Handling Attacker's Preference in Security Domains: Robust Optimization and Learning Approaches

by

Yundi Qian

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)

June 2016

Acknowledgements

First and foremost, I would like to thank my advisor, Prof. Milind Tambe, for all of his support and encouragement throughout my Ph.D. When I first joined the Teamcore research group, little did I know about what it means to do research. Milind, with his endless patience and faithful encouragement, guided me through each step to do meaningful research. I can never forget all the moments when Milind stays up late to the last minute together with me before conference deadlines, challenges me to push every idea to its limit and iterates over paper and talks numerous times to reach their excellence. In addition, I really appreciate the luxurious freedom Milind gives me to explore the research domains I am interested in, with which I really enjoyed the past four-years research experience. Milind's passion about research and commitment to students make him the best academic advisor I could ask for. I learned so much from him, not only about research but also how to be a good mentor with care and patience.

I would also like to thank other members of my thesis committee for providing valuable feedback to my research and pushing me to think about it in another level: Aram Galstyan, Jonathan Gratch, Maged Dessouky and Yilmaz Kocer.

I would also like to thank the many excellent researchers that I have had the privilege to work with over the years. This list includes Jason Tsai, Yevgeniy Vorobeychik,

Christopher Kiekintveld, Chao Zhang, Bhaskar Krishnamachari, Victor Bucarey, Ayan Mukhopadhyay, Arunesh Sinha, William B. Haskell, Albert Xin Jiang, Ariel Procaccia and Nisarg Shah. Special thanks to Jason Tsai for guiding me when I initially joined the Teamcore group, Albert Xin Jiang for his very detailed guidance when I was writing my first paper, William B. Haskell for polishing my paper writing numerous times, Bhaskar Krishnamachari for his valuable insights for my work, Chao Zhang for always being there when I want to discuss my research.

I would like to thank the entire Teamcore family: Albert Xin Jiang, Francesco Delle Fave, William Haskell, Arunesh Sinha, James Pita, Manish Jain, Jason Tsai, Jun-young Kwak, Zhengyu Yin, Rong Yang, Matthew Brown, Eric Shieh, Thanh Nguyen, Leandro Marcolino, Fei Fang, Chao Zhang, Debarun Kar, Benjamin Ford, Haifeng Xu, Amulya Yadav, Aaron Schlenker, Sara McCarthy, Yasi Abbasi, Shahrzad Gholami, Bryan Wilder, and Elizabeth Orrico. It is my great honor to share my Ph.D experience with all of you and I really enjoy the time that we work and play together. Special thanks to Albert Xin Jiang for his tremendous help in doing research, Jun-young Kwak and Fei Fang for being lovely officemates when we were at PHE 101, Eric Shieh for treating me with good food and pushing me to practice my English, Chao Zhang for being an awesome person to talk to.

Finally I would like to thank my friends and families, who gave me tremendous help for me to complete my Ph.D. I would like to thank my parents, for always supporting me, believing the best in me and always be there when I need their help. I would also like to thank my boyfriend Ji Yang for being a constant source of support and encouragement. This thesis would have never been possible without him.

Table of Contents

List Of Tables

List Of Figures

Abstract

Stackelberg security games (SSGs) are now established as a powerful tool in security domains. In order to compute the optimal strategy for the defender in SSG model, the defender needs to know the attacker's preferences over targets so that she can predict how the attacker would react under a certain defender strategy. Uncertainty over attacker preferences may cause the defender to suffer significant losses. Motivated by that, my thesis focuses on addressing uncertainty in attacker preferences using robust and learning approaches.

In security domains with one-shot attack, e.g., counter-terrorism domains, the defender is interested in robust approaches that can provide performance guarantee in the worst case. The first part of my thesis focuses on handling attacker's preference uncertainty with robust approaches in these domains. My work considers a new dimension of preference uncertainty that has not been taken into account in previous literatures: the risk preference uncertainty of the attacker, and propose an algorithm to efficiently compute defender's robust strategy against uncertain risk-aware attackers.

In security domains with repeated attacks, e.g., green security domain of protecting natural resources, the attacker "attacks" (illegally extracts natural resources) frequently, so it is possible for the defender to learn attacker's preference from their previous actions

and then to use this information to better plan her strategy. The second part of my thesis focuses on learning attacker's preferences in these domains. My thesis models the preferences from two different perspectives: (i) the preference is modeled as payoff and the defender learns the payoffs from attackers' previous actions; (ii) the preference is modeled as a markovian process and the defender learns the markovian process from attackers' previous actions.

Chapter 1

Introduction

Stackelberg security games (SSGs) are now established as a successful tool in the infrastructure security domain [20,42,62]. In this domain, the security forces (defender) deploy security resources to protect key infrastructures (targets) against potential terrorists (attackers). With limited resources available, it is usually impossible to protect all targets at all times. SSGs optimize the use of defender resources with the use of game-theoretic approaches. In SSG model, the defender acts first and commits to a mixed strategy while the attacker learns the mixed strategy after long-time surveillance and then chooses one target to attack [55].

The success of SSGs in the infrastructure security domains has inspired researchers' interest in applying game-theoretic models to other security domains with frequent interactions between defenders and attackers, e.g., wildlife protection [12, 58]. However, these two domains are different. In wildlife protection domain, attack (poaching) happens frequently so that it gives defenders (patrollers) the opportunity to learn attackers' (poachers') preferences from their previous actions and then to plan patrol strategies accordingly; while this learning opportunity does not arise in the counter-terrorism domain.

1.1 Problem Addressed

The computation of the optimal strategy for the defender requires the defender to know how the attacker views the importance of every target since it involves predicting the attacker's action under a certain defender strategy. If the defender is unable to predict the attacker's action correctly, she may suffer significant losses. Motivated by that, my thesis focuses on addressing uncertainty in attacker preferences over targets using robust and learning approaches.

The defender's uncertainty about how the attacker views the importance of every target may come from two different perspectives: (i) the uncertainty over attacker's payoffs, i.e., the defender is uncertain about the true payoffs of different targets for the attacker; (ii) the uncertainty over attacker's risk attitude, i.e., the attacker may not be risk-neutral and the defender is uncertain about the attacker's risk attitude.

In security domains with one-shot attack, e.g., counter-terrorism domains, the attack happens rarely so there is no chance for the defender to learn attacker's preference from their previous actions. Thus, the defender is interested in robust approaches that can provide performance guarantee in the worst case. The payoff uncertainty in SSGs has been addressed in previous literature [19] while the risk attitude uncertainty has not been addressed yet. Therefore, the first part of my thesis focuses on handling attacker's risk attitude uncertainty in SSGs with robust approaches.

In security domains with repeated attacks, e.g., green security domain of protecting natural resources, the attacker "attacks" (illegally extracts natural resources) frequently, so it is possible for the defender to learn attacker's preference from their previous actions

and then to use this information to better plan her strategy. Therefore, the second part of my thesis focuses on learning attacker's preferences and then planning accordingly in these domains. In this way, the learned preference is the preference in the attacker's mind, which takes both the payoff and risk attitude into account. My thesis models the preferences from two different perspectives:

1. The preference is modeled as payoff and the defender learns the payoffs from attackers' previous actions and then plan accordingly. However, this work is based on two key assumptions: (i) the attacker follows some certain behavioral patterns that are known to the defender; (ii) both the defender and the attacker can observe their opponent's activities at all targets. However, these two assumptions may not hold in some domains.

2. To relax these two assumptions, I model the preference as a markovian process that transits according to defender's strategies. The defender learns the markovian process from attackers' previous actions and then plans accordingly. This model needs no prior information about the attacker's behavioral patterns and is able to handle the exploration-exploitation tradeoff in these domains, which is caused by the fact that the defender is only able to observe the attack activities happening at protected targets.

1.2 Contributions

My contributions include addressing uncertainty in attackers' preference using robust and learning approaches. My first contribution develops an algorithm to efficiently compute

the robust strategy against risk-aware attackers in SSGs. My second contribution models the preference as payoffs and focuses on learning the payoffs and then planning accordingly in green security domains. My third contribution models the preference as markovian process that transits according to defender's strategies to handle the exploration-exploitation tradeoff in these domains.

1.2.1 Robust Strategy against Risk-aware Attackers in SSGs

The first part of my thesis [51] focuses on handling the uncertainty of attacker's risk preferences in security games. Previous work on game theory for SSGs emphasizes a risk neutral attacker that is trying to maximize his expected reward. However, extensive studies show that the attackers are in fact risk-aware, e.g., terrorist groups in counter-terrorism domains [44, 46, 48] are shown to be risk-averse. If the defender fails to take attacker's risk preference into consideration when designing strategies, she may suffer significant losses. Furthermore, risk awareness encompasses a wide range of behavior — so to say that attackers are risk-aware is not enough for the defender. In other words, the defender has uncertainty over the attacker's degree of risk awareness. To address this issue, the first part of my thesis computes a robust defender strategy that optimizes the worst case against risk-aware attackers with uncertainty in the degree of risk awareness [1], i.e., it provides a solution quality guarantee for the defender no matter how risk-aware the attacker is.

To develop the robust strategy, I firstly build a robust SSG framework against an attacker with uncertainty in level of risk awareness. Second, building on previous work

on SSGs in mixed-integer programs, I propose a novel mixed-integer bilinear programming problem (MIBLP), and find that it only finds locally optimal solutions. While the MIBLP formulation is also unable to scale up, it provides key intuition for my new algorithm. This new algorithm, BeRRA (**B**inary search based **R**obust algorithm against **R**isk-**A**ware attackers) is my third contribution, and it finds globally ϵ-optimal solutions by solving $\mathcal{O}(n \log(\frac{1}{\epsilon}) \log(\frac{1}{\delta}))$ linear feasibility problems. The key idea of the BeRRA algorithm is to reduce the problem from maximizing the reward with a given number of resources to minimizing the number of resources needed to achieve a given reward. This transformation allows BeRRA to scale up via the removal of the bilinear terms and integer variables as well as the utilization of key theoretical properties that prove correspondence of its potential "attack sets" [20] with that of the maximin strategy. Finally, I also show that the defender does not need to consider attacker's risk attitude in zero-sum games. The experimental results show the solution quality and runtime advantages of my robust model and BeRRA algorithm.

1.2.2 Learning Attacker's Preference — Payoff Modeling

The second part of my thesis [50] focuses on learning the attacker's payoffs in green security domains where there are frequent interactions between the defender and the attacker. In green security domains, the "defenders" (law enforcement agencies) try to protect these natural resources and "attackers" (criminals) seek to exploit them. In infrastructure security games, the attacker conducts extensive surveillance on the defender and executes a one-shot attack, while in green security domains, the attacker also observes the defender's strategy but carries out frequent illegal extractions. Therefore, there are

frequent interactions between the defender and the attacker, which gives the defender the opportunity to learn the attacker's payoffs by observing the attacker's actions. Motivated by this, the second part of my thesis develops the model and algorithm for the defender to learn target values from attacker's actions and then uses this information to better plan her strategy.

In this work, I model these interactions between the defender and the attacker as a repeated game. I then adopt a fixed model for the attacker's behavior and recast this repeated game as a partially observable Markov decision process (POMDP). However, my POMDP formulation has an exponential number of states, making current POMDP solvers like ZMDP [54] and APPL [25] infeasible in terms of computational cost. Silver and Veness [53] have proposed the POMCP algorithm which achieves a high level of performance in large POMDPs. It uses particle filtering to maintain an approximation of the belief state of the agent, and then uses Monte Carlo Tree Search (MCTS) for online planning. However, the particle filter is only an approximation of the belief state. By appealing to the special properties of my POMDP, I propose the GMOP algorithm (**G**ibbs sampling based **M**CTS **O**nline **P**lanning) which draws samples directly from the exact belief state using Gibbs sampling and then runs MCTS for online planning. My algorithm provides higher solution quality than the POMCP algorithm. Additionally, for a specific subclass of my game with an attacker who plays a best response against the defender's empirical distribution, and a uniform penalty of being seized across all targets, I provide an advanced sampling technique to speed up the GMOP algorithm along with a heuristic that trades off solution quality for lower computational cost. Moreover, I explore the case of continuous utilities where my original POMDP formulation becomes

a continuous-state POMDP, which is generally difficult to solve. However, the special properties in the specific subclass of game mentioned above make possible the extension of the GMOP algorithm to continuous utilities. Finally, I explore the more realistic scenario where the defender is not only uncertain about the distribution of resources, but also uncertain about the attacker's behavioral model. I address this challenge by extending my POMDP formulation and the GMOP algorithm.

1.2.3 Learning Attacker's Preference — Markovian Modeling

My second contribution [50] assumes that defenders have knowledge of all poaching activities throughout the wildlife protected area. Unfortunately, given vast geographic areas for wildlife protection, defenders do not have knowledge of poaching activities in areas they do not protect. Thus, defenders are faced with the exploration-exploitation tradeoff — whether to protect the targets that are already known to have a lot of poaching activities or to explore the targets that haven't been protected for a long time. My third contribution [52] aims to solve this exploration-exploitation tradeoff.

The exploration-exploitation tradeoff here is different from that in the non-Bayesian stochastic multi-armed bandit problem [4]. In stochastic multi-armed bandit problems, the rewards of every arm are random variables with a stationary unknown distribution. However, in this problem, patrol affects attack activities — more patrol is likely to decrease attack activities and less patrol is likely to increase attack activities. Thus, the random variable distribution is changing depending on player's choice — more selection (patrol) leads to lower reward (less attack activities) and less selection (patrol) leads to higher reward (more attack activities). On the other hand, adversarial multi-armed

bandit problem [5] is also not an appropriate model for this domain. In adversarial multi-armed bandit problems, the reward can arbitrarily change while the attack activities in this problem are unlikely to change rapidly in a short period. This makes the adversarial multi-armed bandit model inappropriate for this domain.

In reality, how patrol affects attack activities would be reasonably assumed to follow a consistent pattern that can be learned from historical data (defenders' historical observations). I model this pattern as a Markov process and provide the following contributions in this work. First, I formulate the problem into a restless multi-armed bandit (RMAB) model to handle the limited observability challenge — defenders do not have observations for arms they do not activate (targets they do not protect). Second, I propose an EM based learning algorithm to learn the RMAB model from defenders' historical observations. Third, I use the solution concept of Whittle index policy to solve the RMAB model to plan for defenders' patrol strategies. However, indexability is required for the existence of Whittle index, so I provide two sufficient conditions for indexability and an algorithm to numerically evaluate indexability. Fourth, I propose a binary search based algorithm to find the Whittle index policy efficiently.

1.3 Overview of Thesis

This thesis is organized in the following manner. Chapter 2 discusses the necessary background materials for the research presented in this thesis. Chapter 3 provides an overview of the relevant research. Chapter 4 discusses the algorithm to compute the robust strategy against risk-aware attackers. Chapter 5 presents the model to learn attackers' payoffs of

different targets and then use this information to better plan defenders' patrol strategies. Chapter 6 demonstrates the model where attacker's preference is modeled as Markovian process. Chapter 7 concludes this thesis and presents ideas for future work.

Chapter 2

Background

2.1 Stackelberg Security Games

An SSG [20,42,62] is a two-player game between a defender and an attacker. We consider the problem with n targets where $\mathbb{T} = \{1, 2, \ldots, n\}$ is the set of targets. The defender has a total number of m resources to allocate among these n targets to protect them from attack. The defender commits to a mixed strategy $\mathbf{c}$ to protect these targets, where $c_i \in [0, 1]$ is the probability that target i is protected. We have the resource constraint $\sum_{i \in \mathbb{T}} c_i \leq m$. The attacker observes the defender's strategy $\mathbf{c}$ and then chooses one target to attack. If the attacker attacks a protected target i, this attack is unsuccessful and the attacker receives utility $U_a^c(i)$ while the defender receives utility $U_d^c(i)$. If the attacker attacks an unprotected target i, this attack is successful and the attacker receives utility $U_a^u(i)$ while the defender receives utility $U_d^u(i)$. Necessarily, $U_d^u(i) < U_d^c(i)$ and $U_a^c(i) < U_a^u(i), \forall i \in \mathbb{T}$. If $U_d^u(i) + U_a^u(i) = 0$ and $U_d^c(i) + U_a^c(i) = 0, \forall i \in \mathbb{T}$, this SSG is a zero-sum game.

We define $U_a(i, \mathbf{c}) \triangleq c_i U_a^c(i) + (1 - c_i) U_a^u(i)$ to be the expected utility for the attacker when the defender's strategy is $\mathbf{c}$ and the attacker chooses to attack target i; similarly,

$U_d(i, \mathbf{c}) \triangleq c_i U_d^c(i) + (1 - c_i) U_d^u(i)$ is the expected utility for the defender. Given the defender strategy $\mathbf{c}$, the attacker would attack the target that maximizes his expected utility. When there are ties, the attacker is assumed to break ties in favor of the defender. Thus, a mixed integer linear program (MILP) can be formulated to compute the defender's optimal strategy, as is shown in Problem (2.1). Here, $\{q_i\}_{i \in \mathbb{T}}$ are auxiliary variables to represent if target i is chosen by the attacker, and M is a constant orders of magnitude larger than all target utilities. The solution $\mathbf{c}$ is called the Strong Stackelberg Equilibrium (SSE) strategy [11, 24, 55] of the game.

$$
\begin{aligned}
\max_{\mathbf{c}, \{q_i\}_{i \in \mathbb{T}}, v, d} \quad & v \\
s.t. \quad & 0 \le c_i \le 1, \forall i \in \mathbb{T} \qquad \sum_{i \in \mathbb{T}} c_i \le m \\
& q_i \in \{0, 1\}, \forall i \in \mathbb{T} \qquad \sum_{i \in \mathbb{T}} q_i = 1 \\
& v \le U_d(i, \mathbf{c}) + (1 - q_i) M, \forall i \in \mathbb{T} \\
& 0 \le d - U_a(i, \mathbf{c}) \le (1 - q_i) M, \forall i \in \mathbb{T}
\end{aligned}
\tag{2.1}
$$

2.2 POMDP

POMDP is a generalization of a Markov decision process (MDP) by assuming that the agent cannot directly observe the underlying state. Instead, the agent observes "observation", which reveals the underlying states via a probability distribution. Therefore, the agent maintains a probability distribution over the set of possible states based on its observations, and also plans its actions according to this distribution.

The POMDP framework can be used to model sequential decision processes in uncertain environments. At the beginning of each round, the agent has a probability distribution of the state it is current in, it then executes the optimal action under this distribution and gets the corresponding observation, it finally uses the observation to update the belief of its new state.

Formally, a POMDP is a 7-tuple $(S, A, T, R, \Omega, O, \gamma)$, where

- S is a set of states
- A is a set of actions
- T is a set of conditional transition probabilities between states
- $R : S \times A \to R$ is the reward function
- Ω is a set of observations
- O is a set of conditional observation probabilities
- $\gamma \in [0, 1]$ is the discount factor

For standard POMDP formulation, the belief update is:

$$b'(s') = \frac{P(o|s', a) \sum_{s \in \mathbf{S}} b(s) P(s'|s, a)}{P(o|b, a)} \tag{2.2}$$

where

$$P(o|b, a) = \sum_{s' \in \mathbf{S}} P(o|s', a) \sum_{s \in \mathbf{S}} b(s) P(s'|s, a)$$

POMDP can be solved by value iteration algorithm, while I will briefly present below: The value function is

$$V'(b) = \max_{a \in \mathbf{A}} \left(\sum_{s \in \mathbf{S}} b(s) R(s,a) + \beta \sum_{o \in \mathbf{O}} P(o|b,a) V(b_a^o) \right)$$

It can be broken up to simpler combinations of other value functions:

$$\begin{aligned} V'(b) &= \max_{a \in \mathbf{A}} V_a(b) \\ V_a(b) &= \sum_{o \in \mathbf{O}} V_a^o(b) \\ V_a^o(b) &= \frac{\sum_{s \in \mathbf{S}} b(s) R(s,a)}{|\mathbf{O}|} + \beta P(o|b,a) V(b_a^o) \end{aligned}$$

All the value functions can be represented as $V(b) = \max_{\alpha \in \mathbf{D}} b \cdot \alpha$ since the update process maintains this property, so we only need to update the set D when updating the value function. The set D is updated according to the following process:

$$\begin{aligned} D' &= \text{purge} \left(\bigcup_{a \in \mathbf{A}} D_a \right) \\ D_a &= \text{purge} \left(\bigoplus_{o \in \mathbf{O}} D_a^o \right) \\ D_a^o &= \text{purge} \left(\{ \tau(\alpha, a, o) | \alpha \in D \} \right) \end{aligned}$$

where $\tau(\alpha, a, o)$ is the $|D|$-vector given by

$$\tau(\alpha, a, o)(s) = (1/|\mathbf{O}|) R(s,a) + \beta \sum_{s' \in \mathbf{S}} \alpha(s') P(o|s', a) P(s'|s, a)$$

and purge($\cdot$) takes a set of vectors and reduces it to its unique minimum form (remove redundant vectors that are dominated by other vectors in the set). $\bigoplus$ represents the cross sum of two sets of vectors: $A \bigoplus B = \{\alpha + \beta | \alpha \in A, \beta \in B\}$.

