,k}}\big)$ if the attacker attacks target t. Since the adversary is perfectly rational, therefore, $\mathrm{F}\big(\mathbf{x}, \mathbf{R^{a,k}}, \mathbf{P^{a,k}}\big) = -\max_t \mathrm{U_t^a}\big(\mathbf{x}, \mathbf{R^{a,k}}, \mathbf{P^{a,k}}\big)$, I can reformulate the relaxed MMR in (8.6) as the following linear minimization problem:

$$\min_{\mathbf{x}\in\mathbf{X}, r\in\mathbb{R}} r \tag{8.25}$$

$$\text{s.t. } r \geq \mathrm{F}\Big(\mathbf{x'^{,k}}, \mathbf{R^{a,k}}, \mathbf{P^{a,k}}\Big) + \mathrm{U_t^a}\Big(\mathbf{x}, \mathbf{R^{a,k}}, \mathbf{P^{a,k}}\Big), \forall k = 1 \dots K, \forall t = 1 \dots T \tag{8.26}$$

where $\mathrm{F}\big(\mathbf{x'^{,k}}, \mathbf{R^{a,k}}, \mathbf{P^{a,k}}\big)$ is the defender's optimal utility against a perfectly rational attacker w.r.t payoff sample $\big(\mathbf{R^{a,k}}, \mathbf{P^{a,k}}\big)$ and $\mathbf{x'^{,k}}$ is the corresponding optimal strategy which is the

Maximin solution. In addition, constraint (8.26) ensures that the regret $r \geq \mathrm{F}\left(\mathbf{x}^{\prime,\mathbf{k}}, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}}\right) + \max_t \mathrm{U}_{\mathrm{t}}^{\mathrm{a}}\left(\mathbf{x}, \mathbf{R}^{\mathbf{a},\mathbf{k}}, \mathbf{P}^{\mathbf{a},\mathbf{k}}\right)$ for all payoff samples. This linear program can be solved exactly in polynomial time using any linear solver, e.g., CPLEX.

8.5.2 M.ARROW-Perfect: Compute Max Regret

Computing max regret (MR) in zero-sum games presents challenges that the regret-based solutions provided in Chapter 7 can not handle since the defender's payoffs are uncertain while assuming these payoff values are known. In this chapter, I propose a new exact algorithm, M.ARROW-Perfect, to compute MR in polynomial time by exploiting insights of zero-sum games.

In zero-sum games with a perfectly rational adversary, Strong Stackelberg Equilibrium is equivalent to Maximin solution (Yin, Korzhyk, Kiekintveld, Conitzer, & Tambe, 2010). Thus, given the strategy $\mathbf{x}^*$ returned by relaxed MMR, max regret in (8.24) can be reformulated as follows:

$$\max_{\mathbf{x}' \in \mathbf{X}, (\mathbf{R}^{\mathbf{a}}, \mathbf{P}^{\mathbf{a}}) \in \mathbf{I}, v} v - \mathrm{F}(\mathbf{x}^*, \mathbf{R}^{\mathbf{a}}, \mathbf{P}^{\mathbf{a}}) \tag{8.27}$$

$$\text{s.t. } v \leq -\left[x'_t P_t^a + (1 - x'_t) R_t^a\right], \forall t \tag{8.28}$$

where v is the Maximin/SSE utility for the defender against the attacker payoff $(\mathbf{R}^{\mathbf{a}}, \mathbf{P}^{\mathbf{a}})$. Moreover, the defender's utility for playing $\mathbf{x}^*$ can be computed as $\mathrm{F}(\mathbf{x}^*, \mathbf{R}^{\mathbf{a}}, \mathbf{P}^{\mathbf{a}}) = -\left[x_j^* P_j^a + (1 - x_j^*) R_j^a\right]$ if the adversary attacks target j. Thus, I divide the attacker payoff space into T subspaces such that within the j^{th} subspace, the adversary always attacks target j against the defender strategy $\mathbf{x}^*$, for all $j = 1 \ldots T$. By solving these T sub-max regret problems corresponding to this division, my final global optimal solution of max regret will be the maximum of all T sub-optimal solutions.

Next, I will explain how to solve these sub-max regret problems. Given the j^{th} attacker payoff sub-space, I obtain the j^{th} sub-max regret problem as:

$$\max_{\mathbf{x}' \in \mathbf{X}, (\mathbf{R}^{\mathbf{a}}, \mathbf{P}^{\mathbf{a}}) \in \mathbf{I}, v} v + (x_j^* P_j^a + (1 - x_j^*) R_j^a) \tag{8.29}$$

$$\text{s.t. } v \leq -[x'_t P_t^a + (1 - x'_t) R_t^a], \forall t \tag{8.30}$$

$$x_j^* P_j^a + (1 - x_j^*) R_j^a \geq x_t^* P_t^a + (1 - x_t^*) R_t^a, \forall t \tag{8.31}$$

where constraints (8.31) ensures that the adversary attacks target j against the defender strategy $\mathbf{x}^*$. Here, constraints (8.30) are non-convex for all targets. I provide the following proposition which allows us to linearize constraints (8.30) for all targets but j.

Proposition 9. *Given target j, the lower bounds of the attacker's payoffs at all targets except j, $\{R^a_{min}(t), P^a_{min}(t)\}_{t\neq j}$, are optimal solutions of $\left\{R^a_j, P^a_j\right\}_{t\neq j}$ for the j^{th} sub-max regret problem.*

Proof. Denote by $(\mathbf{x}'^{,*}, \mathbf{R}^{\mathbf{a},*}, \mathbf{P}^{\mathbf{a},*}, v^*)$ an optimal solution of the j^{th} sub-max regret problem (8.29–8.31). We construct a new solution by replacing the payoff $\left(R^{a,*}_t, P^{a,*}_t\right)$ of the optimal solution with $(R^a_{min}(t), P^a_{min}(t))$ for all $t \neq j$ while keeping the rest of the solution unchanged. We show that this new solution is also an optimal solution.

Since $\left(\mathbf{x}'^{,*}, R^{a,*}_j, P^{a,*}_j, v^*\right)$ remains the same, we only need to verify if constraints (8.30–8.31) are satisfied for all $t \neq j$ with $(R^a_{min}(t), P^a_{min}(t))$. In constraint (8.30), as $P^{a,*}_t \geq P^a_{min}(t)$ and $R^{a,*}_t \geq R^a_{min}(t)$, we obtain the following inequality:

$$v^* \leq -\left[x'^{,*}_t P^{a,*}_t + (1 - x'^{,*}_t) R^{a,*}_t\right] \leq -\left[x'^{,*}_t P^a_{min}(t) + (1 - x'^{,*}_t) R^a_{min}(t)\right], \forall t \neq j \tag{8.32}$$

Similarly, in constraint (8.31), we obtain the follows:

$$x^*_j P^{a,*}_j + (1-x^*_j) R^{a,*}_j \geq x^*_t P^{a,*}_t + (1-x^*_t) R^{a,*}_t \geq x^*_t P^a_{min}(t) + (1-x^*_t) P^a_{min}(t), \forall t \neq j. \tag{8.33}$$

Thus, the new solution is feasible and thus an optimal solution. □

Now, only constraint (8.30) w.r.t target j remains non-convex for which I provide further steps to simplify it. Given the defender strategy $\mathbf{x}'$, I define the attack set as including all targets with the attacker's highest expected utility: $\Gamma(\mathbf{x}') = \{t : \mathrm{U}^{\mathrm{a}}_{\mathrm{t}}(\mathbf{x}', \mathbf{R}^{\mathbf{a}}, \mathbf{P}^{\mathbf{a}}) = \max_{t'} \mathrm{U}^{\mathrm{a}}_{\mathrm{t}'}(\mathbf{x}', \mathbf{R}^{\mathbf{a}}, \mathbf{P}^{\mathbf{a}})\}$. I provide the following observations based on which I can determine the optimal value of the attacker's reward at target j, R^a_j, for the sub-max regret problem (8.29–8.31) (according to the Proposition 10 below):

Observation 1. *If $\mathbf{x}'$ is the optimal solution of computing the j^{th} sub-max regret in (8.29–8.31), target j belongs to the attack set $\Gamma(\mathbf{x}')$.*

Since $\mathbf{x}'$ is the Maximin or SSE solution w.r.t attacker payoffs $(\mathbf{R}^{\mathbf{a}}, \mathbf{P}^{\mathbf{a}})$, the corresponding attack set $\Gamma(\mathbf{x}')$ has the maximal size (Kiekintveld et al., 2009). In other words, if a target t

belongs to the attack set of any defender strategy w.r.t $(\mathbf{R^a}, \mathbf{P^a})$, then $t \in \Gamma(\mathbf{x}')$. In (8.29–8.31), because target j belongs to the attack set $\Gamma(\mathbf{x}^*)$, I obtain $j \in \Gamma(\mathbf{x}')$.

Observation 2. *If $\mathbf{x}'$ is the optimal solution of computing the j^{th} sub-max regret in (8.29–8.31), the defender's coverage at target j: $x'_j \geq x^*_j$.*

Since $j \in \Gamma(\mathbf{x}')$ according to Observation 1, the defender utility for playing $\mathbf{x}'$ is equal to $v = -[x'_j P^a_j + (1 - x'_j)R^a_j]$. Furthermore, the max regret in (8.29) is always not less than zero, meaning that $v \geq -\Big[x^*_j P^a_j + (1 - x^*_j)R^a_j\Big]$. Thus, I obtain $x'_j \geq x^*_j$.

Proposition 10. *Given target j, the upper bound of the attacker's reward at j, $R^a_{max}(j)$, is an optimal solution of the attacker reward R^a_j for the j^{th} sub-max regret problem.*

Proof. Suppose that $R^a_j < R^a_{max}(j)$ is optimal in (8.29–8.31) and $\mathbf{x}'$ is the corresponding defender optimal strategy, then $v = -[x'_j P^a_j + (1 - x'_j)R^a_j]$ according to the Observation 1. I then replace R^a_j with $R^a_{max}(j)$ while other rewards/penalties and $\mathbf{x}'$ remain the same. Since $R^a_j < R^a_{max}(j)$, this new solution is also feasible for (8.29–8.31) and target j still belongs to $\Gamma(\mathbf{x}')$. Therefore, the corresponding utility of the defender for playing $\mathbf{x}'$ will be equal to $-\Big[x'_j P^a_j + (1 - x'_j)R^a_{max}(j)\Big]$. Since $R^a_j < R^a_{max}(j)$ and $x'_j \geq x^*_j$ (Observation 2), the following inequality holds true:

$$-\left[x'_j P^a_j + (1 - x'_j)R^a_{max}(j)\right] + \left[(x^*_j P^a_j + (1 - x^*_j)R^a_{max}(j)\right] \tag{8.34}$$

$$= -\left[x'_j P^a_j + (1 - x'_j)R^a_j\right] + \left[(x^*_j P^a_j + (1 - x^*_j)R^a_j\right] + \left[x'_j - x^*_j\right]\left[R^a_{max}(j) - R^a_j\right] \tag{8.35}$$

$$\geq -\left[x'_j P^a_j + (1 - x'_j)R^a_j\right] + \left[(x^*_j P^a_j + (1 - x^*_j)R^a_j\right]. \tag{8.36}$$

This inequality indicates that the defender's regret w.r.t $R^a_{max}(j)$ (the LHS of the inequality) is no less than w.r.t R^a_j (the RHS of the inequality). Therefore, $R^a_{max}(j)$ is an optimal solution of the attacker's reward at target j for (8.29–8.31). □

Based on the Proposition 9 & 10 and the Observation 1, the j^{th} sub-max regret (8.29–8.31) is simplified to the following optimization problem:

$$\max_{\mathbf{x}' \in \mathbf{X}, P^a_j, v} v + (x^*_j P^a_j + (1 - x^*_j) R^a_{max}(j)) \tag{8.37}$$

$$\text{s.t. } v = -\left[x'_j P^a_j + (1 - x'_j) R^a_{max}(j)\right] \tag{8.38}$$

$$v \leq -\left[x'_t P^a_{min}(t) + (1 - x'_t) R^a_{min}(t)\right], \forall t \neq j \tag{8.39}$$

$$P^a_{max}(j) \geq P^a_j \geq \max\left\{P^a_{min}(j), \frac{C - (1 - x^*_j) R^a_{max}(j)}{x^*_j}\right\} \tag{8.40}$$

where $C = \max_{t \neq j} x^*_t P^a_{min}(t) + (1 - x^*_t) R^a_{min}(t)$ is a constant. In addition, constraints (8.38–8.39) refer to constraint (8.30) (where constraint (8.38) is a result of Observation 1) and constraints (8.40) is equivalent to constraint (8.31). The only remaining non-convex term is $x'_j P^a_j$ in constraint (8.38). I then alleviate the computational cost incurred based on Theorem 4 which shows that if the attack set $\Gamma(\mathbf{x}')$ is known beforehand, I can convert (8.37–8.40) into a simple optimization problem which is straightforward to solve.

Theorem 4. *Given the attack set $\Gamma(\mathbf{x}')$, the j^{th} sub-max regret problem (8.37–8.40) can be represented as the following optimization problem on the variable v only:*

$$\max_v v + \frac{av + b}{cv + d} \tag{8.41}$$

$$\textit{s.t. } v \in [l_v, u_v]. \tag{8.42}$$

where v is the defender utility for playing $\mathbf{x}'$ in (8.37–8.40).

Proof. According to constraint (8.39), we obtain $x'_t = \frac{R^a_{min}(t) + v}{R^a_{min}(t) - P^a_{min}(t)}$ for all $t \neq j, t \in \Gamma(\mathbf{x}')$ and $x'_t = 0$ for all $t \notin \Gamma(\mathbf{x}')$. Since $\sum_t x'_t \leq R$ where R is the number of defender resources, the defender's coverage at target j must satisfy the following upper bound constraint: $x'_j \leq R - \sum_{t \in \Gamma(\mathbf{x}'), t \neq j} \frac{R^a_{min}(t) + v}{R^a_{min}(t) - P^a_{min}(t)}$ in which the RHS is in the form of $cv + d$ where c and d are constants. If $cv + d \geq 1$ or $cv + d = 0$, the problem becomes trivial since $x'_j = 1$ and $x'_j = 0$ respectively will be the optimal solution of x'_j. Therefore, we only consider when $0 \leq cv + d \leq 1$.

Since v is maximized when x'_j is maximized according to constraint (8.38), $x'_j = cv + d$ is thus the optimal solution of x'_j. Then the constraint (8.38) is equivalent to the following equality:

$$v = -\left[(cv+d)P^a_j + (1-cv-d)R^a_{max}(j)\right] \tag{8.43}$$

$$\iff P^a_j = \frac{-v-(1-cv-d)R^a_{max}(j)}{cv+d} \tag{8.44}$$

By replacing P^a_j by the RHS of (8.44) to the objective of the j^{th} sub-max regret problem in (8.37), we obtain the new objective which is in the form of $v + \frac{av+b}{cv+d}$ where (a, b) are constants. Finally, according to the constraint (8.40) and the allocation constraints $0 \leq x_t \leq 1$ for all t, we obtain the lower bound and upper bound on v, (l_v, u_v). □

The constants (a, b, c, d, l_v, u_v) are determined based on the attack set $\Gamma(\mathbf{x}')$, the attacker's payoffs $\{R^a_{min}(t), P^a_{min}(t)\}_{t\neq j}$ and $R^a_{max}(j)$, and the number of the defender resources R. Here, the total number of possible attack sets $\Gamma(\mathbf{x}')$ is maximally T sets according to the property that $R^a_t > R^a_{t'}$ for all $t \in \Gamma(\mathbf{x}')$ and $t' \notin \Gamma(\mathbf{x}')$ (Kiekintveld et al., 2009). Therefore, I can iterate over all these possible attack sets and solve the corresponding optimization problems in (8.41–8.42). The optimal solution of each sub-max regret problem (8.37–8.40) will be the maximum over optimal solutions of (8.41–8.42). The final optimal solution of the max regret problem (8.27–8.28) will be the maximum over optimal solutions of all these sub-max regret problems.