The update of D' and D_a is intuitive, so I briefly explain the update of D_a^o here:

$$
\begin{aligned}
P(o|b,a)V(b_a^o) &= P(o|b,a) \max_{\alpha \in D} \sum_{s' \in \mathbf{S}} \alpha(s') P(s'|b,a,o) \\
&= P(o|b,a) \max_{\alpha \in D} \sum_{s' \in \mathbf{S}} \alpha(s') \frac{P(o|s',a) \sum_{s \in \mathbf{S}} b(s) P(s'|s,a)}{P(o|b,a)} \\
&= \max_{\alpha \in D} \sum_{s' \in \mathbf{S}} \alpha(s') P(o|s',a) \sum_{s \in \mathbf{S}} b(s) P(s'|s,a) \\
&= \max_{\alpha \in D} \sum_{s \in \mathbf{S}} b(s) \cdot \left(\sum_{s' \in \mathbf{S}} \alpha(s') P(o|s',a) P(s'|s,a) \right)
\end{aligned}
$$

Here, $P(s'|b,a,o)$ is the belief of state s' in the next round when the belief in the current round is b, the agent takes action a and get the observation o, which is the $b(s')$ in Equation 2.2.

2.3 Restless Multi-armed Bandit (RMAB) Problem

In RMABs, each arm represents an independent Markov machine. At every round, the player chooses k out of n arms ($k < n$) to activate and receives the reward determined by the state of the activated arms. After that, the states of *all* arms will transition to new states according to certain Markov transition probabilities. The problem is called "restless" because the states of passive arms will also transition like active arms. The aim of the player is to maximize his cumulative reward by choosing which arms to activate

at every round. It has shown by Papadimitriou and Tsitsiklis that it is PSPACE-hard to find the optimal strategy to general RMABs [41].

An index policy assigns an index to each state of each arm to measure how rewarding it is to activate an arm at a particular state. At every round, the index policy chooses to pick the k arms whose current states have the highest indices. Since the index of an arm only depends on the properties of this arm, index policy reduces an n-dimensional problem to n 1-dimensional problems so that the complexity is reduced from exponential with n to linear with n.

Whittle proposed a heuristic index policy for RMABs by considering the Lagrangian relaxation of the problem [57]. It has been shown that Whittle index policy is asymptotically optimal under certain conditions as k and n tend to ∞ with k/n fixed [56]. When k and n are finite, extensive empirical studies have also demonstrated the near-optimal performance of Whittle index policy [3, 13]. Whittle index measures how attractive it is to activate an arm based on the concept of subsidy for passivity. It gives the subsidy m to passive action (not activate) and the smallest m that would make passive action optimal for the current state is defined to be the Whittle index for this arm at this state. Whittle index policy chooses to activate the k arms with the highest Whittle indices. Intuitively, the larger the m is, the larger the gap is between active action (activate) and passive action, the more attractive it is for the player to activate this arm. Mathematically, denote $V_m(x; a = 0)$ ($V_m(x; a = 1)$) to be the maximum cumulative reward the player can

achieve until the end if he takes passive (active) action at the first round at the state x with subsidy m. Whittle index $I(x)$ of state x is then defined to be:

$$I(x) \triangleq \inf_m \{m : V_m(x; a = 0) \geq V_m(x : a = 1)\}$$

However, Whittle index only exists and Whittle index policy can only be used when the problem satisfies a property known as indexability, which I define below. Define $\Phi(m)$ to be the set of states for which passive action is the optimal action given subsidy m:

$$\Phi(m) \triangleq \{x : V_m(x; a = 0) \geq V_m(x : a = 1)\}$$

Definition 2.3.1. *An arm is indexable if $\Phi(m)$ monotonically increases from $\emptyset$ to the whole state space as m increases from $-\infty$ to $+\infty$. An RMAB is indexable if every arm is indexable.*

Intuitively, indexability requires that for a given state, its optimal action can never switch from passive action to active action with the increase of m. The indexability of an RMAB is often difficult to establish and computing Whittle index can be complex.

Chapter 3

Related Work

3.1 Uncertainty in Stackelberg Security Games

Previous approaches that handle uncertainty in SSGs can be divided into two categories:

- model uncertainty in terms of different attacker types and solve a resulting Bayesian Stackelberg game [42, 62]
- apply robust optimization techniques to optimize the worst case for the defender over the range of model uncertainty [19, 34, 60]

3.1.1 Bayesian Stackelberg Games

Bayesian Stackelberg game models uncertainty by allowing different attacker types, where there is some prior probability corresponding to each attacker type. Although this method is used to model payoff uncertainty in previous work [42, 62], it can also be used to model different degrees of attacker risk awareness in SSGs. However, this approach requires a prior distribution of attacker types, which is usually inapplicable for many real-world

security domains [34]. In addition, it is difficult to apply this approach to infinitely many attacker types.

3.1.2 Robust Stackelberg Games

Maximin method Maximin method for addressing uncertainties in SSGs focuses on maximizing the defender's utility against the worst case of uncertainties. Yin et al. [60] computes a defender strategy that is robust against defender execution uncertainty as well as uncertainty in the attacker's observations of the defender's strategy. Kiekintveld et al. [19] focus on interval uncertainty in the attacker's payoffs. Nguyen et al. [34] develop a robust strategy that takes the attacker's bounded rationality into account as well as the uncertainties [19, 60] discuss.

Minimax regret method Minimax regret method captures another concept of robustness when handling uncertainties in SSGs. In particular, it attempts to minimize the maximum "regret" or distance of a decision (e.g., defender's strategy) from the actual optimal decision for any instance within the uncertainty. Nguyen et al. [36] uses this concept of robustness in handling interval uncertainty in the attacker's payoffs.

The previous work has addressed neither attacker risk awareness nor ambiguity about the attacker risk profile. Although Kiekintveld et al. [19] and Nguyen et al. [36] try to capture uncertainty in attacker's utilities, they are unable to fully capture the attacker's risk awareness. The mapped utilities are coupled in the risk awareness setting since they are mapping with the same utility function $\widehat{U}$, and interval uncertainty is unable to model that. For example, Suppose target t_1 is of reward 1 and penalty -2; target t_2 is of reward 2 and penalty -1. The coverage probability $c_1 = c_2 = 0.5$. A risk-averse attacker will

always attack t_2 since $\widehat{U}$ must be strictly increasing. However, the model with interval uncertainty 1 would consider both t_1, t_2 to be potential targets for attack.

3.2 Adversary Behavioral Models

Some previous work explores human's bounded rationality in decision making — humans do not necessarily choose the strategy that provides them the highest expected utility [8]. Quantal response [31, 32] argues that human are more likely to choose the strategy with a higher expected utility. Yang et al. [59] apply the concept of quantal response to security games and compute the optimal strategy for the defender assuming that the attacker's response follows Quantal response. Nguyen et al. [35] propose the SUQR model by extending the quantal response concept with subjective utilities in security games. However, these approaches do not model risk awareness, and nor do they model uncertainty in risk awareness that I model in my thesis. In fact, models such as SUQR essentially address concerns that are orthogonal to the issue of risk awareness; future research may thus consider integrating bounded rationality models with risk awareness.

3.3 Learning Attacker Payoffs

There has been previous work on learning attacker payoffs in repeated security games [26, 30]. Letchford et al. [26] develop an algorithm to uncover the attacker type in as few rounds as possible, while my work focuses on maximizing the defender's utility. Marecki et al. [30] use MCTS to maximize the defender's utility in the first few rounds. However, their algorithm is unable to offer guidance in later rounds because it does not allow for

belief updating, which is a major component of my work. Additionally, Letchford et al. [26] and Marecki et al. [30] both assume that the defender plays a mixed strategy and the attacker plays a pure strategy that maximizes his expected utility given the defender's mixed strategy. However, illegal extractions happen frequently in resource green security domains, so the assumption that the attacker carries out surveillance over a long time to know the exact mixed strategy of the defender does not hold. Furthermore, I relax the assumption that the attacker is perfectly rational to handle more general behavior models such as quantal response.

3.4 Green Security Games

There is a significant body of literature discussing the activity of illegal extraction of natural resources [2, 16, 29]. In particular, this topic has also become popular in the AI community which emphasizes mathematical approaches [15, 18, 58]. Haskell et al. [15] and Yang et al. [58] model the game between the defender and the attacker as a repeated Stackelberg game where the defender plays a mixed strategy and the attacker plays a pure strategy against the mixed strategy at every round. They assume that the attacker is a non-rational SUQR playing agent [35] with unknown parameters. They use MLE (Maximum Likelihood Estimation) to estimate those parameters from the attacker's actions and optimize the defender's strategy against the estimate. Fang et al. [12] extends these works by assuming that the attacker responds to a certain convex combination of the defender's mixed strategies in previous few rounds.

One main difference of my work from these previous works is that I consider a short period as a round so that the defender plays a pure strategy at every round while previous works consider a long period as a round so that the defender plays a mixed strategy at every round. my model has the following advantages: (i) from the modeling perspective, it is difficult for the attacker to realize that the defender has switched from one mixed strategy to another mixed strategy. Furthermore, the attacker carries out illegal extractions frequently so the attacker might not have enough time to fully observe the new mixed strategy; (ii) from the strategy flexibility perspective, my model is capable of designing more flexible strategies since "playing a mixed strategy for a long period" can be represented as "playing a randomized strategy according to some probability distribution everyday during that period" while most short-period-based strategies can not be represented by long-period-based strategies.

These previous works also suffer from another two limitations. First, this research fails to capture the defender's *lack of observation of attacks* — in the real world, given a large area to patrol, the defender only has observations of attacks on the limited set of targets she patrolled in any given round. She does not have full knowledge of all of attackers' actions as assumed in [12, 15, 18, 58], leading to an unaddressed exploration-exploitation tradeoff for defenders: informally, should the defender allocate resources to protect targets that have already been visited and have been observed to have suffered a lot of attacks or should she allocate resources to protect targets that have not visited for a long time and hence where there are no observations of attacks. Second, while significant work in security games has focused on uncertainty over attackers' observations of defender actions [60], the reverse problem has received little attention. Specifically,

given frequent interactions with multiple attackers, the defender herself faces *observation uncertainty* in observing all of the attacker actions *even in the targets she does patrol*. Addressing this uncertainty in the defender's observation is important when estimating attacks on targets and addressing the exploration-exploitation tradeoff.

3.5 Exploration-exploitation Tradeoff in Security Domains

The limited observability property and exploration-exploitation tradeoff is also noticed by Klíma in the domain of border patrol where the border is large area [21, 22]. They model the problem as a stochastic/adversarial multi-armed bandit problem and use (sliding-window) UCB algorithm [4]/EXP3 algorithm [5] to plan for patrollers' strategies. However, the stochastic bandit formulation fails to model patrol's effect on attackers' actions while the adversarial bandit formulation fails to capture attackers' behavioral pattern.

3.6 Indexability of Restless Multi-armed Bandit Problem

There is a rich literature on indexability of restless multi-armed bandit problem. Glazebrook et al. [13] provide some indexable families of restless multi-armed bandit problems. Nino-Mora [37] propose PCL-indexability and GCL-indexability and show that they are sufficient conditions for indexability. Liu and Zhao [27] apply the concept of RMABs in dynamic multichannel access. In their model, every arm is a two-state Markov chain and the player only knows the state of the arm he chooses to activate. They prove the indexability of their problem and find the closed-form solution for the Whittle index. In [38], Ny et al. also consider the same class of RMABs but motivated by the application of

UAV routing. This problem shares some similarity with my problem but my problem is more difficult in the following aspects: (i) I cannot directly observe the states (POMDP vs. MDP); (ii) different actions lead to different transition matrices in my model; (iii) I allow for more states and observations. A further extension to this work discusses the case with probing errors where the player's observation about the state might be incorrect [28]. This concept is similar to what I assume in my model, but the detailed settings are different.

Chapter 4

Robust Strategy against Risk-aware Attackers in SSGs

This Chapter discusses my contribution of computing the robust strategy against risk-aware attackers. I will use the risk-averse attackers as an example to discuss the algorithm to compute the robust strategy, and then extend the algorithm to handle other types of risk-aware attackers.

A major motivation of this work is that the attacker is risk-averse in some domains, e.g., terrorists in the counter-terrorism domain. George Habash of the Popular Front for the Liberation of Palestine has said *"the main point is to select targets where success is* 100% *assured "* [17]. A report from RAND corporation [33] discusses the role of deterrence in counter-terrorism domain. They mention the evidence in the report that:

> In the doctrine of groups like the Provisional Irish Republican Army, requirements for operational planning include explicit consideration of how pre-attack surveillance can be used to manage and reduce operational risks. Similarly, in a document captured from the Islamic State of Iraq/al Qaeda in Iraq, a group member laments the deleterious effects on potential suicide bombers

when they suspect that poor planning may result in their lives being wasted on low-value targets.

The RAND report takes advantage of the fact that terrorists are risk-averse and hate uncertainty and discusses several possible solutions in increasing the uncertainties for terrorists to deter them. Besides that, creating uncertainty is already a key part of some security planning. For example, the Transportation Research Board [40] suggests one goal of security should be to *"create a high degree of uncertainty among terrorists about their chances of defeating the system."*. A similar point was made by the Defense Science Board [39] with respect to deterrence as part of national defense against nuclear terrorism:

> The deterrent aspect of the protection equation involves the often-great differences between how a defender and an attacker will view the relative capabilities of the defense. The long history of offense/defense competitions is strongly characterized by both sides taking ownside-conservative views. More particularly, the annals of terrorism and counterterrorism are replete with instances in which a prospective attacker was deterred by aspects of the defense that may have seemed relatively weak and ineffectual to the defender. The terrorist may not be afraid to die, but he (or his master) does not want to fail. Dissuasion/deterrence by the adversary's fear of failure might work in a variety of ways. One aspect is that an attacker will want to know enough about the defense to design a robust, successful attack. If the capabilities of the defense can be improved enough that the attacker must know the details of defensive measures in place to understand how to best surmount them, then

> the attacker may expose himself to discovery during the planning phases of the attack or be altogether dissuaded from the attempt. Creating uncertainty in the attacker's mind will be critical to maximizing the success of defenses which, realistically, cannot aspire to perfection. To exploit the effects of uncertainty, the defense should be deliberately designed and deployed to create as much ambiguity for the attacker as possible as to where the boundaries of defense performance lie.

There is another thread of work that studies terrorist risk attitudes [45,47,49]. In [47], portfolio theory is applied to study a terrorist group's decision making process, and this research argues that terrorist strategies are risk-averse and are highly sensitive to the group's level of risk aversion. While this finding of risk aversion may appear to be counter-intuitive, notice that it is the terrorist groups (and the planners in these groups) that are found to be risk-averse due to resource limitation; not the individuals in the organization who finally launch an attack. [45] studies the risk preferences of Al Qaeda specifically and concludes that the group is risk-averse and consistently displays the same degree of risk aversion in their activities. This work is further extended in [49] where the degree of risk aversion for Al Qaeda is estimated empirically based on data of attacks over the last decade.

I firstly build a robust SSG framework against an attacker with uncertainty in level of risk aversion. Second, building on previous work on SSGs in mixed-integer programs, I propose a novel mixed-integer bilinear programming problem (MIBLP), and find that it only finds locally optimal solutions. While the MIBLP formulation is also unable to

scale up, it provides key intuition for my new algorithm. This new algorithm, BeRRA (**B**inary search based **R**obust algorithm against **R**isk-**A**verse attackers) is my third contribution, and it finds globally ϵ-optimal solutions by solving $\mathcal{O}(n \log(\frac{1}{\epsilon}) \log(\frac{1}{\delta}))$ linear feasibility problems. The key idea of the BeRRA algorithm is to reduce the problem from maximizing the reward with a given number of resources to minimizing the number of resources needed to achieve a given reward. This transformation allows BeRRA to scale up via the removal of the bilinear terms and integer variables as well as the utilization of key theoretical properties that prove correspondence of its potential "attack sets" [20] with that of the maximin strategy. Finally, I also show that the defender does not need to consider attacker's risk attitude in zero-sum games. The experimental results show the solution quality and runtime advantages of my robust model and BeRRA algorithm.

4.1 Model

The SSE strategy provides the optimal defender strategy when the attacker is risk-neutral. However, as previously discussed, attackers are risk-averse rather than risk-neutral in several key domains. If the defender executes the SSE strategy against a risk-averse attacker, then the defender may suffer significant losses in solution quality. I show in Example 1 that these losses can be arbitrarily large.

Example 4.1.1. *Suppose there are two targets, t_1 and t_2, in the game, and their utilities are as shown in Table 4.1. The defender has only* 1 *resource. The SSE strategy of the game is $c_1 = 0.4, c_2 = 0.6$. If the attacker is risk-averse, he would choose to attack t_1*

(these two targets are identical to the attacker in terms of expected utility, but a risk-averse attacker prefers a small reward with high probability versus a high reward with low probability), and the defender's reward would be $0.4+0.6x$ *for the SSE strategy. However, if the defender executes the strategy of* $c_1 = 1, c_2 = 0$*, then the attacker would attack* t_2 *and the defender would receive reward* -1*. Compared with* -1*, the loss of the SSE strategy can be arbitrarily large since* x *can be arbitrarily small.*

Table 4.1: Utility Example

	U_d^c	U_d^u	U_a^c	U_a^u
t_1	1	x	-1	1
t_2	1	-1	-1	2

This example strongly motivates the need to consider risk-averse attackers. However, real world defenders are uncertain about the attacker's degree of risk aversion, and the defender may suffer significant losses if she incorrectly estimates it. Therefore I focus on a robust strategy in this work, i.e., my aim is to compute a defender strategy that is robust against all possible risk-averse attackers.

In literature on risk, the utility function f, which maps values to utilities, is used to specify the risk preference. f is concave for the risk-averse case and is convex for the risk-seeking case, while the risk-neutral case corresponds to the function $y = Cx, C > 0$. The agent makes decisions based on the mapped utilities.

In my problem, I define the mapping function $\widehat{U}$ that maps the utilities $U_a^c(i)$ and $U_a^u(i)$ to the attacker's mapped utilities. I denote $\widehat{U}_a(i, \mathbf{c}) \triangleq c_i\widehat{U}(U_a^c(i)) + (1 - c_i)\widehat{U}(U_a^u(i))$ as the attacker's expected utility under the mapping $\widehat{U}$. I restrict $\widehat{U}$ to be strictly increasing, concave and satisfying the equality $\widehat{U}(0) = 0$ — strictly increasing reflects the preference for more to less; concavity corresponds to risk aversion; and $\widehat{U}(0) = 0$ distinguishes

between gains and losses. According to this definition, the risk-averse case includes the risk-neutral case.