In summary, I provide the M.ARROW-Perfect algorithm to exactly compute max regret of playing the strategy $\mathbf{x}^*$ against a perfectly rational attacker in zero-sum games by exploiting the insight of extreme points of the uncertainty intervals as Ill as attack sets. Furthermore, I provide Theorem 5 showing that the computational complexity of solving max regret is polynomial.

Theorem 5. *M.ARROW-Perfect provides an optimal solution for computing max regret against a perfectly rational attacker in $O(T^3)$ time.*

Proof. Theorem 4 shows that if the attack set $\Gamma(\mathbf{x}')$ is given, then each sub-max regret problem can be converted into the maximization problem (8.41–8.42) which can be optimally solved in $O(1)$ time since its optimal solution will be at either the lower/upper bounds (l_v, u_v) or at the points where the derivative of (8.41) is zero. In addition, computing the constants (a, b, c, d, l_v, u_v) will take $O(T)$ time where T is the number of targets. Therefore, solving each sub-max regret given the attack set will be $O(T)$ time.

Algorithm 7: Elicitation process

Input: budget: B, regret barrier: δ, uncertainty intervals: $\mathbf{I}$;
Initialize regret $r = +\infty$, cost $c = 0$;
while $c < B$ *and* $r > \delta$ **do**
    $(r, \mathbf{x}^*, (\mathbf{x}'^{,*}, \mathbf{R}^{\mathbf{a},*}, \mathbf{P}^{\mathbf{a},*})) =$ *ARROW*$(\mathbf{I})$;
    $\mathbf{P} =$ *calculatePath*$(\mathbf{x}^*, (\mathbf{x}'^{,*}, \mathbf{R}^{\mathbf{a},*}, \mathbf{P}^{\mathbf{a},*}))$;
    $\mathbf{I} =$ *collectInformationUAV*$(\mathbf{P})$; $c =$ *updateCost*$(\mathbf{P})$;
end
return $(r, \mathbf{x}^*)$;

Furthermore, the total number of possible attack sets is maximally T sets according to the property that $R_t^a > R_{t'}^a$ for all $t \in \Gamma(\mathbf{x}')$ and $t' \notin \Gamma(\mathbf{x}')$. Hence, by iterating over all possible attack sets and solving the corresponding maximization problem (8.41–8.42), we can compute the optimal solution of each sub-max regret (8.37–8.40) in $O(T^2)$ time.

Finally, since there are T sub-max regret problems corresponding to which target is attacked given the defender strategy returned by R.ARROW-Perfect $\mathbf{x}^*$, the max regret problem (8.27–8.28) is optimally solved in $O(T^3)$ time as the maximum over all T sub-max regret problems (8.37–8.40). □

8.6 UAV Planning for Payoff Elicitation (PE)

my final contribution is to provide PE heuristics to select the best UAV path to reduce uncertainty in payoffs, given any adversary behavioral model. Despite the limited availability of mobile sensors in conservation areas (many of them being in developing countries), these UAVs may still be used to collect accurate imagery of these areas periodically, e.g., every six months to reduce payoff uncertainty. Since the UAV availability is limited, it is important to determine the best UAV paths such that reducing payoff uncertainty at targets on these paths could help reducing the defender's regret the most. While a UAV visits multiple targets to collect data, planning an optimal path (which considers all possible outcomes of reducing uncertainty) is computationally expensive. Thus, I introduce the *current solution*-based algorithm which evaluates a UAV path based solely on the $\mathrm{MMR_b}$ solution given current intervals.[4]

I first present a general elicitation process for UAV planning (Algorithm 7). The input includes the defender's initial budget B (e.g., limited time availability of UAVs), the regret barrier

[4] A similar idea was introduced in (Boutilier et al., 2006) although in a very different domain without UAV paths.

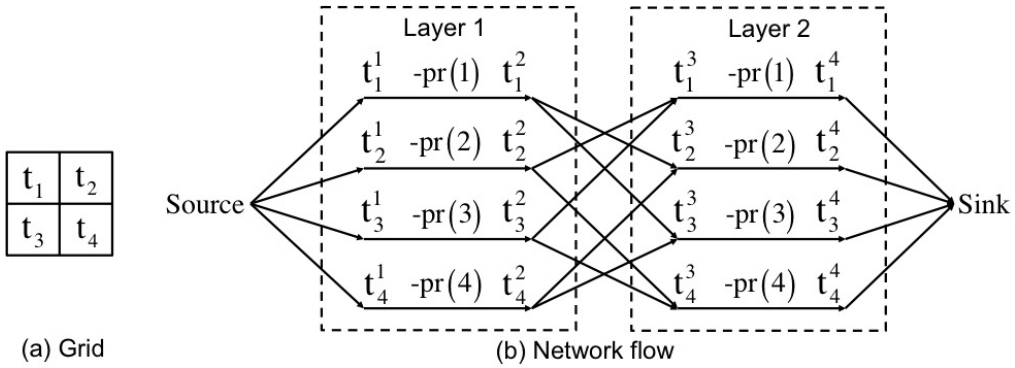


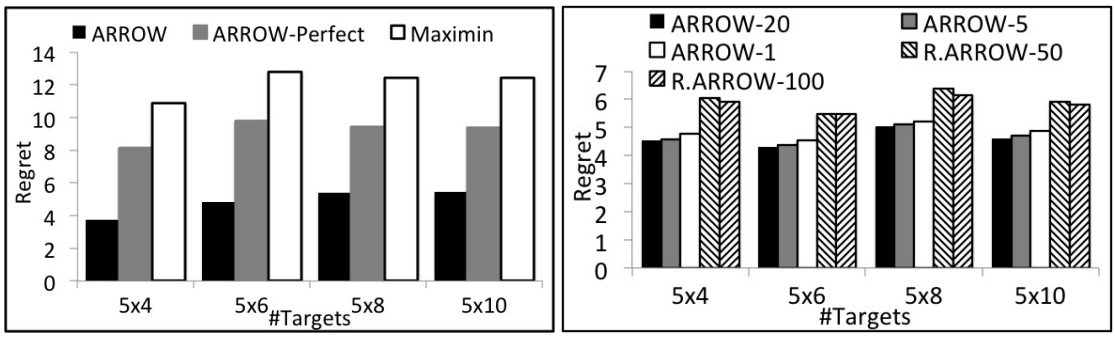
Figure 8.2: Min Cost Network Flow

δ which indicates how much regret (utility loss) the defender is willing to sacrifice, and the uncertainty intervals $\mathbf{I}$. The elicitation process consists of multiple rounds of flying a UAV and stops when the UAV cost exceeds B or the defender's regret is less than δ. At each round, ARROW is applied to compute the optimal $\mathrm{MMR_b}$ solution given current $\mathbf{I}$; ARROW then outputs the regret r, the optimal strategy $\mathbf{x}^*$, and the corresponding most unfavorable strategy and payoffs $(\mathbf{x}'^{,*}, \mathbf{R}^{\mathbf{a},*}, \mathbf{P}^{\mathbf{a},*})$ which provide the defender's max regret (line 4). Then the best UAV path is selected based on these outputs (line 5). Finally, the defender controls the UAV to collect data at targets on that path to obtain new intervals and then updates the UAV flying cost (line 6).

The key aspects of Algorithm 7 are in lines 4 and 5 where the $\mathrm{MMR_b}$ solution is computed by ARROW and the *current solution* heuristic is used to determine the best UAV path. In this heuristic, the *preference value* of a target t, denoted $pr(t)$, is measured as the distance in the defender utility betIen $\mathbf{x}^*$ and the most unfavorable strategy $\mathbf{x}'^{,*}$ against attacker payoffs $(\mathbf{R}^{\mathbf{a},*}, \mathbf{P}^{\mathbf{a},*})$ at that target, which can be computed as follows: $pr(t) = \hat{\mathrm{q}}_{\mathrm{t}}(\mathbf{x}^*, \mathbf{R}^{\mathbf{a},*}, \mathbf{P}^{\mathbf{a},*})\mathrm{U}^{\mathrm{d}}_{\mathrm{t}}(\mathbf{x}^*, \mathbf{R}^{\mathbf{d}}, \mathbf{P}^{\mathbf{d}}) - \hat{\mathrm{q}}_{\mathrm{t}}(\mathbf{x}'^{,*}, \mathbf{R}^{\mathbf{a},*}, \mathbf{P}^{\mathbf{a},*})\mathrm{U}^{\mathrm{d}}_{\mathrm{t}}(\mathbf{x}'^{,*}, \mathbf{R}^{\mathbf{d}}, \mathbf{P}^{\mathbf{d}})$ where $\mathbf{R}^{\mathbf{d}} = -\mathbf{P}^{\mathbf{a},*}$ and $\mathbf{P}^{\mathbf{d}} = -\mathbf{R}^{\mathbf{a},*}$. Intuitively, targets with higher preference values play a more important role in reducing the defender's regret. I use the sum of preference values of targets to determine the best UAV path based on the two heuristics:

Greedy heuristic: The chosen path consists of targets which are iteratively selected with the maximum pr value and then the best neighboring target.

MCNF heuristic: I represent this problem as a Min Cost Network Flow (MCNF) where the cost of choosing a target t is $-pr(t)$. For example, there is a grid of fmy cells (t_1, t_2, t_3, t_4) (Figure 8.2(a)) where each cell is associated with its preference value, namely $(pr(1), pr(2), pr(3), pr(4))$. Suppose that a UAV covers a path of two cells every time it flies



(a) ARROW regret

(b) Influence of parameters

Figure 8.3: Solution quality of ARROW

and its entry locations (where the UAV takes off or land) can be at any cell. The MCNF for UAV planning is shown in Figure 8.2(b) which has two layers where each cell t_i has fmy copies $(t_i^1, t_i^2, t_i^3, t_i^4)$ with edge costs $c(t_i^1, t_i^2) = c(t_i^3, t_i^4) = -pr(i)$. The connectivity betIen these two layers corresponds to the grid connectivity. There are *Smyce* and *Sink* nodes which determine the UAV entry locations. The edge costs betIen the layers and betIen the *Smyce* or *Sink* to the layers are set to zero.

8.7 Experimental Results

I use CPLEX for my algorithms and Fmincon of MATLAB on a 2.3 GHz/4 GB RAM machine. *Key comparison results are statistically significant under bootstrap-t ($\alpha = 0.05$) (Wilcox, 2002).* More results are in the Online Appendix G.

8.7.1 Synthetic Data

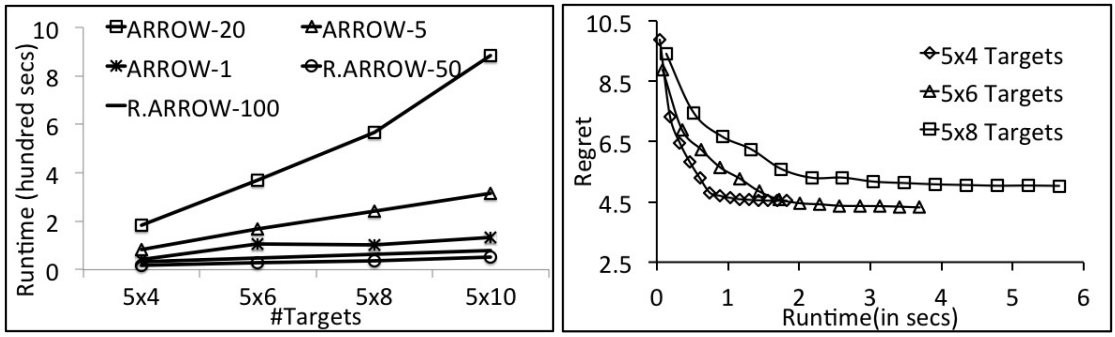
I first conduct experiments using synthetic data to simulate a wildlife protection area. The area is divided into a grid where each cell is a target, and I create different payoff structures for these cells. Each data point in my results is averaged over *40 payoff structures* randomly generated by GAMUT (Nudelman et al., 2004). The attacker reward/defender penalty refers to the animal density while the attacker penalty/defender reward refers to, for example, the amount of snares that are confiscated by the defender (Yang et al., 2014). Here, the defender's regret indicates the animal loss and thus can be used as a measure for the defender's patrolling effectiveness. Upper

and loIr bounds for payoff intervals are generated randomly from [-14, -1] for penalties and [1, 14] for rewards with an interval size of 4.0.

Solution Quality of ARROW. The results are shown in Figure 8.3 where the x-axis is the grid size (number of targets) and the y-axis is the defender's max regret. First, I demonstrate the importance of handling the attacker's bounded rationality in ARROW by comparing solution quality (in terms of the defender's regret) of ARROW with ARROW-Perfect and Maximin. Figure 8.3(a) shows that the defender's regret significantly increases when playing ARROW-Perfect and Maximin strategies compared to playing ARROW strategies, which demonstrates the importance of behavioral MMR.

Second, I examine how ARROW's parameters influence the $\mathrm{MMR_b}$ solution quality; which I show later affects its runtime-solution quality tradeoff. I examine if the defender's regret significantly increases if (i) the number of starting points in M.ARROW decreases (i.e., ARROW with 20 (ARROW-20), 5 (ARROW-5) and 1 (ARROW-1) starting points for M.ARROW and 40 iterations to iteratively add 40 payoff samples into the set S), or (ii) when ARROW only uses R.ARROW (without M.ARROW) to solve relaxed $\mathrm{MMR_b}$ (i.e., R.ARROW with 50 (R.ARROW-50) and 100 (R.ARROW-100) uniformly random payoff samples). Figure 8.3(b) shows that the number of starting points in M.ARROW does not have a significant impact on solution quality. In particular, ARROW-1's solution quality is approximately the same as ARROW-20 after 40 iterations. This result shows that the shortcoming of local search in M.ARROW (where solution quality depends on the number of starting points) is compensated by a sufficient number (e.g., 40) of iterations in ARROW. Furthermore, as R.ARROW-50 and R.ARROW-100 only solve relaxed $\mathrm{MMR_b}$, they both lead to much higher regret. Thus, it is important to include M.ARROW in ARROW.

Runtime Performance of ARROW. Figure 8.4(a) shows the runtime of ARROW with different parameter settings. In all settings, ARROW's runtime linearly increases in the number of targets. Further, Figure 8.3(a) shows that ARROW-1 obtains approximately the same solution quality as ARROW-20 while running significantly faster (Figure 8.4(a)). This result shows that one starting point of M.ARROW might be adequate for solving $\mathrm{MMR_b}$ in considering the trade-off betIen runtime performance and solution quality. Figure 8.4(b) plots the trade-off betIen runtime and the

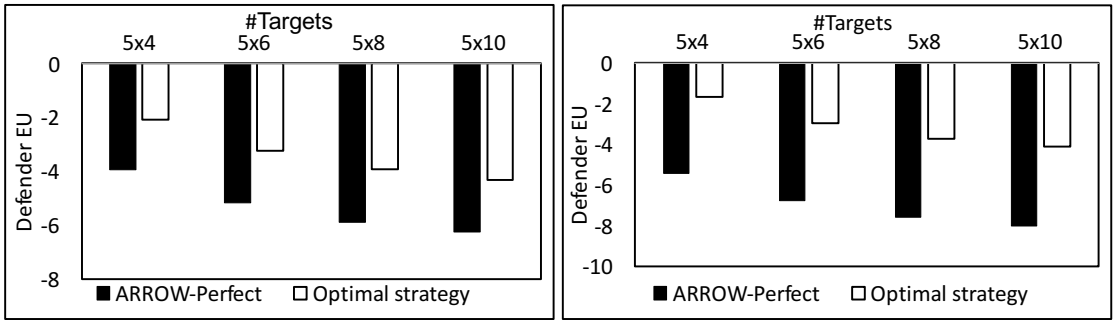


(a) Runtime vs #Targets (b) Runtime vs Regret

Figure 8.4: Runtime performance of ARROW

defender's regret in 40 iterations of ARROW-20 for 20-40 targets which shows a useful anytime profile.