I define $\mathcal{U}$ to be the set of all valid mapping functions $\widehat{U}$. Problem (4.1) describes the robust defender strategy through a bilevel optimization problem. In the upper level, the defender chooses $\mathbf{c}$ to maximize her expected utility $U_d(k, \mathbf{c})$. The constraint $k \in \arg\max_{i \in \mathbb{T}} \widehat{U}_a(i, \mathbf{c})$ requires target k to have the highest expected utility for the attacker under the utility mapping $\widehat{U}$ when the defender's strategy is $\mathbf{c}$. The lower level demonstrates that the defender maximizes her worst-case reward over all possible attacker responses with utility mapping functions $\widehat{U} \in \mathcal{U}$. The lower level also suggests that the attacker breaks ties against the defender due to the concept of robustness. I define the solution $\mathbf{c}$ to be the Robust Stackelberg Equilibrium (RSE) strategy of the game.

$$\begin{aligned} &\max_{\mathbf{c}} \min_{\widehat{U} \in \mathcal{U}, k} \left\{ U_d(k, \mathbf{c}) : k \in \arg\max_{i \in \mathbb{T}} \widehat{U}_a(i, \mathbf{c}) \right\} \\ &s.t. \quad 0 \leq c_i \leq 1, \forall i \in \mathbb{T} \qquad \sum_{i \in \mathbb{T}} c_i \leq m \end{aligned} \tag{4.1}$$

4.2 Preliminaries

In its current form, the optimization problem (4.1) is not tractable because it is a bilevel programming problem that requires the solution of uncountably many inner optimization problems indexed by $\mathcal{U}$ [6]. To take steps towards tractability, in Section 4.2.1 and 4.2.2, I provide key concepts that are used in my MIBLP formulation (Section 4.3) and my BeRRA algorithm (Section 4.4).

4.2.1 Risk Aversion Modeling

In this section, I write the condition $\widehat{U} \in \mathcal{U}$ in a computationally tractable way via linear constraints. For any utility function $\widehat{U} \in \mathcal{U}$, we are actually only interested in its values at 0 and at the points of the attacker's utility set $U_a^c(i)$ and $U_a^u(i)$, which I denote as Θ:

$$\Theta = \{U_a^u(i), U_a^c(i), \forall i \in \mathbb{T}\} \bigcup \{0\} = \{\theta_1, \ldots, \theta_I\},$$

where $\theta_1 < \theta_2 < \cdots < \theta_I$.

Lemma 4.2.1. *Choose $\epsilon_u > 0$.*[1] *$\widehat{U} \in \mathcal{U}$ is equivalent to satisfying the linear constraints (4.2) on the values $\left\{\widehat{U}(\theta)\right\}_{\theta\in\Theta}$, i.e., $\forall \widehat{U} \in \mathcal{U}$, $\widehat{U}$ satisfies the constraints (4.2); $\forall \left\{\widehat{U}'(\theta)\right\}_{\theta\in\Theta}$ that satisfies constraints (4.2), $\exists \widehat{U} \in \mathcal{U}$ such that $\left\{\widehat{U}(\theta) = \widehat{U}'(\theta)\right\}_{\theta\in\Theta}$.*

$$\begin{aligned} \frac{\widehat{U}(\theta_2) - \widehat{U}(\theta_1)}{\theta_2 - \theta_1} &\geq \frac{\widehat{U}(\theta_3) - \widehat{U}(\theta_2)}{\theta_3 - \theta_2} \\ &\geq \ldots \geq \frac{\widehat{U}(\theta_I) - \widehat{U}(\theta_{I-1})}{\theta_I - \theta_{I-1}} \geq \epsilon_u \\ \widehat{U}(0) &= 0 \end{aligned} \tag{4.2}$$

Proof. If $\widehat{U} \in \mathcal{U}$, then $\left\{\widehat{U}(\theta)\right\}_{\theta\in\Theta}$ satisfies constraints (4.2) by definition. Conversely, if $\left\{\widehat{U}'(\theta)\right\}_{\theta\in\Theta}$ satisfies constraints (4.2), the piecewise linear function that connects $\{(\theta_1, \widehat{U}'(\theta_1)), (\theta_2, \widehat{U}'(\theta_2))\}$, $\{(\theta_2, \widehat{U}'(\theta_2)), (\theta_3, \widehat{U}'(\theta_3))\}$, $\ldots$, $\{(\theta_{I-1}, \widehat{U}'(\theta_{I-1})), (\theta_I, \widehat{U}'(\theta_I))\}$ belongs to $\mathcal{U}$. □

[1] Since Problem (4.1) is invariant under scaling of $\widehat{U}$, i.e., the attacker makes the same decision under either $\widehat{U}$ or $\alpha\widehat{U}, \forall \alpha > 0$. Thus, the value of ϵ_u does not affect the result.

Based on Lemma 4.2.1, the condition $\widehat{U} \in \mathcal{U}$ is completely captured by constraints (4.2). From now on I denote the constraints (4.2) compactly as $\widehat{U} \in \mathcal{U}$.

4.2.2 Possible Attack Set

In this section, to better understand Problem (4.1) I study the "possible attack set" $S_p(\mathbf{c})$ and its complement $S_i(\mathbf{c}) = \mathbb{T} - S_p(\mathbf{c})$.

Definition 4.2.2. *Given the coverage probability* $\mathbf{c}$*, Possible Attack Set* $S_p(\mathbf{c})$ *is defined to be the set of targets that may be attacked by a risk-averse attacker, i.e., it is the set of targets that have the highest expected utility for the attacker for some* $\widehat{U} \in \mathcal{U}$*.*

$S_i(\mathbf{c}) = \mathbb{T} - S_p(\mathbf{c})$ *is defined to be the set of targets that the attacker will never attack, i.e., the set of targets that for any* $\widehat{U} \in \mathcal{U}$*, there always exists another target* $i \in S_p(\mathbf{c})$ *with a higher expected utility for the attacker.*

Given the coverage probability $\mathbf{c}$, we can compute $S_p(\mathbf{c})$ and $S_i(\mathbf{c})$ by testing the feasibility of the following constraints for every target.

$$\begin{aligned} \widehat{U}_a(i, \mathbf{c}) \geq \widehat{U}_a(j, \mathbf{c}), \forall j \in T, j \neq i \\ \widehat{U} \in \mathcal{U} \end{aligned} \tag{4.3}$$

If these constraints are feasible for a target i, there exists a mapping $\widehat{U} \in \mathcal{U}$ under which target i has the highest expected utility for the attacker, and thus $i \in S_p(\mathbf{c})$; otherwise, $i \in S_i(\mathbf{c})$.

In Problem (4.1), when the defender's strategy $\mathbf{c}$ is given, the defender's (worst case) reward is:

$$\min_{\widehat{U}\in\mathcal{U},k}\left\{U_d(k,\mathbf{c}) : k\in\arg\max_{i\in\mathbb{T}}\widehat{U}_a(i,\mathbf{c})\right\}$$

which is equivalent to:

$$\min_{i\in S_p(\mathbf{c})}\{U_d(i,\mathbf{c})\}$$

So Problem (4.1) can be written as

$$\begin{aligned}&\max_{\mathbf{c}}\min_{i\in S_p(\mathbf{c})}\{U_d(i,\mathbf{c})\}\\ &\quad s.t.\quad 0\le c_i\le 1,\forall i\in\mathbb{T}\qquad \sum_{i\in\mathbb{T}}c_i\le m\end{aligned}\tag{4.4}$$

4.3 MIBLP Formulation

In this section, I formulate Problem (4.4) as an MIBLP problem to find the RSE strategy for the defender. While this approach does not scale up to large-scale games, it provides several insights for my BeRRA algorithm. As in Problem (2.1), I use integer variables

$\{q_i\}_{i\in\mathbb{T}}$ to denote if target i belongs to $S_p(\mathbf{c})$: I set $q_i = 1$ if $i \in S_p(\mathbf{c})$ and $q_i = 0$ if $i \in S_i(\mathbf{c})$. Problem (4.4) can then be converted to the formulation below

$$\begin{aligned}
\max_{\mathbf{c}} \quad & v \\
s.t. \quad & 0 \le c_i \le 1, \forall i \in \mathbb{T} \qquad \sum_{i\in\mathbb{T}} c_i \le m \\
& q_i \in \{0,1\}, \forall i \in \mathbb{T} \\
& v \le U_d(i,\mathbf{c}) + (1-q_i)M, \forall i \in \mathbb{T} \\
& i \in S_p(\mathbf{c}) \Leftrightarrow q_i = 1 \\
& i \in S_i(\mathbf{c}) \Leftrightarrow q_i = 0
\end{aligned} \tag{4.5}$$

When $i \in S_p(\mathbf{c})$, constraints (4.3) are feasible for target i. When $i \in S_i(\mathbf{c})$, for any utility mapping $\widehat{U} \in \mathcal{U}$, there is always another target with a higher expected utility for the attacker. I approximate this strict inequality with a small $\epsilon_c > 0$:

$$\min_{\widehat{U}\in\mathcal{U}} \left\{ \max_{j\in\mathbb{T}} \widehat{U}_a(j,\mathbf{c}) - \widehat{U}_a(i,\mathbf{c}) \right\} \ge \epsilon_c$$

which states that for any $\widehat{U} \in \mathcal{U}$, there exists a target $j \in \mathbb{T}$ whose expected utility for the attacker is at least ϵ_c more than the expected utility for target i. By substituting $\max_{j\in\mathbb{T}} \widehat{U}_a(j,\mathbf{c})$ with the slack variable λ, the preceding bilevel optimization problem can be reduced to:

$$\begin{aligned} \min_{\widehat{U},\lambda} \quad & \lambda - \widehat{U}_a(i,\mathbf{c}) \\ s.t. \quad & \widehat{U}_a(j,\mathbf{c}) \leq \lambda, \forall j \in \mathbb{T} \\ & \widehat{U} \in \mathcal{U} \end{aligned} \tag{4.6}$$

If the solution of Problem (4.6) is larger than ϵ_c, then $i \in S_i(\mathbf{c})$. Otherwise, $i \in S_p(\mathbf{c})$ (subject to the approximation error introduced by ϵ_c). Since Problem (4.6) is a minimization problem, it cannot substitute the constraint $i \in S_i(\mathbf{c}) \Leftrightarrow q_i = 0$ in Problem (4.5). So, I take the Lagrangian dual of Problem (4.6) to convert it into a maximization problem:

$$\begin{aligned} \max_{\boldsymbol{\alpha},\beta,\boldsymbol{\gamma},\kappa} \quad & \beta\epsilon_u \\ s.t. \quad & \sum_{j\in\mathbb{T}} \gamma_j = 1 \\ & \sum_{k\in\mathbb{T}} \gamma_k c_k \mathbf{1}\{\theta_j = U_a^c(k)\} + \gamma_k(1-c_k)\mathbf{1}\{\theta_j = U_a^u(k)\} \\ & \quad - c_i \mathbf{1}\{\theta_j = U_a^c(i)\} - (1-c_i)\mathbf{1}\{\theta_j = U_a^u(i)\} \\ & \quad + \frac{\alpha_{j-2}\mathbf{1}\{j \geq 3\}}{\theta_j - \theta_{j-1}} - \frac{\alpha_{j-1}\mathbf{1}\{I-1 \geq j \geq 2\}}{\theta_{j+1} - \theta_j} \\ & \quad - \frac{\alpha_{j-1}\mathbf{1}\{I-1 \geq j \geq 2\}}{\theta_j - \theta_{j-1}} + \frac{\alpha_j \mathbf{1}\{j \leq I-2\}}{\theta_{j+1} - \theta_j} \\ & \quad - \frac{\beta\mathbf{1}\{j = I\}}{\theta_j - \theta_{j-1}} + \frac{\beta\mathbf{1}\{j = I-1\}}{\theta_{j+1} - \theta_j} \\ & \quad + \kappa\mathbf{1}\{\theta_j = 0\} = 0, \forall j \in \{1,2,\ldots,I\} \\ & \alpha_j \geq 0, \forall j \in \{1,2,\ldots,I-2\} \\ & \beta \geq 0 \\ & \gamma_j \geq 0, \forall j \in \mathbb{T} \end{aligned} \tag{4.7}$$

For succinct notation, I denote the constraints on the variables $(\boldsymbol{\alpha}, \beta, \boldsymbol{\gamma}, \kappa)$ in the above formulation as $(\boldsymbol{\alpha}, \beta, \boldsymbol{\gamma}, \kappa) \in \mathcal{D}$. By applying Problem (4.3) and Problem (4.7) to every target i to put constraints on q_i, I summarize my final MIBLP formulation in the next theorem.

Theorem 4.3.1. *Problem (4.1) is (approximately)*[2] *equivalent to*

$$
\begin{aligned}
\max \quad & v \\
s.t. \quad & 0 \le c_i \le 1, \forall i \in \mathbb{T} \qquad \sum_{i \in \mathbb{T}} c_i \le m \\
& q_i \in \{0,1\}, \forall i \in \mathbb{T} \\
& v \le U_d(i, \mathbf{c}) + (1 - q_i)M, \forall i \in \mathbb{T} \\
& \widehat{U}_a^i(j, \mathbf{c}) \le \widehat{U}_a^i(i, \mathbf{c}) + (1 - q_i)M, \forall i \in \mathbb{T}, \forall j \in \mathbb{T}, j \ne i \\
& \widehat{U}^i \in \mathcal{U}, \forall i \in \mathbb{T} \\
& \beta^i \epsilon_u \ge \epsilon_c - q_i M, \forall i \in \mathbb{T} \\
& \left(\boldsymbol{\alpha}^i, \beta^i, \boldsymbol{\gamma}^i, \kappa^i\right) \in \mathcal{D}, \forall i \in \mathbb{T}
\end{aligned}
\tag{4.8}
$$

where the superscript i in $\left(\boldsymbol{\alpha}^i, \beta^i, \boldsymbol{\gamma}^i, \kappa^i\right)$ and $\widehat{U}^i$ marks different variables. $\widehat{U}_a^i(j, \mathbf{c})$ is attacker's expected utility for target j under the mapping $\widehat{U}^i$ and defender's strategy $\mathbf{c}$.

Proof. If $q_i = 1$, the constraints $\widehat{U}_a^i(j, \mathbf{c}) \le \widehat{U}_a^i(i, \mathbf{c}) + (1 - q_i)M, \forall j \in \mathbb{T}, j \ne i$ and $\widehat{U}^i \in \mathcal{U}$ ensure that $i \in S_p(\mathbf{c})$; if $q_i = 0$, then these constraints are always feasible and can be ignored.

If $q_i = 0$, the constraints $\beta^i \epsilon_u \ge \epsilon_c - q_i M$ and $\left(\boldsymbol{\alpha}^i, \beta^i, \boldsymbol{\gamma}^i, \kappa^i\right) \in \mathcal{D}$ ensure that $i \in S_i(\mathbf{c})$ (approximately) since there exists a solution $\left(\boldsymbol{\alpha}^i, \beta^i, \boldsymbol{\gamma}^i, \kappa^i\right) \in \mathcal{D}$ that satisfies $\beta^i \epsilon_u \ge \epsilon_c$.

[2]The approximation is due to the introduction of ϵ_c.

Thus, the objective of Problem (4.7) is larger than ϵ_c, and $i \in S_i(\mathbf{c})$. For the other direction, if the objective of Problem (4.7) is larger than ϵ_c, then these two constraints are also satisfied; if $q_i = 1$, these constraints are always feasible and can be ignored. □

In summary, I have converted Problem (4.1) into Problem (4.8), which is an MIBLP: $\{q_i\}_{i\in\mathbb{T}}$ are integer variables; $\widehat{U}_a^i(j,\mathbf{c}) = c_j\widehat{U}^i(U_a^c(j)) + (1-c_j)\widehat{U}^i(U_a^u(j))$ contains bilinear terms since both c_j and $\widehat{U}^i(U_a^c(j))/\widehat{U}^i(U_a^u(j))$ are variables. Problem (4.8) is a non-convex optimization problem and lacks efficient solvers. I used a powerful nonlinear solver — KNITRO to search for local optimal solutions to Problem (4.8). However, this approach does not scale up — the two-target scenario takes about 1 minute and the three-target scenario takes about 15 minutes to solve. Faced with this scalability issue, I develop the BeRRA algorithm that finds the ϵ-optimal solution and provides significant scalability.

4.4 BeRRA Algorithm

Problem (4.8) has two main hindrances to scaling up: the presence of $\Theta(n^2)$ bilinear terms and the presence of n integer variables. Thus, eliminating these bilinear terms and integer variables should allow us to scale the problem up. The bilinear terms in Problem (4.8) have two components: the coverage probability c_i and the mapped attacker utilities $\widehat{U}(U_a^c(i))/\widehat{U}(U_a^u(i))$. Intuitively, we can avoid the bilinearity by fixing one of these two terms. In addition, if the coverage probability $\mathbf{c}$ is fixed, then $S_p(\mathbf{c})$ is also fixed and we no longer need the integer variables $\{q_i\}_{i\in\mathbb{T}}$ to represent if $i \in S_p(\mathbf{c})$. Based on the idea of fixing the coverage probability $\mathbf{c}$, I develop the BeRRA algorithm. This algorithm computes an ϵ-optimal RSE strategy where ϵ can be made arbitrarily small.

The main idea of the BeRRA algorithm is to reduce the problem to computing the minimum amount of resources needed to achieve a given reward, which can be solved efficiently by using special properties of the problem. With this reduction, I use binary search to find the highest reward that the defender can achieve with the given number of resources. The high-level intuition of this reduction is that a fixed defender's reward leads to fixed defender maximin strategy, which eliminates the bilinear terms and integer variables. Additionally, optimal strategy can be derived efficiently from the maximin strategy via the property $S_p(\mathbf{c^{max}}) = S_p(\mathbf{c^{opt}})$.

4.4.1 Binary Search Reduction

Algorithm 1 lists the steps of my BeRRA algorithm. The input to Algorithm 1 is the number of defender resources m and the defender's and the attacker's utilities $\mathbf{U}$. The output is the defender's RSE strategy $\mathbf{c}$ and her reward lb. The lower bound lb and upper bound ub are first set to be the lowest and the highest possible rewards, respectively, that the defender may achieve (Line 2). The function MinimumResources(r, $\mathbf{U}$) returns the strategy $\mathbf{p}$ that uses the minimum number of resources for the defender to achieve reward r. This function will be discussed in detail in Section 4.5. During the binary search phase (Lines $3 \sim 11$), the lower bound is set to be the defender's achievable reward (the strategy $\mathbf{p}$ returned by the MinimumResources function is the solution) and the upper bound is set to be an unachievable reward. Therefore, the BeRRA algorithm achieves the ϵ-optimal solution and we can set ϵ arbitrarily small to get arbitrarily near-optimal solutions.

Algorithm 1 BeRRA Algorithm

function BeRRA $(m, \mathbf{U})$
  $lb \leftarrow \min_{i \in \mathbb{T}} U_d^u(i), ub \leftarrow \max_{i \in \mathbb{T}} U_d^c(i)$
  while $ub - lb \geq \epsilon$ **do**
    $\mathbf{p} \leftarrow$ MinimumResources($\frac{lb+ub}{2}$, $\mathbf{U}$)
    if $\sum_{i \in T} p_i \leq m$ **then**
      $lb \leftarrow \frac{lb+ub}{2}$
      $\mathbf{c} \leftarrow \mathbf{p}$
    else
      $ub \leftarrow \frac{lb+ub}{2}$
    end if
  end while
  return $(\mathbf{c}, lb)$
end function

4.5 Minimum Resources

I present Algorithm 2 in this section. This algorithm computes the defender strategy that requires as few resources as possible to achieve a given reward r, i.e., the Minimum-Resources function in Algorithm 1. I call this resource-minimizing strategy the optimal strategy and denote it as $\mathbf{c^{opt}}$ for succinctness.

Algorithm 2 Minimum Resources

function MinimumResources$(r, \mathbf{U})$
  $(flag, \mathbf{c^{max}}, S_p(\mathbf{c^{max}}), S_i(\mathbf{c^{max}})) \leftarrow$ Maximin$(r, \mathbf{U})$
  if $flag = false$ **then**
    return $(\infty, \infty, \ldots, \infty)^\top$
  end if
  $\mathbf{c^{opt}} \leftarrow$ Reduce$(\mathbf{U}, \mathbf{c^{max}}, S_p(\mathbf{c^{max}}), S_i(\mathbf{c^{max}}))$
  return $\mathbf{c^{opt}}$
end function

Algorithm 2 consists of two functions: Maximin and Reduce. The Maximin function computes the maximin strategy $\mathbf{c^{max}}$ for which the defender achieves reward r, as well as the corresponding sets $S_p(\mathbf{c^{max}})$ and $S_i(\mathbf{c^{max}})$. The variable $flag$ is set to $false$ when the input reward is not achievable for any amount of defender resources. In this

case, Algorithm 2 returns $(\infty, \infty, \ldots, \infty)^\top$ (Lines 3 $\sim$ 5) so that Algorithm 1 knows r is not achievable. I will prove in Theorem 4.5.7 that if the reward r is achievable, then $S_p(\mathbf{c^{max}}) = S_p(\mathbf{c^{opt}})$ and $S_i(\mathbf{c^{max}}) = S_i(\mathbf{c^{opt}})$. Based on this property, the Reduce function derives the optimal strategy $\mathbf{c^{opt}}$ from the maximin strategy $\mathbf{c^{max}}$. Section 4.5.1 and Section 4.5.2 discuss these two functions in detail.

4.5.1 Maximin Function

The Maximin function is summarized in Algorithm 3. It first computes the maximin strategy $\mathbf{c^{max}}$ for which the defender achieves reward r (Lines 2 $\sim$ 4) and then it assigns each target to either $S_p(\mathbf{c^{max}})$ or $S_i(\mathbf{c^{max}})$ (Lines 5 $\sim$ 15). If the reward r is not achievable for any amount of resources, then it returns $flag = false$ (Line 10).

Algorithm 3 Maximin

```
1:  function MAXIMIN(r, U)
2:      for i = 1 → n do
3:          c_i^max ← min{1, max{(r − U_d^u(i)) / (U_d^c(i) − U_d^u(i)), 0}}
4:      end for
5:      S_p(c^max), S_i(c^max) ← ∅
6:      for i = 1 → n do
7:          if Problem (4.3) is feasible for target i given c^max then
8:              S_p(c^max) ← S_p(c^max) ∪ {i}
9:              if r > U_d^c(i) then
10:                 return (false, c^max, S_p(c^max), S_i(c^max))
11:             end if
12:         else
13:             S_i(c^max) ← S_i(c^max) ∪ {i}
14:         end if
15:     end for
16:     return (true, c^max, S_p(c^max), S_i(c^max))
17: end function
```

Lines 2 $\sim$ 4 compute the maximin strategy for a given reward r. Given a coverage probability $\mathbf{c}$, the maximin setting assumes that the attacker attacks target $i =$

$\arg\min_{j\in\mathbb{T}} U_d(j,\mathbf{c})$, and thus the defender's reward will be $\min_{i\in\mathbb{T}} U_d(i,\mathbf{c})$. For the defender to achieve reward r, we should have $U_d(i,\mathbf{c}) \geq r, \forall i \in \mathbb{T}$ so that $c_i^{max} = \frac{r-U_d^u(i)}{U_d^c(i)-U_d^u(i)}$ (which is bounded by $[0,1]$).

Given the maximin strategy $\mathbf{c^{max}}$, Lines $5 \sim 15$ iterate through all targets and assign them to either $S_p(\mathbf{c^{max}})$ or $S_i(\mathbf{c^{max}})$ by testing the feasibility of constraints (4.3). If these constraints are feasible, then $i \in S_p(\mathbf{c})$; otherwise, $i \in S_i(\mathbf{c})$. Next in Lemma 4.5.2 I prove that $\exists i \in S_p(\mathbf{c^{max}})$ that satisfies $r > U_d^c(i)$ if and only if reward r is not achievable. In that case, Algorithm 3 returns $flag = false$ (Lines $9 \sim 11$).

Lemma 4.5.1. *Given coverage probability* $\mathbf{c}$*, the defender's reward is* $\min_{i\in S_p(\mathbf{c})} U_d(i,\mathbf{c})$.

Proof. Follows from the form of problem 4.4. □

Lemma 4.5.2. *Reward* r *is infeasible if and only if Algorithm 3 returns* $flag = false$.

Proof. I first prove that if the reward r is infeasible, Algorithm 3 returns with $flag = false$. If Algorithm 3 returns with $flag = false$, according to the steps of Algorithm 3, $\forall i \in S_p(\mathbf{c^{max}})$, we have $U_d^c(i) \geq r \Rightarrow U_d(i,\mathbf{c^{max}}) \geq r$. So, according to Lemma 4.5.1, the reward of the strategy $\mathbf{c^{max}}$ is at least r, so the reward r is feasible.

Next I prove that if Algorithm 3 returns with $flag = false$, then the reward r is infeasible. If Algorithm 3 returns with $flag = false$, then there exists a target $i \in S_p(\mathbf{c^{max}})$ such that $r > U_d^c(i)$, and we have $c_i^{max} = 1$ for this target. Since $i \in S_p(\mathbf{c^{max}})$, there exists a mapping $\widehat{U} \in \mathcal{U}$ under which $\widehat{U}_a(i,\mathbf{c^{max}}) = \widehat{U}_a^c(i) \geq \widehat{U}_a(j,\mathbf{c^{max}}), \forall j \in \mathbb{T}, j \neq i$. If the reward r is feasible, there must exist a strategy $\mathbf{c}$ which has reward at least r where $i \in S_i(\mathbf{c})$, or else the reward will be at most $U_d^c(i) < r$. Thus, there exists a target $j \neq i$ that maximizes the expected utility of the attacker under the mapping

$\widehat{U}$ and strategy $\mathbf{c}$ such that $\widehat{U}_a(j, \mathbf{c}) > \widehat{U}_a(i, \mathbf{c})$. Thus $\widehat{U}_a(j, \mathbf{c}) > \widehat{U}_a(i, \mathbf{c}) \geq \widehat{U}_a^c(i) \geq \widehat{U}_a(j, \mathbf{c^{max}})$, so $c_j < c_j^{max}$, which implies $U_d(j, \mathbf{c}) < U_d(j, \mathbf{c^{max}})$. However, $c_j < c_j^{max} \Rightarrow c_j^{max} > 0 \Rightarrow c_j^{max} = \min\{1, \frac{r - U_d^u(j)}{U_d^c(j) - U_d^u(j)}\} \Rightarrow U_d(j, \mathbf{c^{max}}) = \min\{U_d^c(j), r\} \leq r$, so we see $U_d(j, \mathbf{c}) < U_d(j, \mathbf{c^{max}}) \leq r$. Since $j \in S_p(\mathbf{c})$, the strategy $\mathbf{c}$ has a reward less than r, which contradicts the assumption that $\mathbf{c}$ has a reward at least r. In conclusion, r is infeasible if Algorithm 3 returns with $flag = false$. □

Theorem 4.5.7 demonstrates why we compute $\mathbf{c^{max}}$, $S_p(\mathbf{c^{max}})$ and $S_i(\mathbf{c^{max}})$. We see that $S_p(\mathbf{c^{max}}) = S_p(\mathbf{c^{opt}})$ and $S_i(\mathbf{c^{max}}) = S_i(\mathbf{c^{opt}})$. Therefore, we get $S_p(\mathbf{c^{opt}})$ and $S_i(\mathbf{c^{opt}})$ by computing $S_p(\mathbf{c^{max}})$ and $S_i(\mathbf{c^{max}})$. I introduce supporting lemmas before proving Theorem 4.5.7.

The next two lemmas explain how the set $S_p(\mathbf{c})$ changes when the coverage probability for a certain target decreases. Lemma 4.5.3 shows that if the coverage probability for a target $i \in S_p(\mathbf{c})$ decreases, then the set $S_p(\mathbf{c})$ "shrinks". Lemma 4.5.4 shows that if the coverage probability for a target $i \in S_i(\mathbf{c})$ decreases, then the set $S_p(\mathbf{c})$ also "shrinks" but target i might be added to it.

Lemma 4.5.3. *Given coverage probability* $\mathbf{c}$ *and another coverage probability* $\mathbf{c}'$ *which satisfies* $c'_i < c_i$ *for a target* $i \in S_p(\mathbf{c})$ *and* $c'_j = c_j, \forall j \in \mathbb{T}, j \neq i$, *we have* $S_p(\mathbf{c}') \subseteq S_p(\mathbf{c})$.

Proof. We prove $S_i(\mathbf{c}') \supseteq S_i(\mathbf{c})$.

$\forall j \in S_i(\mathbf{c})$, $\forall \widehat{U} \in \mathcal{U}$, $\exists k \in S_p(\mathbf{c})$ such that $\widehat{U}_a(k, \mathbf{c}) > \widehat{U}_a(j, \mathbf{c})$. For this mapping $\widehat{U}$, the targets j and k, $\widehat{U}_a(k, \mathbf{c}') \geq \widehat{U}_a(k, \mathbf{c})$ since $c'_k \leq c_k$ and $\widehat{U}_a(j, \mathbf{c}') = \widehat{U}_a(j, \mathbf{c})$ since $c'_j = c_j$. So, we have $\widehat{U}_a(k, \mathbf{c}') > \widehat{U}_a(j, \mathbf{c}')$ and thus $j \in S_i(\mathbf{c}')$. □

Lemma 4.5.4. *Given coverage probability* $\mathbf{c}$ *and another coverage probability* $\mathbf{c}'$ *which satisfies* $c'_i < c_i$ *for a target* $i \in S_i(\mathbf{c})$ *and* $c'_j = c_j, \forall j \in \mathbb{T}, j \neq i$, *we have* $S_p(\mathbf{c}') \subseteq S_p(\mathbf{c}) \bigcup \{i\}$.

Proof. We prove $S_i(\mathbf{c}') \supseteq S_i(\mathbf{c}) \backslash \{i\}$.

$\forall j \in S_i(\mathbf{c}), j \neq i, \forall \widehat{U} \in \mathcal{U}$, $\exists k \in S_p(\mathbf{c})$ such that $\widehat{U}_a(k, \mathbf{c}) > \widehat{U}_a(j, \mathbf{c})$. For this mapping $\widehat{U}$, the targets j and k, $\widehat{U}_a(k, \mathbf{c}') = \widehat{U}_a(k, \mathbf{c})$ since $c'_k = c_k$ and $\widehat{U}_a(j, \mathbf{c}') = \widehat{U}_a(j, \mathbf{c})$ since $c'_j = c_j$, so we have $\widehat{U}_a(k, \mathbf{c}') > \widehat{U}_a(j, \mathbf{c}')$ and thus $j \in S_i(\mathbf{c}')$. □

The next two lemmas discuss key properties of $\mathbf{c^{opt}}$. Lemma 4.5.5 shows that the coverage probability for a target $i \in S_p(\mathbf{c^{opt}})$ must be $\max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}$; Lemma 4.5.6 shows that the coverage probability for a target $i \in S_i(\mathbf{c^{opt}})$ is at most $\min\{1, \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}\}$. This property is used in the Reduce function that derives $\mathbf{c^{opt}}$ from $\mathbf{c^{max}}$, as well as in the proof of Theorem 4.5.7.

Lemma 4.5.5. *Given a feasible reward* r, *all* $i \in S_p(\mathbf{c^{opt}})$ *must satisfy* $U_d^c(i) \geq r$ *and have expected reward* $\max\{U_d^u(i), r\}$ *for the defender, i.e.,* $c_i^{opt} = \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}, \forall i \in S_p(\mathbf{c^{opt}})$.

Proof. According to Lemma 4.5.1, $\forall i \in S_p(\mathbf{c^{opt}})$, $U_d(i, \mathbf{c^{opt}}) \geq r$, so we have $U_d^c(i) \geq U_d(i, \mathbf{c^{opt}}) \geq r$. Additionally, $U_d(i, \mathbf{c^{opt}}) \geq r \Rightarrow c_i^{opt} \geq \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}$. Next I will prove $c_i^{opt} = \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}$ by contradiction.

Suppose there exists a target $i \in S_p(\mathbf{c^{opt}})$ with coverage probability $c_i^{opt} > \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}$. I show that a more resource-conservative strategy $\mathbf{c}$ with $c_i = \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\} < c_i^{opt}, c_j = c_j^{opt}, \forall j \in \mathbb{T}, j \neq i$ also has reward at least r for the defender. According to Lemma 4.5.1, we have $U_d(i, \mathbf{c^{opt}}) \geq r, \forall i \in S_p(\mathbf{c^{opt}})$. According to Lemma

4.5.3, $S_p(\mathbf{c}) \subset S_p(\mathbf{c^{opt}})$, so $\forall k \in S_p(\mathbf{c}), k \in S_p(\mathbf{c^{opt}})$, if $k = i$, $U_d(k, \mathbf{c}) = U_d(i, \mathbf{c}) = \max\{U_d^u(i), r\} \geq r$; if $k \neq i$, $U_d(k, \mathbf{c}) = U_d(k, \mathbf{c^{opt}}) \geq r$, so the strategy $\mathbf{c}$ also provides reward at least r. Thus $\mathbf{c^{opt}}$ is not optimal, which is a contradiction. □

Lemma 4.5.6. *Given a feasible reward r, $\forall i \in S_i(\mathbf{c^{opt}})$, i has expected reward at most $\min\{U_d^c(i), \max\{U_d^u(i), r\}\}$ for the defender, i.e., $c_i^{opt} \leq \min\{1, \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}\}, \forall i \in S_i(\mathbf{c^{opt}})$.*

Proof. $\forall i \in S_i(\mathbf{c^{opt}})$, if $U_d^c(i) < r$, $\max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\} > 1$, so we have $\min\{1, \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}\} = 1$. Clearly, $c_i^{opt} \leq 1$.

$\forall i \in S_i(\mathbf{c^{opt}})$, if $U_d^c(i) \geq r$, $\min\{1, \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}\} = \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\} \leq 1$. I prove $c_i^{opt} \leq \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}$ by contradiction.

Suppose there exists a target $i \in S_i(\mathbf{c^{opt}})$ with coverage probability $c_i^{opt} > \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\}$. I show that a more resource-conservative strategy $\mathbf{c}$ where $c_i = \max\{\frac{r - U_d^u(i)}{U_d^c(i) - U_d^u(i)}, 0\} < c_i^{opt}, c_j = c_j^{opt}, \forall j \in \mathbb{T}, j \neq i$ has reward at least r for the defender. According to Lemma 4.5.1, we have $U_d(k, \mathbf{c^{opt}}) \geq r, \forall k \in S_p(\mathbf{c^{opt}})$. According to Lemma 4.5.4, $S_p(\mathbf{c}) \subset S_p(\mathbf{c^{opt}}) \bigcup \{i\}$. It follows that $\forall k \in S_p(\mathbf{c}), k \in S_p(\mathbf{c^{opt}}) \bigcup \{i\}$, if $k = i$, $U_d(k, \mathbf{c}) = U_d(i, \mathbf{c}) = \max\{U_d^u(i), r\} \geq r$; if $k \in S_p(\mathbf{c^{opt}})$, $U_d(k, \mathbf{c}) = U_d(k, \mathbf{c^{opt}}) \geq r$, so the strategy $\mathbf{c}$ also provides reward at least r. Thus $\mathbf{c^{opt}}$ is not optimal, which is a contradiction. □

We are now ready to combine these preliminary lemmas to prove Theorem 4.5.7.

Theorem 4.5.7. *Given a feasible reward r, $S_p(\mathbf{c^{max}}) = S_p(\mathbf{c^{opt}})$ and $S_i(\mathbf{c^{max}}) = S_i(\mathbf{c^{opt}})$.*

Proof. First I present two results that will be used in the proof: (i) $\forall i \in S_p(\mathbf{c^{max}})$, since the reward r is feasible, Lemma 4.5.2 and Algorithm 3 imply that $U_d^c(i) \geq r$ so that $c_i^{max} = \max\{\frac{r-U_d^u(i)}{U_d^c(i)-U_d^u(i)}, 0\}$; (ii) $\forall i \in S_p(\mathbf{c^{opt}})$, according to Lemma 4.5.5, $U_d^c(i) \geq r$ and $c_i^{opt} = \max\{\frac{r-U_d^u(i)}{U_d^c(i)-U_d^u(i)}, 0\}$. Furthermore, $U_d^c(i) \geq r$ implies that $c_i^{max} = \max\{\frac{r-U_d^u(i)}{U_d^c(i)-U_d^u(i)}, 0\}$ according to Algorithm 3. Thus, $c_i^{max} = c_i^{opt}, \forall i \in S_p(\mathbf{c^{opt}})$.

I will prove by contradiction that $S_p(\mathbf{c^{opt}}) \subseteq S_p(\mathbf{c^{max}})$. Suppose there exists a target $i \in S_p(\mathbf{c^{opt}})$ with $i \in S_i(\mathbf{c^{max}})$, we have $c_i^{max} = c_i^{opt}$ according to result (ii). Since $i \in S_p(\mathbf{c^{opt}})$, $\widehat{U}_a(i, \mathbf{c^{opt}}) \geq \widehat{U}_a(j, \mathbf{c^{opt}}), \forall j \in \mathbb{T}, j \neq i$ under some mapping $\widehat{U} \in \mathcal{U}$ by definition. Since $i \in S_i(\mathbf{c^{max}})$, for this mapping $\widehat{U}$, $\exists j \neq i, j \in S_p(\mathbf{c^{max}})$ such that $\widehat{U}_a(j, \mathbf{c^{max}}) > \widehat{U}_a(i, \mathbf{c^{max}})$ where $c_j^{max} = \max\{\frac{r-U_d^u(j)}{U_d^c(j)-U_d^u(j)}, 0\}$ according to result (i). So $\widehat{U}_a(j, \mathbf{c^{max}}) > \widehat{U}_a(i, \mathbf{c^{max}}) = \widehat{U}_a(i, \mathbf{c^{opt}}) \geq \widehat{U}_a(j, \mathbf{c^{opt}})$, which leads to $\widehat{U}_a(j, \mathbf{c^{max}}) > \widehat{U}_a(j, \mathbf{c^{opt}})$. Thus, we have $c_j^{opt} > c_j^{max} = \max\{\frac{r-U_d^u(j)}{U_d^c(j)-U_d^u(j)}, 0\}$, which contradicts Lemmas 4.5.5 and 4.5.6. So, it must be that $i \in S_p(\mathbf{c^{max}})$ which implies $S_p(\mathbf{c^{opt}}) \subseteq S_p(\mathbf{c^{max}})$.

I will prove by contradiction that $S_i(\mathbf{c^{opt}}) \subseteq S_i(\mathbf{c^{max}})$. Suppose there exists a target $i \in S_i(\mathbf{c^{opt}})$ with $i \in S_p(\mathbf{c^{max}})$. We have $c_i^{max} = \max\{\frac{r-U_d^u(i)}{U_d^c(i)-U_d^u(i)}, 0\}$ according to result (i). Since $i \in S_p(\mathbf{c^{max}})$ there exists a mapping $\widehat{U} \in \mathcal{U}$ such that $\widehat{U}_a(i, \mathbf{c^{max}}) \geq \widehat{U}_a(j, \mathbf{c^{max}})$, $\forall j \in \mathbb{T}, j \neq i$. Since $i \in S_i(\mathbf{c^{opt}})$, for this mapping $\widehat{U}$, $\exists j \neq i, j \in S_p(\mathbf{c^{opt}})$ such that $\widehat{U}_a(j, \mathbf{c^{opt}}) > \widehat{U}_a(i, \mathbf{c^{opt}})$ by definition. We have $c_j^{max} = c_j^{opt}$ according to result (ii). Thus $\widehat{U}_a(i, \mathbf{c^{max}}) \geq \widehat{U}_a(j, \mathbf{c^{max}}) = \widehat{U}_a(j, \mathbf{c^{opt}}) > \widehat{U}_a(i, \mathbf{c^{opt}})$, which yields $\widehat{U}_a(i, \mathbf{c^{max}}) > \widehat{U}_a(i, \mathbf{c^{opt}})$. Then $c_i^{opt} > c_i^{max} = \max\{\frac{r-U_d^u(i)}{U_d^c(i)-U_d^u(i)}, 0\}$, which contradicts Lemma 4.5.6. It follows that $i \in S_i(\mathbf{c^{max}})$ which implies $S_i(\mathbf{c^{opt}}) \subseteq S_i(\mathbf{c^{max}})$.

To conclude, $S_p(\mathbf{c^{opt}}) = S_p(\mathbf{c^{max}})$, $S_i(\mathbf{c^{opt}}) = S_i(\mathbf{c^{max}})$. □

4.5.2 Reduce Function: Derive $\mathbf{c^{opt}}$ from $\mathbf{c^{max}}$

Section 4.5.1 demonstrated that Algorithm 2 returns $(\infty, \infty, \ldots, \infty)^\top$ if the reward r is infeasible; if the reward r is feasible, then $S_p(\mathbf{c^{max}})$ and $S_i(\mathbf{c^{max}})$ are the same as $S_p(\mathbf{c^{opt}})$ and $S_i(\mathbf{c^{opt}})$. It follows that Algorithm 4 correctly derives the optimal strategy $\mathbf{c^{opt}}$ from $\mathbf{c^{max}}$.