Solution quality of ARROW-Perfect. Figure 8.5 shows the solution quality of ARROW-Perfect. The x-axis is the number of targets and the y-axis indicates the defender's expected utility for playing ARROW-Perfect compared with the optimal utility against the worst payoff instance. The result shows that the utility loss due to payoff uncertainty increases linearly with the interval size. Especially, the defender's utility loss compared with the optimal utility is approximately the same when the number of targets varies. This result clearly shows that ARROW-Perfect helps the defender in keeping a low loss in her utility given payoff uncertainty.



(a) Interval = 2 (b) Interval = 4

Figure 8.5: Solution quality of ARROW-Perfect

Runtime Performance of ARROW-Perfect. Figure 8.6 shows the runtime performance of ARROW-Perfect compared to ARROW and a non-linear solver (i.e., fmincon of Matlab) to compute MMR of the perfectly rational attacker case. While the runtime of both ARROW and non-linear solver increase quickly w.r.t the number of targets (e.g., it takes them approximately 20

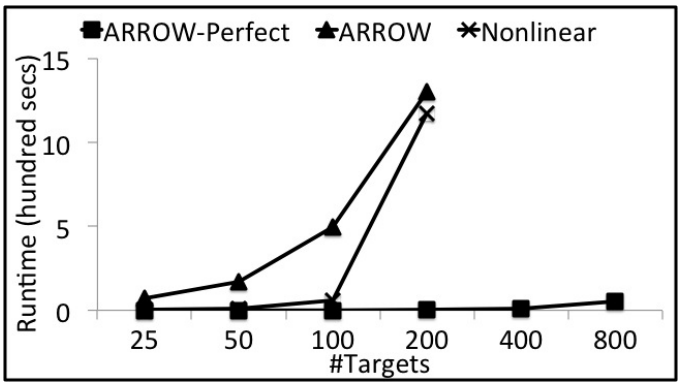
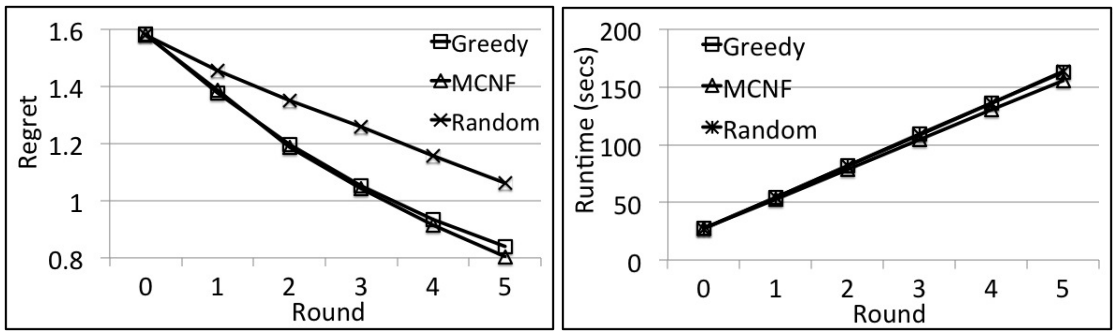


Figure 8.6: Runtime Performance of ARROW-Perfect

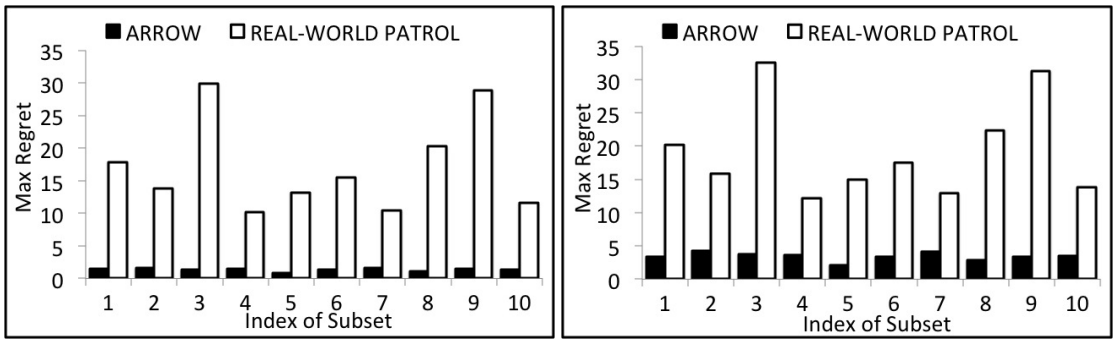
minutes to solve a 200-target game on average), ARROW-Perfect's runtime slightly increases and reaches 53 seconds to solve a 800-target game on average. This result shows that ARROW-Perfect is scalable for large security games.

Payoff Elicitation. I evaluate my PE strategies using synthetic data of 5×5-target (target = 2×2 km cell) games. The UAV path length is 3 cells and the budget for flying a UAV is set to 5 rounds of flying. I assume the uncertainty interval is reduced by half after each round. my purpose is to examine how the defender's regret is reduced over different rounds. The empirical results are shown in Figure 8.7 where the x-axis is the number of rounds and the y-axis is the regret obtained after each round (Figure 8.7(a)) or the accumulative runtime of the elicitation process over rounds (Figure 8.7(b)). I compare three heuristics: 1) randomly choosing a path (Random) 2) Greedy, and 3) MCNF. Figure 8.7 shows that the defender's regret is reduced significantly by using Greedy and MCNF in comparison with Random. As mentioned, the difference are statistically significant ($\alpha = 0.05$). Also, both Greedy and MCNF run as quickly as Random.



(a) Solution quality (b) Runtime performance

Figure 8.7: UAV planning: uncertainty reduction over rounds



(a) Small interval (b) Larger interval

Figure 8.8: Real world max regret comparison

8.7.2 Real-world Data

Lastly, I use my wildlife dataset from Uganda (Section 8.2) to analyze the effectiveness of past patrols conducted by rangers (in the wildlife park) compared with the patrol strategies generated by ARROW. I choose multiple subsets of 50 grid cells each, randomly sampled from the 2423 grid cells for my analysis. Before these wildlife areas Ire patrolled, there was uncertainty in the features values in those areas. I simulate these conditions faced by real world patrollers by introducing uncertainty intervals in the real-world payoffs. In my experiments, I impose uncertainty intervals on the animal density for each target, though two cases: a small and a large interval of sizes 5 and 10 respectively. Figures 8.8(a) and 8.8(b) show the comparison of the max regret achieved by ARROW and real world patrols for 10 such subsets, under the above mentioned cases of payoff uncertainty intervals. The x-axis refers to 10 different random subsets and the y-axis is the corresponding max regret. These figures clearly show that ARROW generates patrols with significantly less regret as compared to real-world patrols.

8.8 Summary

Whereas previous work in wildlife protection had assumed that there was an abundance of data in these domains, such data is not always available. To address such situations, I provide four main contributions: 1) for the first time, I compare key behavioral models, e.g., LensQR/QR on real-world data and show LensQR's usefulness in predicting adversary decisions; 2) I propose a novel algorithm, ARROW, to solve the $\mathrm{MMR_b}$ problem addressing both the attacker's bounded rationality and payoff uncertainty (when there is sufficient data to learn adversary behavioral

models); 3) I present a new scalable MMR-based algorithm, ARROW-Perfect, to address payoff uncertainty against a perfectly rational attacker (when learning behavioral models is infeasible), and 4) I introduce new PE strategies for mobile sensors, e.g., UAV to reduce payoff uncertainty.