Algorithm 4 Computing $\mathbf{c^{opt}}$ by reducing $\mathbf{c^{max}}$

```
1: function REDUCE(U, c^max, S_p(c^max), S_i(c^max))
2:     c^opt = c^max
3:     for every i ∈ S_i(c^max) do
4:         lb ← 0, ub ← c_i^opt
5:         while ub − lb ≥ δ do
6:             c_i^opt ← (lb+ub)/2
7:             if Problem (4.3) is feasible for target i given c^opt then
8:                 lb ← (lb+ub)/2
9:             else
10:                ub ← (lb+ub)/2
11:            end if
12:        end while
13:        c_i^opt ← ub
14:    end for
15:    return c^opt
16: end function
```

Given $\mathbf{c^{max}}$, $S_p(\mathbf{c^{max}})$ and $S_i(\mathbf{c^{max}})$, Algorithm 4 returns $c_i^{opt} = c_i^{max}$ for $i \in S_p(\mathbf{c^{max}})$. For $i \in S_i(\mathbf{c^{max}})$, Algorithm 4 uses binary search to find the minimum coverage probability c_i such that any further decrease[3] in coverage probability would add target i to the set $S_p(\mathbf{c^{opt}})$. The next lemma shows that this mechanism leads to the optimal strategy $\mathbf{c^{opt}}$.

Lemma 4.5.8. *Given a feasible reward r, Algorithm 4 returns the optimal strategy $\mathbf{c^{opt}}$.*

[3] δ can be arbitrarily small

Proof. Given a feasible reward r, we have $S_p(\mathbf{c^{opt}}) = S_p(\mathbf{c^{max}})$, $S_i(\mathbf{c^{opt}}) = S_i(\mathbf{c^{max}})$ according to Theorem 4.5.7. Denote the output of Algorithm 4 to be $\mathbf{c}$.

I first prove that $\mathbf{c}$ satisfies $S_p(\mathbf{c}) = S_p(\mathbf{c^{opt}})$, $S_i(\mathbf{c}) = S_i(\mathbf{c^{opt}})$ by proving $S_p(\mathbf{c}) = S_p(\mathbf{c^{max}})$ and $S_i(\mathbf{c}) = S_i(\mathbf{c^{max}})$. The steps of Algorithm 4 ensures that $S_i(\mathbf{c^{max}}) \subseteq S_i(\mathbf{c})$, so we have $S_p(\mathbf{c}) \subseteq S_p(\mathbf{c^{max}})$. Next I prove $S_p(\mathbf{c^{max}}) \subseteq S_p(\mathbf{c})$. $\forall i \in S_p(\mathbf{c^{max}})$, $\exists \widehat{U} \in \mathcal{U}$ such that $\widehat{U}_a(i, \mathbf{c^{max}}) \geq \widehat{U}_a(j, \mathbf{c^{max}}), \forall j \in \mathbb{T}, j \neq i$. If $i \in S_i(\mathbf{c})$, for this mapping $\widehat{U}$, $\exists j \in S_p(\mathbf{c}) \subseteq S_p(\mathbf{c^{max}})$ such that $\widehat{U}_a(j, \mathbf{c}) > \widehat{U}_a(i, \mathbf{c})$. According to the steps of Algorithm 4, since $i, j \in S_p(\mathbf{c^{max}})$, $c_i^{max} = c_i$ and $c_j^{max} = c_j$, we have $\widehat{U}_a(j, \mathbf{c}) > \widehat{U}_a(i, \mathbf{c}) = \widehat{U}_a(i, \mathbf{c^{max}}) \geq \widehat{U}_a(j, \mathbf{c^{max}}) = \widehat{U}_a(j, \mathbf{c})$, which leads to a contradiction. Thus $i \in S_p(\mathbf{c})$ and $S_p(\mathbf{c^{max}}) \subseteq S_p(\mathbf{c})$. So $S_p(\mathbf{c}) = S_p(\mathbf{c^{max}}) = S_p(\mathbf{c^{opt}})$ and $S_i(\mathbf{c}) = S_i(\mathbf{c^{max}}) = S_i(\mathbf{c^{opt}})$.

Next I prove that the strategy $\mathbf{c}$ has reward at least r. $\forall i \in S_p(\mathbf{c})$, $i \in S_p(\mathbf{c^{max}})$. According to the steps of Algorithm 4 and Algorithm 3, $U_d(i, \mathbf{c}) = U_d(i, \mathbf{c^{max}}) \geq r$.

Finally, I prove $\mathbf{c} = \mathbf{c^{opt}}$. If $\mathbf{c} \neq \mathbf{c^{opt}}$, $\mathbf{c^{opt}}$ consumes fewer resources than $\mathbf{c}$, so $\sum_{i \in \mathbb{T}} c_i^{opt} < \sum_{i \in \mathbb{T}} c_i$. According to Lemma 4.5.5, $c_i^{opt} = c_i, \forall i \in S_p(\mathbf{c^{opt}})$, so there exists at least one target $i \in S_i(\mathbf{c^{opt}})$ with $c_i^{opt} < c_i$. However, Algorithm 4 is designed so that any further decrease in c_i would lead to a mapping $\widehat{U} \in \mathcal{U}$ such that $\widehat{U}_a(i, \mathbf{c}) \geq \widehat{U}_a(j, \mathbf{c}), \forall j \in S_p(\mathbf{c^{opt}})$. Thus at least one target in $S_i(\mathbf{c^{opt}})$ would be added to $S_p(\mathbf{c^{opt}})$, which contradicts the assumption that $\mathbf{c^{opt}}$ is the optimal solution. I conclude $\mathbf{c} = \mathbf{c^{opt}}$. □

Theorem 4.5.9. *Given reward r, Algorithm 2 either detects its infeasibility or provides the optimal strategy $\mathbf{c^{opt}}$.*

Proof. Follows from Lemmas 4.5.2 and 4.5.8. □

4.6 Discussions

4.6.1 Computational Cost of BeRRA

The main computational cost of my BeRRA algorithm comes from evaluating the feasibility of the linear constraints (4.3), which is a linear feasibility problem and can be solved in polynomial time. Algorithm 2 is called $\mathcal{O}(\log(\frac{1}{\epsilon}))$ times, and every call to Algorithm 2 involves solving Problem (4.3) $\mathcal{O}(n + |S_i(\mathbf{c^{max}})| \log(\frac{1}{\delta}))$ times, which is bounded by $\mathcal{O}(n \log(\frac{1}{\delta}))$. Thus Problem (4.3) is solved $\mathcal{O}(n \log(\frac{1}{\epsilon}) \log(\frac{1}{\delta}))$ times in total for my BeRRA algorithm.

4.6.2 Extensions of BeRRA to General Risk Awareness

Notice that we only require $\mathcal{U}$ to be increasing in the preceding proofs and algorithms. Thus, my GMOP algorithm can also be used to compute the optimal robust strategy against other kinds of risk-aware attacker types, e.g., risk-seeking criminals [7, 14]. Besides, the attacker may be risk-averse for gains and risk-seeking for losses (S-shaped utility mapping function as is shown in the right figure) as is suggested in the nobel-prize-winning prospect theory (PT)[4] model [31,32]. Here I use risk-seeking as an example to show how to apply GMOP to other attacker types. If the attacker is risk-seeking, $\widehat{U}$ should be a strictly increasing, convex function and satisfies $\widehat{U}(0) = 0$. Therefore, when adapting my GMOP algorithm to deal with risk-seeking attackers, the only difference is in testing feasibility of constraints (4.3), where the condition $\widehat{U} \in \mathcal{U}$ in constraints (4.3) should be written as:

[4]PT also involves a mapping of the probability. I don't consider this factor in this thesis.

$$\epsilon_u \leq \frac{\widehat{U}(\theta_2) - \widehat{U}(\theta_1)}{\theta_2 - \theta_1} \leq \frac{\widehat{U}(\theta_3) - \widehat{U}(\theta_2)}{\theta_3 - \theta_2} \leq \ldots \leq \frac{\widehat{U}(\theta_I) - \widehat{U}(\theta_{I-1})}{\theta_I - \theta_{I-1}} \tag{4.9}$$

$$\widehat{U}(0) = 0$$

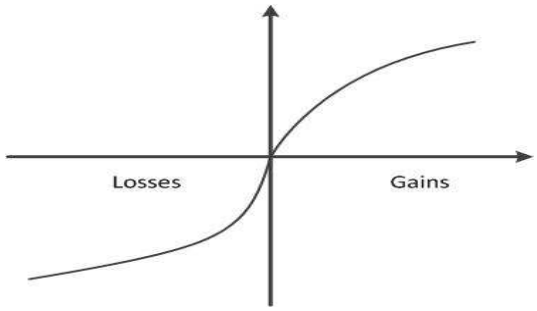


Figure 4.1: Prospect Theory

4.6.3 Zero-sum Game

When the game is a zero-sum game, the utilities for the defender and the attacker are strongly correlated with correlation coefficient -1, i.e., $U_a^c(i) = -U_d^c(i)$ and $U_a^u(i) = -U_d^u(i), \forall i \in \mathbb{T}$. Based on this property, I obtain the following theorem.

Theorem 4.6.1. *For zero-sum games, the defender's Robust Stackelberg Equilibrium (RSE) strategy and Maximin strategy are the same.*

Proof. I first prove that given a defender's strategy $\mathbf{c}$, the defender's reward is the same in both settings.

Given the defender's strategy $\mathbf{c}$, the defender's reward in the Maximin setting is $\min_{j \in \mathbb{T}} U_d(j, \mathbf{c})$, which is a lower bound on the defender's reward in the Robust Stackelberg game setting since $\min_{j \in \mathbb{T}} U_d(j, \mathbf{c})$ is the minimum reward the defender can possibly

achieve with $\mathbf{c}$. Meanwhile, since the risk-neutral case is a special case of the risk-averse case, $i = \arg\min_{j \in \mathbb{T}} U_d(j, \mathbf{c}) = \arg\max_{j \in \mathbb{T}} U_a(j, \mathbf{c}) \in S_p(\mathbf{c})$. Thus, the attacker might attack target i so that the expected reward the defender achieves if the attacker attacks target i — $\min_{j \in \mathbb{T}} U_d(j, \mathbf{c})$ is also an upper bound on the defender's reward in the Robust Stackelberg game setting. Therefore, the defender's reward in the Robust Stackelberg game setting is $\min_{j \in \mathbb{T}} U_d(j, \mathbf{c})$.

Since the defender's reward given the defender's strategy $\mathbf{c}$ is the same in both settings, the strategy $\mathbf{c}$ that maximizes the defender's reward in both settings is also the same. □

It is known that the solution concepts of Nash Equilibrium, minimax, maximin, and SSE all give the same answer for finite two-person zero-sum games. Therefore, Theorem 4.6.1 adds RSE to this equivalence list.

4.7 Experimental Evaluation

I will evaluate the performance of my algorithms in this section through extensive numerical experiments. Unless otherwise stated, all of the experiment results are averaged over 20 instances. $U_d^c(i)$ and $U_a^u(i)$ are generated as random variables between 11 and 40; $U_d^u(i)$ and $U_a^c(i)$ are generated as random variables between -11 and -40. To generate payoff matrixes with correlation between the defender's and the attacker's utilities, I set $U_a^c(i) \leftarrow -\alpha U_d^c(i) + \sqrt{1-\alpha^2} U_a^c(i)$ and $U_a^u(i) \leftarrow -\alpha U_d^u(i) + \sqrt{1-\alpha^2} U_a^u(i)$, where $-\alpha$ is the correlation coefficient between $U_a^c(i)(U_a^u(u))$ and $U_d^c(i)(U_d^u(i))$. $\alpha = 1$ corresponds to zero-sum games. n is the number of targets in the game and m is the number of resources the defender has.

4.7.1 MIBLP vs BeRRA

I first compare the performance of the MIBLP formulation and my BeRRA algorithm. Due to the scalability of the MIBLP, I only compare the case when $n = 2$ and $n = 3$. m is set to be 1. The KNITRO solver is used to solve the MIBLP formulation.

MIBLP vs BeRRA: Solution Quality The solution quality of the MIBLP formulation and my BeRRA algorithm is shown in Table 4.2. We can see from the table that BeRRA algorithm has a higher average reward compared with MIBLP, and the difference becomes larger as the number of targets n increases. This is because KNITRO can only find the locally optimal solution while my BeRRA algorithm finds the globally ϵ-optimal solution, and larger game scale leads to worse solution quality of the local optimum.

Table 4.2: MIBLP vs BeRRA in Solution Quality

(a) $n = 2$

	MIBLP	BeRRA
$\alpha = 0$	3.18	3.41
$\alpha = 0.2$	2.78	2.99
$\alpha = 0.4$	2.45	2.62
$\alpha = 0.6$	1.72	1.82
$\alpha = 0.8$	0.75	0.81
$\alpha = 1.0$	0.53	0.53

(b) $n = 3$

	MIBLP	BeRRA
$\alpha = 0$	-5.69	-4.60
$\alpha = 0.2$	-5.71	-5.32
$\alpha = 0.4$	-6.75	-5.84
$\alpha = 0.6$	-6.92	-6.31
$\alpha = 0.8$	-7.24	-6.96
$\alpha = 1.0$	-7.64	-7.64

MIBLP vs BeRRA: Runtime The runtime of the MIBLP formulation and my BeRRA algorithm is shown in Table 4.3. We can see from the table that BeRRA is much faster than MIBLP, and the difference becomes larger as the number of targets n increases. This is because solving MIBLP is difficult and the computational complexity increases exponentially with the problem size, while BeRRA only requires solving $\mathcal{O}(n \log(\frac{1}{\epsilon}) \log(\frac{1}{\delta}))$ linear feasibility problems. For MIBLP, it takes about 15 minutes for the very trivial case $n = 3$, which means it cannot scale up at all.

Table 4.3: MIBLP vs BeRRA in Runtime (s)

(a) $n = 2$

	MIBLP	BeRRA
$\alpha = 0$	70.4	0.95
$\alpha = 0.2$	71.2	0.95
$\alpha = 0.4$	72.0	0.94
$\alpha = 0.6$	68.9	0.77
$\alpha = 0.8$	73.4	0.48
$\alpha = 1.0$	64.4	0.20

(b) $n = 3$

	MIBLP	BeRRA
$\alpha = 0$	863.1	1.75
$\alpha = 0.2$	1004.4	1.63
$\alpha = 0.4$	958.8	1.53
$\alpha = 0.6$	886.9	1.36
$\alpha = 0.8$	1119.8	1.10
$\alpha = 1.0$	859.3	0.27

Runtime of BeRRA Figure 4.2 further analyzes the runtime of my BeRRA algorithm. m is set to be $n/2$ and all results are averaged over 100 instances. We observe that the runtime increases almost linearly with the number of targets n, and the game with 50 targets only takes about 2 minutes to solve, which demonstrates BeRRA's ability to scale up to larger problems. The figure also shows that the runtime does not change significantly with different α, but it decreases significantly when α is increased from 0.9999 to 1. This is because $|S_i(\mathbf{c}^{\max})|$ is almost 0 in zero-sum games.

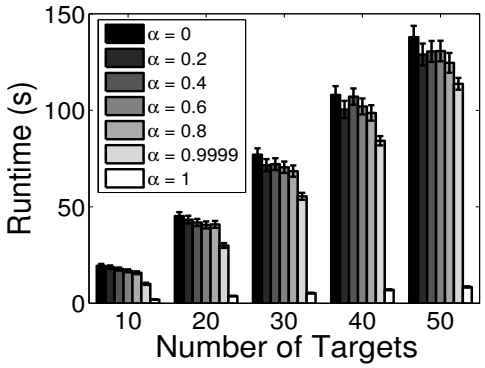


Figure 4.2: Runtime of BeRRA

4.7.2 Performance Evaluation of RSE strategy against Risk-averse Attackers

In this section, I will evaluate solution quality of the RSE strategy in detail. Since BeRRA shows advantages in both solution quality and runtime compared with the MIBLP formulation, I use BeRRA to evaluate the performance of RSE strategy.

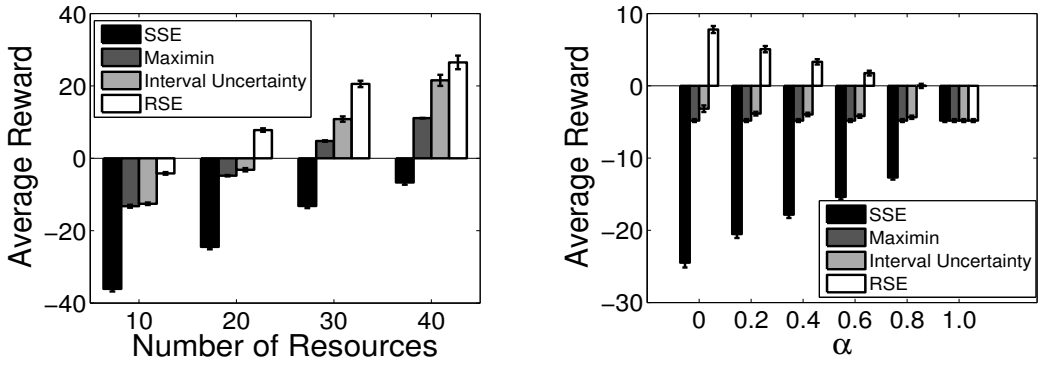
Solution Quality in Worst Case Figures 4.3(a) and 4.3(b) show the solution quality of RSE strategy in the worst case — the attacker attacks target $i = \arg\min_{j \in S_p(\mathbf{c})} U_d(j, \mathbf{c})$. I compare its performance with the SSE strategy, Maximin strategy and the robust strategy against interval uncertainty of $U_a^c(i)$ and $U_a^u(i)$ [19]. For values of the intervals, I tried different intervals ranging from 1 to 20 and pick the best one among them.

Figure 4.3(a) shows how the performance comparison changes with different number of resources m. The RSE strategy significantly outperforms all of the other strategies. Since the robust strategy against interval uncertainty considers some type of "robustness", it outperforms the SSE strategy and the Maximin strategy. However, since the interval uncertainty does not fully capture the risk aversion of the attacker, it is worse than the RSE strategy. The Maximin strategy is a more conservative strategy compared with the SSE strategy, leading to better performance when compared with SSE.

Figure 4.3(b) shows the performance comparison with different α. It shows the similar patten that RSE > Interval Uncertainty > Maximin > SSE as in Figure 4.3(a). Another observation is that the difference between these four strategies becomes less as α increases, and when $\alpha = 1$, these four strategies perform the same, as is proved in Theorem 4.6.1[5].

Solution Quality in Average Case Figures 4.4(a) and 4.4(b) show the solution quality of the RSE strategy in the average case — the attacker randomly attacks a target

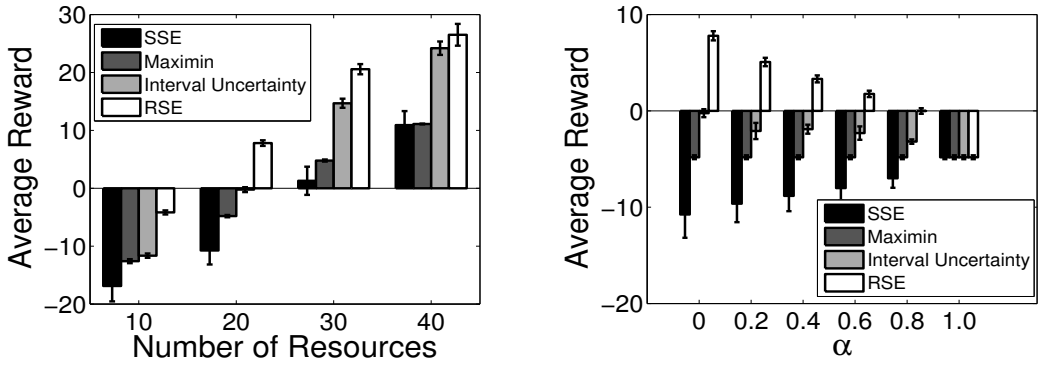
[5]Interval Uncertainty = Maximin can be proved with similar techniques in the proof of Theorem 4.6.1.



(a) $\alpha = 0, n = 50$ (b) $m = 20, n = 50$

Figure 4.3: Solution Quality of RSE in Worst Case

i in $S_p(\mathbf{c})$. I explore this case since unknown risk-averse attackers in the real world would not necessarily minimize the defender's reward. The performance comparison of these four strategies in the average case shows similar patterns compared with that in the worst case. Thus even in the average case, the RSE strategy still performs the best among them.



(a) $\alpha = 0, n = 50$ (b) $m = 20, n = 50$

Figure 4.4: Solution Quality of RSE in Average Case

"Price" of Being Robust Figure 4.5 compares the three strategies (SSE, Maximin, RSE) when the attacker is risk-neutral and arbitrarily bad risk-averse. SSE is the optimal

strategy when the attacker is risk-neutral and RSE is the optimal strategy when the attacker is arbitrarily bad risk-averse.

We can see from the figure that the performance of SSE strategy drops significantly if it wrongly estimates the attacker type (SSE-averse is very bad). However, for RSE strategy, its performance is only a little worse than the SSE strategy even if the attack is risk-neutral (compared with the bad performance of SSE-averse). This figure shows that compared with the significant loss of wrongly estimating attacker type, the loss of executing the robust strategy, which is the "price" of being robust, is acceptable.

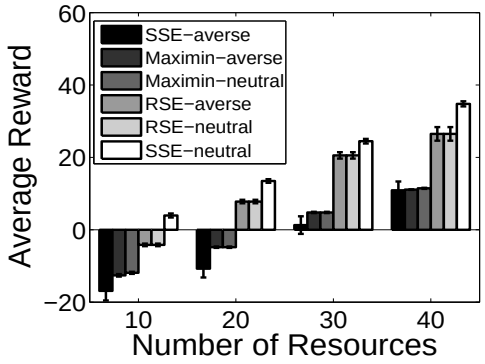
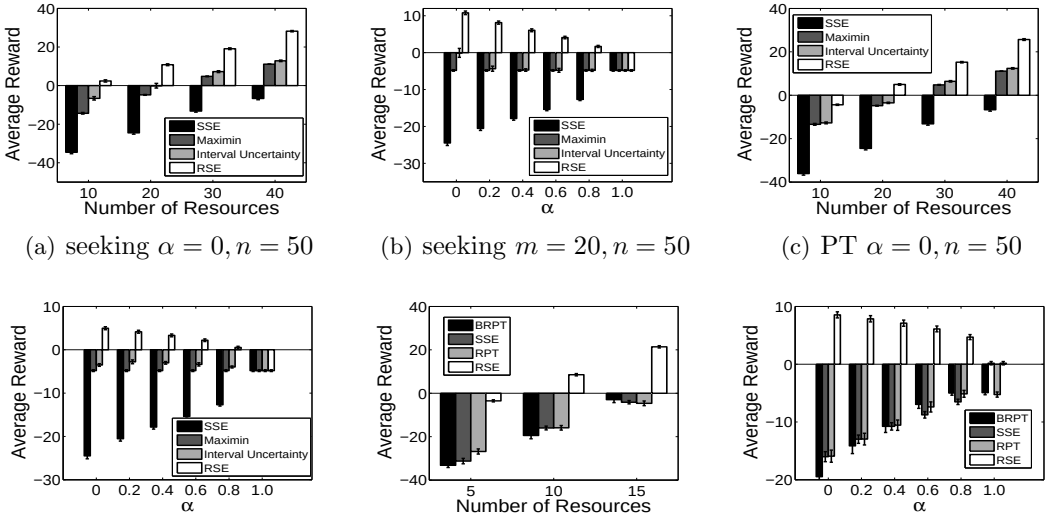


Figure 4.5: "Price" of Being Robust

4.7.3 Performance Evaluation of RSE strategy against other attacker types

In this section, I will evaluate solution quality of the RSE strategy against risk-seeking attackers and prospect theory attackers (S-shaped utility mapping function) in detail. Since the GMOP algorithm shows advantages in both solution quality and runtime compared with MIBLP, I use GMOP to evaluate the performance of RSE strategy.



(a) seeking $\alpha = 0, n = 50$ (b) seeking $m = 20, n = 50$ (c) PT $\alpha = 0, n = 50$

(d) PT $m = 20, n = 50$ (e) PT $\alpha = 0, n = 20$ (f) PT $m = 10, n = 20$

Figure 4.6: Solution Quality of RSE for Risk-aware Attackers

Figures 4.6(a) and 4.6(b) show the solution quality of RSE strategy against risk-seeking attackers. I compare its performance with the SSE strategy, Maximin strategy and the robust strategy against interval uncertainty of $U_a^c(i)$ and $U_a^u(i)$ [19]. For values of the intervals, I tried different intervals ranging from 1 to 20 and pick the best one.

Figure 4.6(a) shows how the performance comparison changes with different number of resources m. The RSE strategy significantly outperforms all of the other strategies. Since the robust strategy against interval uncertainty considers some type of "robustness", it outperforms the SSE strategy and the Maximin strategy. However, since the interval uncertainty does not fully capture the risk awareness of the attacker, it is worse than the RSE strategy. The Maximin strategy is a more conservative strategy compared with the SSE strategy, leading to better performance when compared with SSE.

Figure 4.6(b) shows the performance comparison with different α. The main observation is that the difference between these four strategies becomes less as α increases, and when $\alpha = 1$, these four strategies perform the same, as is proved in Theorem 4.6.1[6].

Figures 4.6(c) and 4.6(d) show the solution quality of RSE strategy against unknown prospect-theory attackers (S-shaped utility mapping function). They show similar patterns with the risk-seeking case.

Figure 4.6(e) and 4.6(f) show the performance comparison of the RSE strategy and the BRPT and RPT algorithm [59] against unknown prospect theory attackers. The RSE strategy computes the robust strategy against any S-shaped utility mapping attackers (risk-averse for gains and risk-seeking for losses). BRPT algorithm computes the defender's best response against a specific utility mapping function. RPT algorithm adds some "robustness" against interval uncertainty on the basis of the BRPT algorithm. We can see from Figure 4.6(e) and Figure 4.6(f) that the performance of BRPT and RPT is similar to that of SSE, and is much worse than the RSE strategy. The reason is that both BRPT and RPT fail to capture the uncertainty in the degree of risk-awareness.

[6]The proof of Interval Uncertainty = Maximin is similar to that of Theorem 4.6.1.

Chapter 5

Learning Attacker's Preference — Payoff Modeling

In some security domains, e.g., wildlife protection domain, the attacker's frequent attacks provide the defender with the opportunity to learn about the attacker's payoffs by observing the attacker's behavior. I begin with the assumption that at every round, the defender chooses one target to protect and the attacker simultaneously chooses one target to attack. Both the attacker and the defender have full knowledge about each other's previous actions (I will discuss my model and assumptions in more detail in Section 5.1). My work focuses on constructing an online policy for the defender to maximize her utility given observations of the attacker.

In this work, I model these interactions between the defender and the attacker as a repeated game. I then adopt a fixed model for the attacker's behavior and recast this repeated game as a partially observable Markov decision process (POMDP). However, my POMDP formulation has an exponential number of states, making current POMDP solvers like ZMDP [54] and APPL [25] infeasible in terms of computational cost. Silver and Veness [53] have proposed the POMCP algorithm which achieves a high level of performance in large POMDPs. It uses particle filtering to maintain an approximation

of the belief state of the agent, and then uses Monte Carlo Tree Search (MCTS) for online planning. However, the particle filter is only an approximation of the belief state. By appealing to the special properties of my POMDP, I propose the GMOP algorithm (**G**ibbs sampling based **M**CTS **O**nline **P**lanning) which draws samples directly from the exact belief state using Gibbs sampling and then runs MCTS for online planning. My algorithm provides higher solution quality than the POMCP algorithm. Additionally, for a specific subclass of my game with an attacker who plays a best response against the defender's empirical distribution, and a uniform penalty of being seized across all targets, I provide an advanced sampling technique to speed up the GMOP algorithm along with a heuristic that trades off solution quality for lower computational cost. Moreover, I explore the case of continuous utilities where my original POMDP formulation becomes a continuous-state POMDP, which is generally difficult to solve. However, the special properties in the specific subclass of game mentioned above make possible the extension of the GMOP algorithm to continuous utilities. Finally, I explore the more realistic scenario where the defender is not only uncertain about the distribution of resources, but also uncertain about the attacker's behavioral model. I address this challenge by extending my POMDP formulation and the GMOP algorithm.

The rest of the chapter is organized as follows: Section 5.1 discusses my model and POMDP formulation. Section 5.2 presents the GMOP algorithm in the basic scenario. Section 5.3 speeds up the GMOP algorithm by exploring special structure when the attacker plays a best response against the defender's empirical distribution. Section 5.4 extends my GMOP algorithm to the continuous utility scenario and Section 5.5 extends

the GMOP algorithm to the situation where the attacker model is also unknown. Section 5.6 provides extensive experimental results.

5.1 Model

5.1.1 Motivating Domain

My work is motivated by the domain of resource conservation, for example, illegal fishing, illegal oil extraction, water theft, crop theft, and illegal diamond mining, etc. In each case, illegal extractions happen frequently and the resources are spread over a large area that is impossible for the defender to cover in its entirety.

In this model, I make the assumption that the defender and the attacker fully observe their opponent's actions. The defender is usually a powerful government agency that has access to satellite imaging, multiple patrol assets, and the reports of local residents. The attacker learns about law enforcement tactics by exchanging information internally, covert observation, and by buying information from other sources.

The flagship example is the real-world problem faced by the U.S. Coast Guard (USCG) in the Gulf of Mexico of illegal fishing by fishermen from across-the-border. In this domain, the defender (USCG) performs daily aircraft patrol surveillance[1]; satellites are also used to monitor illegal fishing[2]. Furthermore, illegal fishermen have well-organized support from across-the-border; USCG provided evidence that fishermen perform surveillance on USCG boats.

[1] http://www.uscg.mil/d8/sectCorpusChristi/

[2] http://wwf.panda.org/?206301/WWF-new-approach-to-fight-illegal-unreported-and-unregulated-fishing

5.1.2 Formal Model

I now formalize the preceding story into a two-player repeated game between a defender and an attacker[3]. While both players are assumed to be humans or human organizations in this game, I assume that the defender is aided by my decision aid but the attacker is not. In my model, the amount of resources at each target will be fixed and the attacker will have full knowledge of this distribution. The defender will have to learn this distribution by observing the attacker's behavior.

I operate over the finite time horizon $t \in \mathbb{T} \triangleq \{1, \ldots, T\}$. There are n targets indexed by $\mathbb{N} \triangleq \{1, 2, \ldots, n\}$ that represent the locations of the natural resource in question: the attacker wants to steal resources from these targets and the defender wants to interdict the attacker. I represent the value of the targets to the attacker in terms of their utilities. Each target has a utility $u(i)$ that is only known to the attacker. The utility space is discretized into m levels, $u(i) \in \mathbb{M} \triangleq \{1, 2, \ldots, m\}$. Human beings cannot distinguish between tiny differences in utilities in the real world, so I am justified in discretizing these utilities. For n targets and m utility levels, there are m^n possible sets of utilities across all targets. The distribution of resources is then captured by the vector of utilities at each target, and the set of possible resource distributions is:

$$\mathbb{U} \triangleq \{(u(1), u(2), \ldots, u(n)) \,:\, u(i) \in \mathbb{M},\, \forall i \in \mathbb{N}\} = \times_{i \in \mathbb{N}} \mathbb{M}. \tag{5.1}$$

[3]Note that in this work I have begun with the assumption that there is a single extractor, and this already leads to very significant research challenges, needing significant research contributions, that I provide in this work. Generalizing to multiple extractors is clearly an important issue — that although will require me to scale-up my algorithm, fits naturally within my algorithmic framework – but it is left for future work.

At the beginning of the game, the defender may have some prior knowledge about the resource levels $u(i)$ at each target $i \in \mathbb{N}$. This prior knowledge is represented as a probability density function $p(u(i))$ over $\mathbb{M}$. If the defender does not know anything about $u(i)$, then I adopt a uniform prior for $u(i)$ over $\mathbb{M}$.

At each time $t \in \mathbb{T}$, the defender chooses a target $a_t \in \mathbb{N}$ to protect and the attacker simultaneously chooses a target $o_t \in \mathbb{N}$ from which to steal. If $a_t = o_t$, the defender catches the attacker and the attacker is penalized by the amount $P(o_t) < 0$; if $a_t \neq o_t$, the attacker successfully steals resources from target o_t and gets a payoff of $u(o_t)$. For clarity, the defender's interdiction is always successful whenever she visits the same site as the attacker. Additionally, the defender fully observes the moves of the attacker, likewise, the attacker fully observes the moves of the defender. Note that the penalty $P(i), i \in \mathbb{N}$ is known to both the defender and the attacker. I adopt a zero-sum game, so the defender is trying to minimize the attacker's payoffs. In most resource conservation domains, the attacker pays the same penalty P if he is seized independent of the target he visits. I allow for varying penalties across targets for greater generality.

In this work, I make the basic assumption that the attacker is more likely to steal from targets with higher utilities $u(i)$, lower penalties $P(i)$, and that have not been visited often by the defender. Based on this assumption, I assume that the attacker's actions depend on $u(i), P(i), i \in \mathbb{N}$ along with the defender's actions in previous rounds. A reasonable assumption about the attacker's behavioral model is the fictitious Quantal Response (FQR) model. Specifically, a fictitious attacker assumes the defender's empirical distribution will be his mixed strategy in the next round, and quantal response (QR) [31]

has been shown to be effective in describing human's behavior through human subject experiments.

For the FQR model, the attacker's behavior could be described in the following way: in every round, he (i) computes the empirical coverage probability c_i for every target i based on the history of the defender's actions; (ii) computes the expected utility $EU(i) = c(i)P(i) + (1 - c(i))u(i)$ for every target; (iii) attempts to steal from the target i with the probability $p(i)$ proportional to $e^{\lambda EU(i)}$:

$$p(i) = \frac{e^{\lambda EU(i)}}{\sum_{j \in \mathbb{N}} e^{\lambda EU(j)}},$$

where $\lambda \geq 0$ is the parameter representing the rationality of the player (higher λ represents a more rational player).

5.1.3 Protector's POMDP Formulation

To implement the model from Section 5.1.2, I must resolve two technical questions. First, at every round t, based on her current belief about u, how should the defender choose targets to protect in the next round? Second, after each round, how should the defender use the observation of the latest round to update her beliefs about u? I am studying decision making and belief updating in a partially observable environment where the payoffs u are unobservable and the attacker's actions are observable, which is the exact setup for a POMDP. I now setup my two-player game as a POMDP $\{\mathbb{S}, \mathbb{A}, \mathbb{O}, T, \Omega, R\}$ where the attacker follows a quantal response model.

State space: The state space of my POMDP is $\mathbb{S} = \mathbb{U} \times \mathbb{Z}^n$, which is the cross product of the utility space and the count space. $\mathbb{U}$ is the utility space as defined in Equation 5.1. $\mathbb{Z}^n$ is the set of possible counts of the defender's visits to each target, where $C_t \in \mathbb{Z}^n$ is an integer-valued vector where $C_t(i), i \in \mathbb{N}$ is the number of times that the defender has protected target i at the beginning of round $t \in \mathbb{T}$. A particular state $s \in \mathbb{S}$ is written as $s = (u, C)$, where u is the vector of utility levels for each target and C is the current state count. The initial beliefs are expressed by a distribution over $s = (u, 0)$, induced by the prior distribution on u. I define $c_t(i) \triangleq \frac{C_t(i)}{t-1}$ to be the frequency with which the defender visits target i at the beginning of round $t \in \mathbb{T}$. I set $c_1 \triangleq 0$ by convention.

Action space: The action space $\mathbb{A}$ is $\mathbb{N}$, representing the target the defender chooses to protect.

Observation space: The observation space $\mathbb{O}$ is $\mathbb{N}$, representing the target from which the attacker attempts to steal.

Conditional transition probability: Let $e_a \in \mathbb{R}^n$ denote the unit vector with a 1 in slot $a \in \mathbb{N}$ and zeros elsewhere. The conditional transition probability T governing the evolution of the state is

$$T\left(s' = \left(u',\, C'\right) | s = (u, C),\, a\right) = \begin{cases} 1, & u = u',\, C' = C + e_a, \\ 0, & \text{otherwise.} \end{cases}$$

Specifically, the evolution of the state is deterministic. The underlying utilities do not change, and the count for the target visited by the defender increases by one while all others stay the same.

Conditional observation probability: I define $EU(u, C) \in \mathbb{R}^n$ to be the vector of empirical expected utilities for the attacker for all targets when the actual utility is u and the count is C,

$$[EU(u, C)](i) = c(i)P(i) + (1 - c(i))u(i), \forall i \in \mathbb{N},$$

when $t \geq 1$. I set $[EU\,(u,\,0)]\,(i) = u(i)$ by convention. Hence, the observation probabilities Ω are explicitly

$$\Omega(o|s' = (u, C), a) = \frac{e^{\lambda[EU(u,C-e_a)](o)}}{\sum_{i\in\mathbb{N}} e^{\lambda[EU(u,C-e_a)](i)}},$$

the probability of observing the attacker takes action o when the defender takes action a and arrives at state s'. Note that both a and o are the actions the defender/attacker take at the same round.

Reward function: The reward function R is

$$R\left(s = (u, C), s' = (u, C + e_a), a, o\right) = \begin{cases} -P(o), & a = o, \\ -u\left(o\right), & a \neq o. \end{cases}$$

5.2 GMOP Algorithm

In Section 5.1, I modeled the repeated game as a POMDP in order to update the defender's beliefs about the resource distribution and to allocate patrol assets. However, the size of the utility space $\mathbb{U}$ is m^n, and the size of the count space is $\mathcal{O}(\frac{T^n}{n!})$. The computational cost of the latest POMDP solvers such as ZMDP and APPL soon becomes unaffordable as the problem size grows. For a small instance like $n = 4$, $m = 5$ and 5

rounds, there are 78750 states in the POMDP. Both the ZMDP and APPL solvers run out of memory when attempting to solve this POMDP. This challenge is non-trivial because the models in reality are much larger than this toy example.

Silver and Veness [53] have proposed the POMCP algorithm, which provides high quality solutions for large POMDPs. The POMCP algorithm uses a particle filter to approximate the belief state. Then, it uses Monte Carlo tree search (MCTS) for online planning where (i) state samples are drawn from the particle filter and (ii) the action with the highest expected utility based on Monte Carlo simulations is chosen. However, the particle filter is only an approximation of the true belief state and is likely to move further away from the actual belief state as the game goes on, especially when most particles get depleted and new particles need to be added. Adding new particles will either (i) make the particle filter a worse approximation of the exact belief state, if the added particles do not follow the distribution of the belief state or (ii) be as difficult as drawing samples directly from the belief state, if the added particles do follow the distribution of the belief state. However, if we could efficiently draw samples directly from the exact belief state, then there would be no need to use a particle filter.

This POMDP has specific structure that we can exploit. The count state in $\mathbb{S}$ is known and the utility state does not change, making it possible to draw samples directly from the exact belief state using Gibbs sampling. I propose the GMOP algorithm that draws samples directly from the exact belief state using Gibbs sampling, and then runs MCTS. The samples drawn directly from the belief state better represent the true belief state compared to samples drawn from a particle filter. I thus conjecture that the GMOP

algorithm will yield higher solution quality than the POMCP algorithm for the problem, and this intuition is confirmed in the experiments.

5.2.1 GMOP Algorithm Framework

The GMOP algorithm is outlined in Algorithm 5. At a high level, in round t the defender draws samples of state s from its belief state $B_t(s)$ using Gibbs sampling and then it uses MCTS to simulate what will happen in the next few rounds with those samples. Finally, it executes the action with the highest average reward in the MCTS simulation. MCTS starts with a tree that only contains a root node. Since the count state C_t is already known, the defender only needs to sample the utility state u from B_t. The sampled state s is comprised of the sampled utility u and the count C_t.

Algorithm 5 GMOP Algorithm Framework

```
1: function PLAY(C_t)
2:     Initialize Tree
3:     for i = 1 → numSamples do
4:         u ← GIBBSSAMPLING
5:         SIMULATE(s = (u, C_t))
6:     end for
7:     a_t ← action with the highest average reward
8: end function
```

It has been shown that the UCT algorithm converges to the optimal value function in fully observable MDPs [23]. Based on this result, Silver and Veness have established the convergence of MCTS in POMDP online planning as long as the samples are drawn from the true belief state $B_t(s)$. It follows that the convergence of the GMOP algorithm is guaranteed.