Chapter 9

Conclusion and Future Work

Game-theoretic approaches to security, referred to *security games*, have been sucessfully applied to solving many real-world security problems, ranging from protecting critical national infrastructure such as airports, flights, and ports from terrorists to protecting environmental resources such as forests, wildlife, and fishery from smugglers, poachers, and illegal fisherman. In particular, Stackelberg security games, a leader-follower game-theoretic model, are at the heart of many real-world security applications, which assist security agencies to optimally randomize the allocation of *limited* security resources to protect important targets from being attacked by adversaries (e.g., terrorists). Standard SSGs require perfect knowledge about the game and unrealistic assumptions, such as: (i) the adversary is perfectly rational; (ii) the payoffs of both the defender and adversary are precisely estimated; and (iii) the defender's strategy is always executed perfectly and is always fully observed by the adversary. However, these assumptions are not ideal for solving real-world security problems since there exist a variety of uncertainties w.r.t the adversary's rationality, the players' payoff values, as well as the defender's execution and the adversary's observations. While adopting such assumptions is a reasonable start for developing the first generation of security game applications, it is critical to address uncertainties in security games in order to obtain effective patrolling strategies for the defender.

My research focuses on providing innovative techniques and significant advances for addressing the challenges of uncertainties in real-world security problems. In order to understand better the real-world security problems, in particular the wildlife protection problem, I co-organized a workshop on wildlife protection in collaboration with the World Wildlife Fund (WWF) held in Bandar Lampung, Indonesia in May, 2015. The goal of the workshop is to demonstrate the

(a) Workshop participants (b) Anti-poaching games

Figure 9.1: Workshop on wildlife protection held in Bandar Lampung, Indonesia

(a) Rhino camp (b) Collect information

Figure 9.2: Ranger patrols in Bandar Lampung, Indonesia

value of game-theoretic solutions for anti-poaching problems to security experts who protect wildlife. The workshop is held in Bandar Lampung, Indonesia which involves the participants from both the government and NGOs (Indonesian National Park Service, WWF, Wildlife Conservation Society, Indonesian Rhino Foundation, and Prosecution Officers from the Courts). These park rangers and law enforcement officers have a great deal of experience in wildlife protection and domain expertise in wildlife crime and protection. I'm fortunate to work directly with these domain experts, learning from their experience and expertise to improve my game-theoretic anti-poaching solutions, and especially going on the field with them to understand how patrols are conducted in the real world (Figure 9.1 and Figure 9.2). To that end, my thesis has the following five key contributions.

9.1 Contributions

- Stochastic model of adversary behavior—LensQR: This work demonstrates the importance of modeling human adversary decision making in SSGs. In particular, I introduce a new behavioral model called LensQR which is a novel integration of the *Lens utility function* with the Quantal Response model. Through extensive experiments, I provided the following contributions: (i) I show that my LensBRQR algorithm, which involves the new LensQR behavioral model, significantly outperforms both MATCH and its improved versions; (ii) I am the first to present experimental results with security intelligence experts, and find that even though the experts are more rational than AMT workers, LensBRQR performs better than its competition against the experts; (iii) I show the advantage of LensBRQR in a new game setting and demonstrate that additional data can further boost the performance of LensBRQR over MATCH. Finally, I show that LensQR is the best model compared to existing behavioral models in predicting poachers' behavior in wildlife protection.

- Sophisticated model of poacher behavior in wildlife protection—LensQR-Poacher: I introduce a new behavioral models of poachers, LensQR-Poacher, which is integrated in my new predictive anti-poaching tool, CAPTURE. The LensQR-Poacher model provides a significant advance over the state-of-the-art in modeling poachers in security games (Fang et al., 2016) and in conservation biology (Hofer et al., 2000; Critchlow et al., 2015) via 1) addressing the challenge of imperfect observations of the rangers; 2) incorporating temporal effects on the poachers' behaviors; and 3) not requiring a known number of attackers. I provide two new heuristics: parameter separation and target abstraction to reduce the computational complexity of learning the model parameters. Furthermore, CAPTURE incorporates a new planning algorithm to generate optimal patrolling strategies for the rangers, taking into account the new complex poacher model. Finally, this application presents an evaluation on the largest sample of real-world data in the security games literature, i.e., over 12-years of data of attacker defender interactions in QENP. The experimental results demonstrate the superiority of my model compared to other existing models. CAPTURE will be tested in QENP in early 2016.

- Robust maximin algorithm—URAC, ORAC, and GMM: I present three robust algorithms to handle multiple uncertainties in security games that maximize the defender's utility against the worst-case uncertainty. URAC is a unified robust algorithm that addresses all types of uncertainties. The first novel idea of URAC is to reduce the number of dimensions in the uncertainty space to reduce the complexity of finding the worst-case scenario for the defender due to uncertainties. URAC then follows a divide-and-conquer method to decouple the dependency of the adversary's strategies on the defender's strategies. Essentially, URAC divides the problem into sub-maximin problems; each sub-problem is associated with a subset of the defender's strategies and a uncertainty subset of the adversary's strategies which is independent from the defender's strategies. URAC then solves these sub-maximin problems using a standard optimization approach and combines the resulting sub-optimal solutions to find the global optimal solution. ORAC and GMM, on the other hand, are approximate robust scalable algorithms which focus on subsets of uncertainties, exploiting intrinsic properties of the attacker's rationality. Finally, I show through my experiments that my algorithms improve runtime performance and/or solution quality.

- Regret-based algorithm—MIRAGE: Despite significant applications of SSGs for protecting major critical infrastructure, research on *robustness* in SSGs has, to date, focused only on one concept, maximin over interval uncertainty of payoffs. I have proposed the use of MMR as a decision criterion for payoff-uncertain SSGs and presented an efficient algorithm, MIRAGE, for computing MMR for such games. The key idea of MIRAGE is to use incremental payoff generation; it starts by solving a relaxed minimax regret problem given a small set of payoff samples. Then new payoff samples are iteratively generated and added into the current set of samples until the optimal solution is obtained. I then introduce two new algorithms, bCISM and ALARMS, to solve the relaxed minimax regret and the max regret probblems. In particular, bCISM follows a two-level decomposition technique which decomposes relaxed minimax regret into simpler sub-problems based on individual payoff samples and targets the adversary can attack. ALARMS focuses on exploiting extreme points in the uncertainty space to compute max regret. Furthermore, I have addressed, for the first time, the challenge of preference elicitation in SSGs, providing novel regret-based

solution strategies. Experimental results validate the effectiveness of my approaches w.r.t. both computational and informational efficiency.

- Regret-based algorithm for wildlife protection—ARROW and ARROW-Perfect: Whereas previous work in wildlife protection had assumed that there was an abundance of data in these domains, such data is not always available. To address such situations, I propose a novel algorithm, ARROW, to solve the $\mathrm{MMR_b}$ problem addressing both the attacker's bounded rationality and payoff uncertainty (when there is sufficient data to learn adversary behavioral models). Furthermore, I present a new scalable MMR-based algorithm, ARROW-Perfect, to address payoff uncertainty against a perfectly rational attacker (when learning behavioral models is infeasible). Both ARROW and ARROW-Perfect follow the incremental payoff generation approach to solve minimax regret. ARROW applies piecewise linear approximation to linearize the non-convex components of the behavioral model of the poachers and represents the relaxed minimax regret problem as a MILP. ARROW-Perfect exploits extreme points in the uncertainty space to solve both relaxed minimax regret and max regret in polynomial time. Finally, I introduce new PE strategies for mobile sensors, e.g., UAVs, to reduce payoff uncertainty. I conduct extensive experiments, including evaluations of ARROW based on data from a wildlife park.

9.2 Future Work

In this thesis, I have shown how game-theoretic approaches (i.e., security games) can be applied to solving real-world security problems, with the focus on addressing uncertainties in two specific security domains: 1) infrastructure security and 2) wildlife protection. My future work will continue to focus on methodological advancements in human behavioral modeling and robust optimization techniques for addressing uncertainties. My work will support applications in not only infrastructure, green, and cyber security domains but also in other domains, such as public heath management and decision analysis, which include complex human decision making and multiple types of uncertainties. My research is inspired by large-scale interdisciplinary research challenges that arise in these domains, involving multi-disciplinary research areas including machine learning, multi-agent systems, and optimization.

One possible future direction is to further improve the behavioral model and patrol planning in CAPTURE for wildlife protection. I want to explore a richer class of behavioral models that explore spatial-temporal dependence in poachers' decision making as well as leverage all information w.r.t the poachers' activities. Currently, LensQR-Poacher does not take into account the spatial dependence of poachers' behavior. For example, if the poachers often go to certain areas to poach animals, they are likely to go to the neighborhood of that area as well. In fact, neighbor areas tend to share similarity in animal density and terrain structure, etc. Thus we can exploit this spatial property in reasoning about the poachers' behavior, further improving the prediction accuracy of the model. Moreover, LensQR-Poacher only considers a binary observation of the rangers, meaning that the model only examines whether the rangers observe any poaching signs or not at a certain location. As a result, LensQR does not fully exploit the observations of the rangers, which exhibit multiple levels of poaching signs at every location within the park. Furthermore, in planning patrols for the rangers, CAPTURE mainly focuses on generating coverage probabilities over grid cells. While this is a reasonable start, it is critical for CAPTURE to consider the terrain structure of the park in generating patrols, including elevation change and the habitat of the area, since it is infeasible for the rangers to walk through high moutains or rivers. While (Fang et al., 2016) presents the first solution for addressing this challenge by building a street map in which rangers can follow paths in the map, this work still adopts the simple LensQR model for predicting the poacher behavior. How to address the challenge of generating feasible and effective patrol routes for the rangers while taking into account the complex behavioral model of poachers, i.e., LensQR-Poacher is an interesting research direction for the future.

One other direction is the problem of exploration-exploitation tradeoff in adversary behavioral learning. Generally, we assume that our new behavioral model can capture accurately the adversary's behavior and then compute the optimal patrolling strategy for the defender relying on this assumption. However, in many real-world security domains, the available data for learning the adversary's behavior is insufficient, leading to an important issue of bias in modeling. For example, in wildlife protection, the rangers usually can only patrol a small portion of the area, leaving other areas unexplored. As a result, the model of the poachers' behavior can be learnt only based on the data collected by the rangers within that small portion. This issue may result

in an inaccurate model that cannot represent the poachers' behavior over the whole park. Therefore, it is important for us to go beyond the resulting model in terms of: 1) keep patrolling areas with a lot of poaching signs based on the resulting behavioral model (exploitation phase); and 2) patrolling other unexplored areas together with collecting poaching signs (exploration phase). I am interested in determining the level of exploitation and exploration given available data for learning.

Another possible direction relates to efficient computation of an optimal, robust, and feasible strategy for the defender. Currently, my robust algorithms mainly focus on generating an optimal robust marginal strategy for the defender without considering any patrolling constraints. As pointed out, in wildlife protection as well as other security domains, there are several constraints on the defender's feasible patrols (e.g., the rangers have to follow a set of patrol routes). As a result, the marginal strategy generated by my algoirthms may not be implementable. Therefore, in future work, I would like to extend my algorithms to handle such constraints. A key challenge is that these constraints could lead to a large number of the defender's pure strategies that need to be considered, making the problem of finding an optimal strategy for the defender computationally expensive. For example, in wildlife protection there is an exponential number of possible routes that the rangers can follow to patrol. Therefore, I want to address this scalability challenge together with the uncertainty challenge in security games.

To that end, I aim at building up a general unified game-theoretic framework for addressing uncertainties in security. This unified framework will allows us to easily adapt and extend the general game-theoretic methodologies presented to deal with uncertainties in any specific security domain. Specifically, in modeling the adversary's behavior, there will be multiple components (e.g., temporal and spatial components) integrated in predicting the adversary's decision making. The planning part will consist of different robust and efficient algorithms (based on maximin and minimax regret) to generate an optimal patrolling strategy for the defender. Integrated algorithms will be built to address different types of uncertainties, taking into account behavioral models of the adversary as well as the scalability challenge.

Bibliography

Abbasi, Y. D., Short, M., Sinha, A., Sintov, N., Zhang, C., & Tambe, M. (2015). Human adversaries in opportunistic crime security games: Evaluating competing bounded rationality models. In *Conference on Advances in Cognitive Systems*.

An, B., Brown, M., Vorobeychik, Y., & Tambe, M. (2013). Security games with surveillance cost and optimal timing of attack execution. In *AAMAS*.

An, B., Tambe, M., Ordonez, F., Shieh, E., & Kiekintveld, C. (2011). Refinement of strong stackelberg equilibria in security games. In *AAAI*.

Avrahami-Zilberbrand, D., & Kaminka, G. A. (2014). Keyhole adversarial plan recognition for recognition of suspicious and anomalous behavior. In *Plan, Activity, and Intent Recognition*, pp. 87–121.

Basilico, N., Gatti, N., & Amigoni, F. (2009). Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS*, pp. 57–64.

Basilico, N., & Gatti, N. (2011). Automated abstractions for patrolling security games.. In *AAAI*.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Boutilier, C. (2013). Computational decision support: Regret-based models for optimization and preference elicitation..

Boutilier, C., Patrascu, R., Poupart, P., & Schuurmans, D. (2006). Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artificial Intelligence*, *170*(8), 686–713.

Boutilier, C., Sandholm, T., & Shields, R. (2004). Eliciting bid taker non-price preferences in (combinatorial) auctions. In *AAAI*, pp. 204–211.

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, *30*(7), 1145–1159.

Braziunas, D., & Boutilier, C. (2010). Assessing regret-based preference elicitation with the utpref recommendation system. In *Proceedings of the 11th ACM conference on Electronic commerce*, pp. 219–228. ACM.

Breton, M., Alj, A., & Haurie, A. (1988). Sequential stackelberg equilibria in two-person games. *Journal of Optimization Theory and Applications*, *59*(1), 71–97.

Brown, G., Carlyle, M., Salmerón, J., & Wood, K. (2006). Defending critical infrastructure. *Interfaces*, *36*(6), 530–544.

Brunswik, E. (1952). *The conceptual framework of psychology*, Vol. 1. Univ of Chicago Pr.

Choi, S., Gale, D., & Kariv, S. (2012). Social learning in networks: a quantal response equilibrium analysis of experimental data. *Review of Economic Design*, *16*(2-3), 135–157.

Conitzer, V. (2012). Computing game-theoretic solutions and applications to security.. In *AAAI*.

Conitzer, V., & Sandholm, T. (2006). Computing the optimal strategy to commit to. In *ACM*.

Conlisk, J. (1996). Why bounded rationality?. *Journal of Economic Literature*, *34*(2), 669.

Critchlow, R., Plumptre, A., Driciru, M., Rwetsiba, A., Stokes, E., Tumwesigye, C., Wanyama, F., & Beale, C. (2015). Spatiotemporal trends of illegal activities from ranger-collected data in a ugandan national park. *Conservation Biology*, *29*(5), 14581470.

Dawes, R. M. (1979). The robust beauty of improper linear models in decision making. *American psychologist*, *34*(7), 571–582.

Dawes, R. M., Faust, D., & Meehl, P. E. (1989). Clinical versus actuarial judgment. *Science*, *243*(4899), 1668–1674.

De Bruin, J. S., Cocx, T. K., Kosters, W., Laros, J. F., Kok, J. N., et al. (2006). Data mining approaches to criminal career analysis. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pp. 171–177. IEEE.

De Farias, D. P., & Van Roy, B. (2004). On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of operations research*, *29*(3), 462–478.

Fang, F., Nguyen, T. H., Pickles, R., Lam, W. Y., Clements, G. R., An, B., Singh, A., Tambe, M., & Lemieux, A. (2016). Deploying paws: Field optimization of the protection assistant for wildlife security. In *IAAI*.

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

French, S. (1986). *Decision theory: an introduction to the mathematics of rationality*. Halsted Press.

Ganzfried, S., & Sandholm, T. (2011). Game theory-based opponent modeling in large imperfect-information games. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 533–540. International Foundation for Autonomous Agents and Multiagent Systems.

Greenwood, P., & Nikulin, M. (1996). A guard to chi-squared testing..

Grove, W. M., & Meehl, P. E. (1996). Comparative efficiency of informal (subjective, impressionistic) and formal (mechanical, algorithmic) prediction procedures: The clinical–statistical controversy.. *Psychology, Public Policy, and Law*, *2*(2), 293.