From Algorithm 5, we see that each iteration of the GMOP algorithm is composed of two parts: GIBBSSAMPLING which draws samples u directly from $B_t(u)$ using Gibbs

sampling, and SIMULATE which does Monte Carlo simulation of the sampled states $s = (u, C_t)$ to find the "best" action to execute. The sampling technique will be discussed in detail in Section 5.2.2 while the details of MCTS for POMDP are available in [53].

5.2.2 Drawing Samples

5.2.2.1 Gibbs Sampling Overview

Gibbs sampling [9] is a Markov chain Monte Carlo (MCMC) algorithm for sampling from multivariate probability distributions. Let $X = (x_1, x_2, \ldots, x_n)$ be a general random vector with n components and with finite support described by the multivariate probability density $p(X)$. Gibbs sampling only requires the conditional probabilities $p(x_i|x_{-i})$ to simulate X, where $x_{-i} = (x_j)_{j \neq i}$ denotes the subset of all components of X except component i. Gibbs sampling is useful when direct sampling from $p(X)$ is difficult.

Suppose we want to obtain k samples of $X = (x_1, x_2, \ldots, x_n)$. Algorithm 6 shows how Gibbs sampling works in general to produce these samples using only the conditional probabilities $p(x_i|x_{-i})$. It constructs a Markov chain whose steady-state distribution is given by $p(X)$, so that the samples we draw also follow the distribution $p(X)$. The states of this Markov chain are the possible realizations of $X = (x_1, x_2, \ldots, x_n)$, and a specific state X_i is denoted as $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ (there are finitely many such states by my assumption). The transition probabilities of this Markov chain, $Pr(X_j|X_i)$, follow from the conditional probabilities $p(x_i|x_{-i})$. Specifically, $Pr(X_j|X_i) = p(x_l|x_{-l})$ when $x_{jv} = x_{iv}$ for all v not equal to l, and is equal to zero otherwise, i.e., the state transitions only change one component of the vector-valued sample at a time. This Markov chain is

reversible (meaning $p(X_i)Pr(X_j|X_i) = p(X_j)Pr(X_i|X_j), \forall i, j$) so $p(X)$ is its steady-state distribution.

Algorithm 6 Gibbs Sampling

```
Initialization: X = {x_1, x_2, ..., x_n} satisfying p(X) > 0
for i = 1 → k do
    for j = 1 → n do
        x_j ~ p(x_j | x_{-j})
    end for
    X_i ← {x_1, x_2, ..., x_n}
end for
```

5.2.2.2 Applying Gibbs Sampling in GMOP

I let B_t be the probability distribution representing the defender's beliefs about the true utilities at the beginning of round $t \geq 1$; B_1 represents the defender's prior beliefs when the game starts. I adopt the notation $B_t(u)$ to denote the probability of the vector of utilities u with respect to the distribution B_t.

Let B be the prior belief distribution and B' be the posterior belief distribution. The Bayesian belief update rule to obtain B' from B and the observation is explicitly

$$\begin{aligned} B'(s' = (u, C)) =& \eta\Omega(o|s', a) \sum_{s \in \mathbb{S}} T(s'|s, a)B(s) \\ =& \eta\Omega(o|s', a)B(s = (u, C - e_a)). \end{aligned}$$

If a_t and o_t represent the actions that the defender and the attacker choose to take at round t, we have

$$\begin{aligned} B_t(u) =& \eta B_{t-1}(u)\Omega(o_{t-1}|s=(u,C_t),a_{t-1}) \\ =& \eta' B_1(u)\Pi_{i=1}^{t-1}\Omega(o_i|s=(u,C_{i+1}),a_i). \end{aligned} \tag{5.2}$$

It follows that the posterior belief B_t is proportional to the prior belief B_1 multiplied by the observation probabilities over the entire history. Since there are m^n possible utilities, it is impractical to store and update B_t when m and n are large, and thus it is impossible to sample directly from B_t. Hence, I turn to Gibbs sampling, where we only need the conditional probabilities $p(u_i|u_{-i}), \forall i \in \mathbb{N}$

$$\begin{aligned} & p(u_i|u_{-i}) = \eta p(u_i,u_{-i}) = \eta B_t(u_i,u_{-i}) \\ =& \eta' B_1(u_i,u_{-i})\Pi_{j=1}^{t-1}\Omega(o_j|s=(u=(u_i,u_{-i}),C_{j+1}),a_j) \\ =& \eta' B_1(u_i)B_1(u_{-i})\Pi_{j=1}^{t-1}\Omega(o_j|s=(u=(u_i,u_{-i}),C_{j+1}),a_j) \\ =& \eta'' B_1(u_i)\Pi_{j=1}^{t-1}\Omega(o_j|s=(u=(u_i,u_{-i}),C_{j+1}),a_j). \end{aligned} \tag{5.3}$$

This quantity is easy to compute where $B_1(u_i)$ is the prior probability that target i has utility u_i. In this way, we are able to draw samples directly from the exact belief state in this POMDP using Gibbs sampling. Thus, the GMOP algorithm has much better solution quality compared with the POMCP algorithm which draws samples from the approximate belief state maintained by particle filter.

Besides the conditional probability, we also need to find a valid u with $B_t(u) > 0$ to initialize Gibbs sampling as shown in Line 1 of Algorithm 6. Finding such a u is

easy in my FQR model because any u with $B_1(u) > 0$ satisfies $B_t(u) > 0$ since $B_t(u) = \eta' B_1(u) \Pi_{i=1}^{t-1} \Omega(o_i | s = (u, C_{i+1}), a_i)$ and $\Omega(o_i | s = (u, C_{i+1}), a_i) > 0, \forall i = 1, 2, ..., t-1$. In other behavior models, where finding a valid u is not so intuitive, one possibility is to check the sampled utilities at the latest round to pick a valid one.

5.3 Fictitious Best Response

In this section, I focus attention on a limiting case of the FQR model, a fictitious best response playing (FBR) attacker. An FBR attacker plays a best response against the empirical distribution of the defender and breaks ties randomly, a similar assumption is found in [26, 30]. Additionally, I assume that all targets share the same penalty P. This assumption is satisfied in most resource conservation games. We will see that these two assumptions allow us to greatly speed up the GMOP algorithm. I also put forward a computationally inexpensive heuristic that offers high quality solutions.

When the attacker is FBR, the POMDP is roughly the same as in the FQR case except that the conditional observation probabilities Ω are now

$$\Omega(o | s' = (u, C), a) = \begin{cases} \frac{1}{|A(u, C - e_a)|}, & o \in A\left(u, C - e_a\right), \\ 0, & \text{otherwise}, \end{cases} \tag{5.4}$$

where $A\left(u,\, C\right)$ is the set of targets with maximal empirical expected utility when the actual utility is u and the count is C, i.e.,

$$A\left(u,\, C\right) = \arg\max_{i \in \mathbb{N}} \left[EU\left(u,\, C\right)\right](i) \subset \mathbb{N}.$$

The FBR attacker is actually a limiting case of the more general FQR model: we obtain this case by taking $\lambda \to \infty$. If we run the POMCP algorithm for an FBR attacker, the particles produced by the particle filter will be depleted very quickly and most utility states will take on probability 0 after only a few rounds. For example, if $n = 10$ and the defender observes that the attacker visits target 3 in the first round, then approximately 90% of possible utility states take on probability 0. Compared with FQR, more new particles must be added in the FBR case. Thus, the particle filter is a worse approximation of the belief state, leading to worse performance of the POMCP algorithm.

5.3.1 Speeding Up GMOP

Gibbs sampling requires computation of the conditional probability $p(u_i|u_{-i})$ as described in Equation 5.3. However, t grows as the game evolves and the computational cost increases linearly with t. Under the assumptions of an FBR attacker and uniform penalties, we can use an advanced algorithm to compute $p(u_i|u_{-i})$ with computational cost bounded by constant time.

Define

$$I_t(i,j) \triangleq \begin{cases} I_{t-1}(i,j), & i \neq o_{t-1}, \\ \max\{I_{t-1}(i,j), \frac{1-c_{t-1}(j)}{1-c_{t-1}(i)}\}, & i = o_{t-1}, \end{cases}$$

and $I_1(i,j) \triangleq 0, \forall i,j \in \mathbb{N}$. The quantities $I_t(i,j)$ can be computed recursively from $I_{t-1}(i,j)$ at very little computational cost. Intuitively, $I_t(i,j)$ maintains the minimum allowed ratio $\frac{u(i)-P}{u(j)-P}$ for any u satisfying $\Pi_{j=1}^{t-1}\Omega(o_j|s = (u, C_{j+1}), a_j) > 0$ as the game evolves. By checking if u satisfies $\frac{u(i)-P}{u(j)-P} \geq I_t(i,j), \forall i,j \in \mathbb{N}$, we can figure out if

$\Pi_{j=1}^{t-1}\Omega(o_j|s = (u, C_{j+1}), a_j)$ is equal to 0 or not. I then compute the exact value of $\Pi_{j=1}^{t-1}\Omega(o_j|s = (u, C_{j+1}), a_j)$ whenever this probability is not 0.

Proposition 5.3.1. *For a specific u, $\Pi_{i=1}^{t-1}\Omega(o_i|s = (u, C_{i+1}), a_i) > 0 \iff \frac{u(i)-P}{u(j)-P} \geq I_t(i,j), \forall i, j \in \mathbb{N}$.*

Proof. From Equation 5.3 and 5.4, $\Pi_{j=1}^{t-1}\Omega(o_j|s = (u, C_{j+1}), a_j) > 0 \iff o_j \in A(u, C_j), \forall j \in \{1, 2, \ldots, t-1\}$.

$o \in A(u, C) \iff [EU(u, C)](o) \geq [EU(u, C)](i), \forall i \in \mathbb{N}$

$$c(o)P + (1 - c(o))u(o) \geq c(i)P + (1 - c(i))u(i), \forall i \in \mathbb{N}$$
$$\frac{u(o) - P}{u(i) - P} \geq \frac{1 - c(i)}{1 - c(o)}, \forall i \in \mathbb{N}$$

$\forall u$ that $\frac{u(i)-P}{u(j)-P} \geq I_t(i,j), \forall i, j \in \mathbb{N}$, we have $o_j \in A(u, C_j), \forall j \in \{1, 2, \ldots, t-1\}$ by the definition of $I_t(i,j)$; $\forall u$ that $\exists i, j \in \mathbb{N}$ $\frac{u(i)-P}{u(j)-P} < I_t(i,j)$, for that i, j, $\exists$ round $k \in \{1, 2, \ldots, t-1\}$ that $\frac{1-c_k(j)}{1-c_k(i)} = I_t(i,j)$ and $i = o_k$, so we have $o_k \notin A(u, C_k)$ because $\frac{u(i)-P}{u(j)-P} < I_t(i,j) = \frac{1-c_k(j)}{1-c_k(i)}$. Here I proved $o_j \in A(u, C_j), \forall j \in \{1, 2, \ldots, t-1\} \iff \frac{u(i)-P}{u(j)-P} \geq I_t(i,j), \forall i, j \in \mathbb{N}$. □ □

By checking if u satisfies $\frac{u(i)-P}{u(j)-P} \geq I_t(i,j), \forall i, j \in \mathbb{N}$, we can figure out if $\Pi_{j=1}^{t-1}\Omega(o_j|s = (u, C_{j+1}), a_j)$ equals 0. I now explain how to compute $\Pi_{j=1}^{t-1}\Omega(o_j|s = (u, C_{j+1}), a_j)$ if it does not equal 0. Define $V_t(i) \triangleq \{k : o_k = i, k \in \{1, 2, \ldots, t-1\}\}, \forall i \in \mathbb{N}$ to be the set of rounds where the attacker attempts to steal from target i; define $V_t^{eq}(i,j) \triangleq \{k : j \in A(u, C_k), k \in V_t(i)\}, \forall i, j \in \mathbb{N}$ to be the set of rounds where the attacker attempts to steal from target i, but where target j gives the attacker the same expected utility. I define

$V_t^{neq}(i,j) \triangleq \{k : j \notin A(u, C_k), k \in V_t(i)\}, \forall i, j \in \mathbb{N}$ to be the set of rounds where the attacker attempts to steal from target i and target j gives the attacker lower expected utility. Additionally, I define

$$\text{Tie}_t(i,j) \triangleq \{k : I_t(i,j) = \frac{1-c_k(j)}{1-c_k(i)}, k \in V_t(i)\}, \forall i, j \in \mathbb{N}$$

Like $I_t(i,j)$, $\text{Tie}_t(i,j)$ can be computed recursively at very little cost. By definition, $V_t^{eq}(i,j) \cap V_t^{neq}(i,j) = \phi$, $V_t^{eq}(i,j) \cup V_t^{neq}(i,j) = V_t(i)$ and $\text{Tie}_t(i,j) \subseteq V_t(i)$, $\forall i, j \in \mathbb{N}$.

Proposition 5.3.2. *If* $\frac{u(i)-P}{u(j)-P} = I_t(i,j)$, $V_t^{eq}(i,j) = Tie_t(i,j)$, $V_t^{neq}(i,j) = V_t(i) - Tie_t(i,j)$; *If* $\frac{u(i)-P}{u(j)-P} > I_t(i,j)$, $V_t^{neq}(i,j) = V_t(i)$, $V_t^{eq}(i,j) = \phi$.

Proof. If $\frac{u(i)-P}{u(j)-P} = I_t(i,j)$:

$\forall k \in V_t^{eq}(i,j)$, $c_k(i)P + (1-c_k(i))u(i) = c_k(j)P + (1-c_k(j))u(j)$ since $i, j \in A(u, C_k)$, so $\frac{1-c_k(j)}{1-c_k(i)} = \frac{u(i)-P}{u(j)-P} = I_t(i,j)$, $k \in \text{Tie}_t(i,j)$. So we have $V_t^{eq}(i,j) \subseteq \text{Tie}_t(i,j)$

$\forall k \in \text{Tie}_t(i,j)$, $\frac{u(i)-P}{u(j)-P} = I_t(i,j) = \frac{1-c_k(j)}{1-c_k(i)}$, so $c_k(i)P + (1-c_k(i))u(i) = c_k(j)P + (1-c_k(j))u(j)$, $j \in A(u, C_k)$. So we have $\text{Tie}_t(i,j) \subseteq V_t^{eq}(i,j)$.

Till now I proved when $\frac{u(i)-P}{u(j)-P} = I_t(i,j)$, $V_t^{eq}(i,j) = \text{Tie}_t(i,j)$, so $V_t^{neq}(i,j) = V_t(i) - \text{Tie}_t(i,j)$ by definition. The proof when $\frac{u(i)-P}{u(j)-P} > I_t(i,j)$ is similar so I omit it here. □

□

Algorithm 7 shows how my advanced sampling technique resamples $u(k)$ from the conditional probabilities $p(u_i|u_{-i})$ by using the quantities I_t and Tie_t. The input u is the current set of sampled utilities; k is the index of u to be resampled according to $p(u_k|u_{-k})$; and I and 'Tie' are the latest I and 'Tie' that have been computed recursively.

I set $\#A(j) = |A(u, C_j)|$ to denote the number of sites that have maximal expected utility for the attacker at round j, and I initialize these quantities to be 1 because $o_k \in A(u, C_k)$ by definition. Then, I check every pair of targets $i, j \in \mathbb{N}$: (i) if $\frac{u(i)-P}{u(j)-P} < I_t(i, j)$, then I set $B_t(u) = 0$ according to Proposition 5.3.1; (ii) if $\frac{u(i)-P}{u(j)-P} = I_t(i, j)$, then I set $V_t^{eq}(i, j) = \text{Tie}_t(i, j)$ according to Proposition 5.3.2, and I increase $\#A(k)$ by 1 for those $k \in \text{Tie}_t(i, j)$ because $j \in A(u, C_k), \forall k \in V_t^{eq}(i, j) = \text{Tie}_t(i, j)$; (iii) if $\frac{u(i)-P}{u(j)-P} > I_t(i, j)$, then $V_t^{eq}(i, j) = \phi$ according to Proposition 5.3.2, so I do nothing. After checking all pairs $i, j \in \mathbb{N}$, I determine: (i) whether $B_t(u) = 0$ and (ii) $\#A(k), \forall k \in \{1, 2, \ldots, t-1\}$ if $B_t(u) > 0$. Based on these evaluations, the conditional probability $Prob = p(u_i|u_{-i})$ used to resample $u(k)$ is computed according to Equation 5.3. Finally, $Prob$ is normalized and then I sample the new $u(k)$.

Algorithm 7 Advanced Sampling Technique

```
1: function DrawSample(u, k, I, Tie)
2:     Prob = B_1(u_k)
3:     for i = 1 → m do
4:         u(k) ← i
5:         #A(j) ← 1, ∀j = 1 → currRound − 1
6:         for p = 1 → n, q = 1 → n do
7:             if (u(p)−k)/(u(q)−k) < I(p, q) then
8:                 Prob(i) ← 0
9:                 break
10:            else if (u(p)−k)/(u(q)−k) = I(p, q) then
11:                #A(j) ← #A(j) + 1, ∀j ∈ Tie(p, q)
12:            end if
13:        end for
14:        if Prob(i) ≠ 0 then
15:            Prob(i) ← Prob(i) * Π_{j=1}^{currRound−1} 1/#A(j)
16:        end if
17:    end for
18:    Normalize Prob
19:    u(k) ∼ Prob
20:    return u
21: end function
```

5.3.2 Myopic Planning Heuristic

For GMOP algorithm, larger sample sizes in MCTS leads to higher solution quality but at the expense of greater computational cost. Some domains require decisions to be made very quickly, so the defender may get poor performance with the GMOP algorithm due to an insufficient number of samples. With this motivation, I provide a myopic planning heuristic. This heuristic offers slightly lower solution quality compared with GMOP, but costs much less computing time.

The myopic planning heuristic works as follows: it (i) approximately computes the posterior marginal probabilities of all targets' utilities based on all previous observations; (ii) computes the expected $u(i)$ for each target using the posterior marginal probabilities; (iii) plans myopically—protects the target with the highest estimated expected utility for the attacker based on the expected $u(i)$ computed in step (ii) and the empirical visit counts C (ties are broken with even probabilities).

The key issue lies in step (i)—the computation of the posterior marginal probabilities of the utilities $u(i)$. This step can be viewed as inference in a Bayesian network. An example Bayesian network where $n = 4$ is shown in Figure 5.1. Here, $u(i), \forall i \in \mathbb{N}$ are treated as the unobserved random variables in the Bayesian network, and they have prior probabilities $B_1(u(i))$ for all $i \in \mathbb{N}$. I define $f(u(i), u(j)), \forall i, j \in \mathbb{N}$ to be observable binary random variables that depend on $u(i), u(j)$:

$$f(u(i), u(j)) \triangleq \begin{cases} 1, & \frac{u(i)-P}{u(j)-P} \geq I_t(i,j), \frac{u(j)-P}{u(i)-P} \geq I_t(j,i), \\ 0, & \text{otherwise.} \end{cases}$$

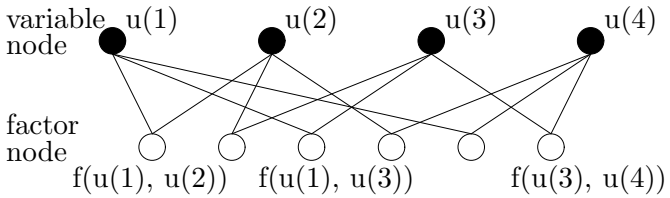


Figure 5.1: Bayesian Network when $n = 4$

In the Bayesian Network, we have observations of $f(u(i), u(j)) = 1, \forall i, j \in \mathbb{N}$, and the aim is to infer the posterior marginal probabilities for $u(i), \forall i \in \mathbb{N}$. According to Proposition 5.3.1, these factor nodes $f(u(i), u(j)) = 1, \forall i, j \in \mathbb{N}$ fully describe the conditions a specific u must satisfy in order to have a positive posterior probability. I then use the widely known belief propagation algorithm [43] for inference, which yields approximate marginal probabilities.