Haile, P. A., Hortacsu, A., & Kosenok, G. (2008). On the empirical content of quantal response equilibrium. *The American Economic Review*, *98*(1), 180–200.

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

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning 2nd edition..

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.

Hyafil, N., & Boutilier, C. (2004). Regret minimizing equilibria and mechanisms for games with strict type uncertainty. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pp. 268–277. AUAI Press.

Jain, M., Tsai, J., Pita, J., Kiekintveld, C., Rathi, S., Tambe, M., & Ordóñez, F. (2010). Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces*, *40*(4), 267–290.

Jiang, A. X., Nguyen, T. H., Tambe, M., & Procaccia, A. D. (2013). Monotonic maximin: A robust stackelberg solution against boundedly rational followers. In *GameSec*.

Johnson, M., Fang, F., & Tambe, M. (2012). Patrol strategies to maximize pristine forest area. In *AAAI*.

Kahneman, D., & Tversky, A. (1979). Prospect theory: An analysis of decision under risk. *Econometrica*, *47*(2), 263–292.

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 *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Karelaia, N., & Hogarth, R. M. (2008). Determinants of linear judgment: a meta-analysis of lens model studies.. *Psychological bulletin*, *134*(3), 404.

Kaufmann, E., & Athanasou, J. A. (2009). A meta-analysis of judgment achievement as defined by the lens model equation.. *Swiss Journal of Psychology/Schweizerische Zeitschrift für Psychologie/Revue Suisse de Psychologie*, *68*(2), 99.

Kiekintveld, C., Islam, T., & Kreinovich, V. (2013). Security games with interval uncertainty. In *AAMAS*.

Kiekintveld, C., Jain, M., Tsai, J., Pita, J., Ordez, F., & Tambe, M. (2009). Computing optimal randomized resource allocations for massive security games. In *The Eighth International Conference on Autonomous Agents and Multiagent Systems*.

Kiekintveld, C., Marecki, J., & Tambe, M. (2011). Approximation methods for infinite bayesian stackelberg games: modeling distributional payoff uncertainty.. In *AAMAS*.

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

Kouvelis, P., & Yu, G. (1997). *Robust discrete optimization and its applications*, Vol. 14. Springer.

Luce, R. D., & Raiffa, H. (2012). *Games and decisions: Introduction and critical survey*. Courier Corporation.

MacKenzie, D. I., Nichols, J. D., Lachman, G. B., Droege, S., Andrew Royle, J., & Langtimm, C. A. (2002). Estimating site occupancy rates when detection probabilities are less than one. *Ecology*, *83*(8), 2248–2255.

March, J. G. (1978). Bounded rationality, ambiguity, and the engineering of choice. *Bell Journal of Economics*, *9*(2), 587–608.

McFadden, D. (1972). Conditional logit analysis of qualitative choice behavior. Tech. rep..

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

Meehl, P. E. (1963). *Clinical versus statistical prediction: A theoretical analysis and a review of the evidence*. University of Minnesota Press.

Meng, X.-L., & Rubin, D. B. (1993). Maximum likelihood estimation via the ecm algorithm: A general framework. *Biometrika*, *80*(2), 267–278.

Montesh, M. (2013). Rhino poaching: A new form of organised crime1. Tech. rep., University of South Africa.

Nath, S. V. (2006). Crime pattern detection using data mining. In *Web Intelligence and Intelligent Agent Technology Workshops, 2006. WI-IAT 2006 Workshops. 2006 IEEE/WIC/ACM International Conference on*, pp. 41–44. IEEE.

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 *GameSec*.

Nguyen, T. H., Jiang, A., & Tambe, M. (2014). Stop the compartmentalization: Unied robust algorithms for handling uncertainties in security games. In *AAMAS*.

Nguyen, T. H., Sinha, A., Gholami, S., Plumptre, A., Joppa, L., Tambe, M., Driciru, M., Wanyama, F., Rwetsiba, A., Critchlow, R., & Beale, C. (2016). Capture: A new predictive anti-poaching tool for wildlife protection. In *Under Review*.

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. (2013). Analyzing the effectiveness of adversary modeling in security games. In *AAAI*.

Nudelman, E., Wortman, J., Shoham, Y., & Leyton-Brown, K. (2004). Run the gamut: A comprehensive approach to evaluating game-theoretic algorithms. In *AAMAS*, pp. 880–887.

Oatley, G., Ewart, B., & Zeleznikow, J. (2006). Decision support systems for police: Lessons from the application of data mining techniques to "soft" forensic evidence. *Artificial Intelligence and Law*, *14*(1-2), 35–100.

on Foreign Affairs, U. H. C. (2015). Poaching and terrorism: A national security challenge (serial no. 114-25). Washington:US Government Publishing Office.

Paruchuri, P., Pearce, J., Marecki, J., Tambe, M., Ordonez, F., & Kraus, S. (2008). Playing games for security: an efficient exact algorithm for solving bayesian stackelberg games. In *AAMAS*, pp. 895–902.

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

Pita, J., John, R., Maheswaran, R., Tambe, M., & Kraus, S. (2012). A robust approach to addressing human adversaries in security games. In *ECAI*, pp. 660–665.

Pita, J., Jain, M., Ordonez, F., Tambe, M., Kraus, S., & Magori-Cohen, R. (2009). Effective solutions for real-world stackelberg games: When agents must deal with human uncertainties. In *AAMAS*.

Quiggin, J. (1990). Stochastic dominance in regret theory. *The Review of Economic Studies*, *57*(3), 503–511.

Regan, K., & Boutilier, C. (2009). Regret-based reward elicitation for markov decision processes. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 444–451. AUAI Press.

Renou, L., & Schlag, K. H. (2010). Minimax regret and strategic uncertainty. *Journal of Economic Theory*, *145*(1), 264–286.

Salo, A. A., & Hamalainen, R. P. (2001). Preference ratios in multiattribute evaluation (prime)-elicitation and decision procedures under incomplete information. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, *31*(6), 533–545.

Sandholm, T., & Singh, S. (2012). Lossy stochastic game abstraction with bounds. In *EC*, pp. 880–897. ACM.

Savage, L. (1972). *The foundations of statistics*. DoverPublications. com.

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

Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., & Meyer, G. (2012). Protect: A deployed game theoretic system to protect the ports of the united states. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Sinha, A., Kar, D., & Tambe, M. (2016). Learning adversary behavior in security games: A pac model perspective. In *AAMAS*.

Southey, F., Bowling, M. P., Larson, B., Piccione, C., Burch, N., Billings, D., & Rayner, C. (2012). Bayes' bluff: Opponent modelling in poker. *CoRR*, *abs/1207.1411*.

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

Tsai, J., Kiekintveld, C., Ordonez, F., Tambe, M., & Rathi, S. (2009). Iris-a tool for strategic security allocation in transportation networks. In *AAMAS*.

Von Stengel, B., & Zamir, S. (2004). Leadership with commitment to mixed strategies. Tech. rep..

Wilcox, R. (2002). *Applying contemporary statistical techniques*. Academic Press.

Wright, J. R., & Leyton-Brown, K. (2014). Level-0 meta-models for predicting human behavior in games. In *ACM-EC*, pp. 857–874.

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

Yang, R., Kiekintveld, C., Ordonez, F., Tambe, M., & John, R. (2011). Improving resource allocation strategy against human adversaries in security games. In *IJCAI*.

Yang, R., Ordonez, F., & Tambe, M. (2012). Computing optimal strategy against quantal response in security games.. AAMAS.

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

Yin, Z., Jain, M., Tambe, M., & Ordonez, F. (2011). Risk-averse strategies for security games with execution and observational uncertainty. In *AAAI*.

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

Yin, Z., & Tambe, M. (2012a). A unified method for handling discrete and continuous uncertainty in bayesian stackelberg games. In *AAMAS*.

Yin, Z., & Tambe, M. (2012b). A unified method for handling discrete and continuous uncertainty in bayesian stackelberg games. In *AAMAS*.

Zhang, C., Sinha, A., & Tambe, M. (2015). Keeping pace with criminals: Designing patrol allocation against adaptive opportunistic criminals. In *AAMAS*.