Note that this heuristic does not take into consideration possible ties in the attacker's decision-making. In particular, recall that $\frac{u(i)-P}{u(j)-P} = I_t(i,j)$ and $\frac{u(i)-P}{u(j)-P} > I_t(i,j)$ correspond to two different cases, as I have shown in Proposition 5.3.2, and they are treated separately in Algorithm 7. Yet, the Bayesian network is unable to distinguish between these two cases and it treats them both as $\frac{u(i)-P}{u(j)-P} \geq I_t(i,j)$. Hence, the Bayesian network does not utilize all of the information that the defender has obtained and thus cannot offer an accurate description of the true belief state. Subsequently, the posterior probability I compute from the Bayesian network is inaccurate even if an exact inference algorithm is used. However, I note that in the experiments, I get satisfactory solution quality even though both the Bayesian network formulation and the belief propagation algorithm are inexact.

5.4 Continuous Utility Scenario

In previous sections, I discussed the model with discretized utilities with the justification that humans can not distinguish between tiny differences, and this can be captured in a discrete model with a sufficiently large m. However, it is difficult to tell how "large" is large enough; furthermore, larger m leads to more computational cost so that we can not increase m arbitrarily. In this section, I try to extend my model to continuous utilities, i.e., $u(i) \in [0, 1]$, making it more expressive in describing human's perception of utilities.

In Section 5.1.3, the POMDP was built for discrete utilities. When the utilities are continuous, this formulation remains the same except that the utility space $\mathbb{U}$ becomes

$$\mathbb{U} \triangleq \{(u(1), u(2), \ldots, u(n)) \ : \ u(i) \in [0, 1], \forall i \in \mathbb{N}\}$$

which is in continuous space. Thus, the previous POMDP formulation becomes a continuous-state POMDP, which lacks efficient solutions.

The GMOP algorithm is composed of two steps: sampling from the utility space and running MCTS with those samples. For continuous utilities, the latter step remains the same so that the key issue here is to sample from the continuous utility space, which involves the computation of the conditional probability:

$$p(u_i|u_{-i}) = \eta'' B_1(u_i) \Pi_{j=1}^{t-1} \Omega(o_j|s = (u = (u_i, u_{-i}), C_{j+1}), a_j).$$

In general cases, this computation involves the multiplication of several functions ($\Pi_{j=1}^{t-1} \Omega(o_j|s = (u = (u_i, u_{-i}), C_{j+1}), a_j)$ with u_i as the variable), which is generally hard to compute

unless those functions have special properties, e.g., when the attacker is a FBR attacker and all sites share the same penalty P.

Proposition 5.4.1. *When the attacker is an FBR player and all sites share the same penalty P, $\Pi_{j=1}^{t-1}\Omega(o_j|s=(u=(u_i,u_{-i}),\ C_{j+1}),a_j)$ is a boxcar function[4] with non-zero interval $[a,b]$, where $a=\max_{j\in\mathbb{N},j\neq i}\{P+I_t(i,j)(u(j)-P)\}$, $b=\min_{j\in\mathbb{N},j\neq i}\{P+\frac{u(j)-P}{I_t(j,i)}\}$, and the height at $[a,b]$ equals $\frac{1}{b-a}$.*

Proof. When the attacker is an FBR player and all sites share the same penalty P, $\Omega(o_j|s=(u=(u_i,u_{-i}),C_{j+1}),a_j)$ are all boxcar functions, so that their multiplication is also a boxcar function.

Recall Proposition 5.3.1:

$$\Pi_{i=1}^{t-1}\Omega(o_i|s=(u,C_{i+1}),a_i)>0 \iff \frac{u(i)-P}{u(j)-P}\geq I_t(i,j),\forall i,j\in\mathbb{N}$$

In this situation, u_{-i} is given, and we are trying to find the smallest and largest u_i satisfying $\Pi_{i=1}^{t-1}\Omega(o_i|s=(u,C_{i+1}),a_i)>0$, i.e., $\frac{u(i)-P}{u(j)-P}\geq I_t(i,j),\forall i,j\in\mathbb{N}$. So we have:

$$u_i \geq P+(u_j-P)I_t(i,j),\forall j\in\mathbb{N},j\neq i,$$
$$u_i \leq P+\frac{u_j-P}{I_t(j,i)},\forall j\in\mathbb{N},j\neq i.$$

Since the total area under the boxcar function is 1 (it is a probability distribution), the height is $\frac{1}{b-a}$. □ □

[4] A boxcar function is any function which is zero over the entire real line except for a single interval where it is equal to a constant.

With Proposition 5.4.1, we can compute $\Pi_{j=1}^{t-1}\Omega(o_j|s = (u = (u_i, u_{-i}), C_{j+1}), a_j)$ very efficiently by only computing the lower limit a and upper limit b, making it possible for us to draw samples from the continuous state space using Gibbs sampling and then to search for the best strategy with MCTS.

Finally, clearly at this stage the continuous utility scenario only works in restricted cases; nonetheless, the restriction of all sites sharing a single penalty is reasonable in some real-world domains, and the FBR restriction on the extractor may be a useful approximation for some situations. Understanding the appropriate use of the continuous utility scenario given the need to model human extractors, scaling it up and relaxing the restrictions imposed all remain a topic for future work.

5.5 Unknown Extractor Scenario—Model Ensemble

The discussions in previous sections are based on the assumption that the attacker is an FQR attacker, and the defender knows the true attacker model. However, in real-world applications, the attacker may follow other behavioral models, and the defender may only have some rough idea what the behavioral model is instead of knowing it exactly. In this section, I will firstly discuss how to extend the POMDP formulation and GMOP algorithm to other behavioral models, and then discuss the scenario where the defender is uncertain about the attacker's behavioral model.

5.5.1 Dealing with Other Behavioral Models

In addition to the FQR model, there are other human behavioral models that might well describe the attacker's actions. For example, the attacker may have limited memory or

weigh recent defender activities more heavily, and optimizes against this limited memory or "biased" memory. For most generality, I only keep the very basic assumption that the attacker's decisions depend on $u(i), P(i), i \in \mathbb{N}$ along with the defender's actions in previous rounds, i.e., the probability that the attacker visits target $i, i \in \mathbb{N}$ is a function of u, P, and a. In this section, I will discuss how the POMDP formulation and GMOP algorithm can be modified accordingly to deal with this broader category of behavioral models.

With this broader category of behavioral models, the attacker's decision making depends on the sequence of the defender's actions more than the count C, the previous state $\mathbb{S} = \mathbb{U} \times \mathbb{Z}^n$ is no longer enough to determine the attacker's actions. In response, I modify the previous POMDP formulation using a more expressive state space.

State space: The state space of the POMDP becomes $\mathbb{S} = \mathbb{U} \times \{\mathbb{N}^i : i \in \{0, 1, 2, \ldots, T\}\}$. $\mathbb{U}$ is still the utility space and $\{\mathbb{N}^i : i \in \{0, 1, 2, \ldots, T\}\}$ is now the entire history of the defender's actions, where $h \in \{\mathbb{N}^i : i \in \{0, 1, 2, \ldots, T\}\}$ is a vector where $h(t), t \in \mathbb{T}$ is the target the defender visits at round $t \in \mathbb{T}$. A particular state $s \in \mathbb{S}$ is written as $s = (u, h)$, where u is the vector of utility levels for each target and h is the current history of the defender's actions. The initial beliefs are expressed by a distribution over $s = (u, \emptyset)$, induced by the prior distribution on u.

Action space: The action space $\mathbb{A}$ remains $\mathbb{N}$, representing the target the defender chooses to protect.

Observation space: The observation space $\mathbb{O}$ remains $\mathbb{N}$, representing the target from which the attacker attempts to steal.

Conditional transition probability: The conditional transition probability is modified to describe how the defender's action history evolves.

$$T\left(s' = \left(u',\, h'\right) | s = (u, h)\,,\, a\right) = \begin{cases} 1, & u = u',\, h' = h + a, \\ 0, & \text{otherwise.} \end{cases}$$

The notation "+" here means appending a at the end of h.

Conditional observation probability: Suppose the behavioral model function $f(u, P, h, o)$ determines the attacker's probability of stealing from target o with the utilities u, the penalties P and the history h, the conditional observation probability is defined as

$$\Omega(o|s' = (u, h), a) = f(u, P, h - a, o).$$

The notation $-$ here means removing a from the end of h.

Reward function: The reward function R becomes

$$R\left(s = (u, h), s' = (u, h + a), a, o\right) = \begin{cases} -P(o), & a = o, \\ -u\left(o\right), & a \neq o. \end{cases}$$

With the new definition of POMDP, the GMOP algorithm remains the same except that the computation of the conditional probability during Gibbs sampling becomes

$$p(u_i|u_{-i}) = \eta B_1(u_i)\Pi_{j=1}^{t-1}\Omega(o_j|s = (u = (u_i, u_{-i}), h_{j+1}), a_j). \tag{5.5}$$

5.5.2 Dealing with Unknown Extractor

In the real world, it is more likely that the defender only has some rough idea what the attacker's behavioral model is rather than knowing it exactly. In this section, I extend my model to this scenario. To formally define the problem, I model the defender's "rough idea" of the attacker's behavioral model as a set of candidate behavioral models, and the defender is uncertain about which model best fits the attacker's behavior. As the game goes on, the defender gets a better idea about the utilities of different targets as well as the attacker's behavioral model.

Suppose that there are k candidate attacker behavioral models Type 1, Type 2, ..., Type k that might model the attacker's behavior. I define the attacker's behavioral model space $\mathbb{B}$ to be the set of these k behavioral models:

$$\mathbb{B} = \{\text{Type 1}, \text{Type 2}, ..., \text{Type k}\}.$$

When the defender has perfect knowledge about the attacker's behavioral model, I formulate a POMDP formulation where the state space is the cross product of the utility space and the defender's action history space: $\mathbb{S} = \mathbb{U} \times \{\mathbb{N}^i : i \in \{0, 1, 2, \ldots, T\}\}$ since they are the only two variables that determine the attacker's actions. However, when the defender is uncertain what the attacker's behavioral model is, the attacker's action is also dependent on his behavioral model in addition to these two variables. Thus, the attacker's behavioral model should also be included in the state space of the POMDP formulation.

State space: The state space becomes the cross product of the behavioral model space $\mathbb{B}$, the utility space $\mathbb{U}$ and the history space $\{\mathbb{N}^i : i \in \{0, 1, 2, \ldots, T\}\}$: $\mathbb{S} = \mathbb{B} \times \mathbb{U} \times \mathbb{N}^i$. A particular state $s \in \mathbb{S}$ is written as $s = (b, u, h)$ where $b \in \mathbb{B}$ represents the attacker's behavioral model.

Action space: The action space $\mathbb{A}$ remains $\mathbb{N}$, representing the target the defender chooses to protect.

Observation space: The observation space $\mathbb{O}$ remains $\mathbb{N}$, representing the target from which the attacker attempts to steal.

Conditional transition probability: The conditional transition probability is modified accordingly.

$$T\left(s' = \left(b', u', h'\right) | s = (b, u, h)\,,\, a\right) = \begin{cases} 1, & b = b', u = u', h' = h + a, \\ 0, & \text{otherwise.} \end{cases}$$

The notation $+$ here means appending a at the end of h.

Conditional observation probability: Suppose the behavioral model function $f(b, u, P, h, o)$ determines the attacker's probability of stealing from target o with the behavioral model b, the utilities u, the penalties P and the history h, the conditional observation probability is defined as

$$\Omega(o|s' = (b, u, h), a) = f(b, u, P, h - a, o).$$

The notation $-$ here means removing a from the end of h.

Reward function: The reward function R becomes

$$R\left(s=(b,u,h),s'=(b,u,h+a),a,o\right)=\begin{cases}-P(o), & a=o,\\ -u\left(o\right), & a\neq o.\end{cases}$$

For this new POMDP, we need to draw samples from the behavioral model space in addition to the utility space for GMOP algorithm. We can combine the behavioral model variable b and the utility variable u together to be a new multivariate. Gibbs sampling is then used to draw samples of b and u, which are fed into MCTS to find the optimal action for the defender to take. In Gibbs sampling, the computation of the conditional probability for utility variable u_i becomes

$$p(u_i|u_{-i},b)=\eta B_1(u_i)\Pi_{j=1}^{t-1}\Omega(o_j|s=(b,u=(u_i,u_{-i}),h_{j+1}),a_j). \tag{5.6}$$

Similarly, the computation of the conditional probability for behavioral model variable b is

$$\begin{aligned}&p(b|u)=\eta p(b,u)=\eta B_t(b,u)\\ =&\eta' B_1(b,u)\Pi_{j=1}^{t-1}\Omega(o_j|s=(b,u,h_{j+1}),a_j)\\ =&\eta'' B_1(b)\Pi_{j=1}^{t-1}\Omega(o_j|s=(b,u,h_{j+1}),a_j).\end{aligned} \tag{5.7}$$

In the original formulation, the multivariate we sample in Gibbs sampling is the n-dimensional utility variable u. In this extension, the multivariate is a combination of the behavioral model variable b and the utility variable u, which is of the dimension $n+1$.

Thus, this extension to unknown extractor costs roughly $\frac{n+1}{n}$ times the computational time in the original formulation, and the extra time is used to compute $p(b|u)$.

5.6 Experimental Evaluation

I evaluate the performance of my models and algorithms in this section through extensive numerical experiments. The results strongly support the benefits of the techniques introduced in this work. For the experiment settings, unless stated otherwise, I follow: $n = 10$, $m = 10$, the penalty across all targets is $P = -50$, and the prior probability distribution $B_1(u_i), i \in \mathbb{N}$ is uniform. All results are averaged over 1000 simulation runs. For each simulation run, I randomly draw the true utilities $u(i), i \in \mathbb{N}$ and then simulate the actions the defender and the attacker would take over the rounds of the game. Solution quality is assessed in terms of the average reward that the defender gets in the first few rounds of the game. There are two parameters in MCTS for the GMOP algorithm: $numSamples$ is the number of samples that are used to simulate in the MCTS and $maxHorizon$ is the depth of the tree, i.e., the number of time steps we look ahead in the POMDP.

5.6.1 GMOP Algorithm Evaluation

In this section, I will use the FBR and FQR attacker model to evaluate the performance of the GMOP algorithm in both the discrete and continuous utility scenario.

5.6.1.1 GMOP vs ZMDP/APPL

To begin, I compare the GMOP algorithm with the ZMDP solver [54] and the APPL solver [25], both are general POMDP solvers. I show that the GMOP algorithm achieves almost the same solution quality as ZMDP/APPL solvers on small problem instances. For a small problem instance like $n = 4$, $m = 5$ and total rounds $maxHorizon = 5$, there are 78750 states in the POMDP. Both ZMDP and APPL solvers run out of memory even in this small problem instance. Hence, I test the two solvers together on an even smaller instance with $n = 3$, $m = 5$, $P = -10$ and $maxHorizon = 5$, so that the resulting POMDP has only 7000 states. As a base line, I also include a fixed policy where the defender randomly chooses one site to protect at each round. Table 5.1 reports the average reward of these three algorithms for both the FQR ($\lambda = 0.5, 1$ and 1.5) and FBR attacker. In this table, the columns titled H_i for $i = 1, \ldots, 5$ represent the GMOP algorithm with $maxHorizon$ set to be i and $numSamples$ set to be 10000. We see that the GMOP algorithm with $maxHorizon$ varying from 1 to 5 and the APPL/ZMDP solvers are very close in terms of average reward, and all algorithms outperform the random policy for the FQR and FBR attacker models.

Table 5.1: ZMDP/APPL vs GMOP in Solution Quality

	Random	ZMDP	APPL	H_1	H_2	H_3	H_4	H_5
FQR(0.5)	1.13	3.85	3.85	3.90	3.89	3.95	3.90	3.91
FQR(1)	1.05	4.84	4.81	4.75	4.80	4.87	4.97	4.79
FQR(1.5)	1.03	5.35	5.39	5.35	5.36	5.42	5.36	5.34
FBR	1.09	6.32	6.31	6.25	6.24	6.27	6.32	6.36

5.6.1.2 Analysis of GMOP

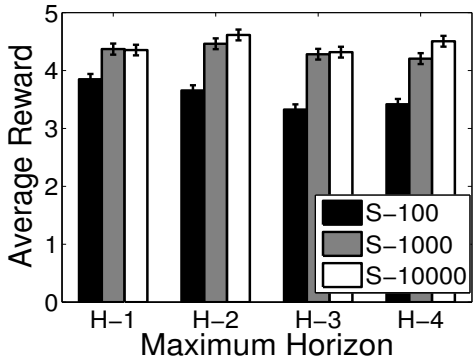
Now I investigate the effect of $numSamples$ and $maxHorizon$ on the performance of MCTS in the GMOP algorithm for discrete utilities. Figure 5.2(a)/5.2(b) reports the results for the FQR model while Figure 5.4(a)/5.4(b) reports the results for the FBR model. Figure 5.2(a)/5.4(a) show that the performance of MCTS improves as I increase $numSamples$ while holding $maxHorizon$ fixed, demonstrating the convergence of MCTS. Figures 5.2(b)/5.4(b) together show that: (i) if $numSamples$ is large enough to ensure convergence for both larger $maxHorizon$ and smaller $maxHorizon$ ($numSamples = 10000$ here), planning with more horizons ahead (larger $maxHorizon$) increases the reward the defender can get; (ii) if $numSamples$ is not large enough to ensure convergence for larger $maxHorizon$ ($numSamples = 100$ here), the reward the defender can get decreases as $maxHorizon$ increases because larger $maxHorizon$ indicates a deeper Monte Carlo tree so that more samples are needed to ensure convergence and it deteriorates the performance of MCTS if convergence is not reached.

Figure 5.3(a)/5.3(b) reports the results for FBR model for continuous utilities. Note that we are unable to solve problems with FQR model in the continuous utility scenario. Those figures show similar patterns as Figure 5.2(a)/5.2(b) and Figure 5.4(a)/5.4(b). However, an interesting observation is that continuous utilities requires more samples to ensure convergence. $numSamples = 10000$ is reasonably large enough for convergence in discrete utilities while it is not large enough for continuous utilities. The reason is that it involves a far larger utility space in continuous utilities so that it requires more samples to reach convergence.

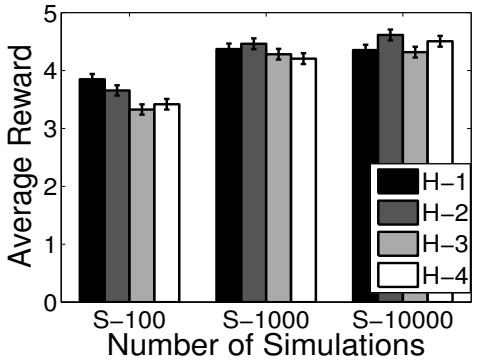
An interesting phenomenon observed in Figure 5.2(b)/5.4(b)/5.3(b) is that the performance difference between different $maxHorizon$ is tiny. However, this is not always the case. Here I provide an example where $maxHorizon$ makes a big difference in performance. Suppose that we have 3 targets: $u(1) = 5$; $u(2)$ is 10 with the probability 40% and 4 with the probability 60%; $u(3)$ has the same utility as $u(2)$. The penalties across all targets are all 0. We have 2 rounds in total. In round 1, if the defender chooses to protect target 1, the attacker gets $0.4 * 10 = 4$; if the defender chooses to protect target 2 or 3, the attacker gets $0.4 * 0.5 * 10 + 0.6 * 5 = 5$. Thus the defender will choose to protect target 1 if the $maxHorizon = 1$. In round 2, if the defender chooses to protect target 1 in round 1, the attacker attacks target 2 or 3 with equal probability, so he will get $0.5 * 0.6 * 4 + 0.5 * 0.4 * 10 = 3.2$, which gives the attacker the utility 7.2 in total; if the defender chooses to protect target 2 or 3 in round 1, the defender will learn exactly where the attacker is going to attack, and the attacker gets 0, which gives the attacker the utility 5 in total. Thus the defender will choose to protect target 2 or 3 in round 1 if $maxHorizon = 2$ and gets the expected utility -5 in these two rounds while getting the utility -7.2 if $maxHorizon = 1$.

5.6.1.3 POMCP (Particle Filter) vs GMOP (Gibbs Sampling)

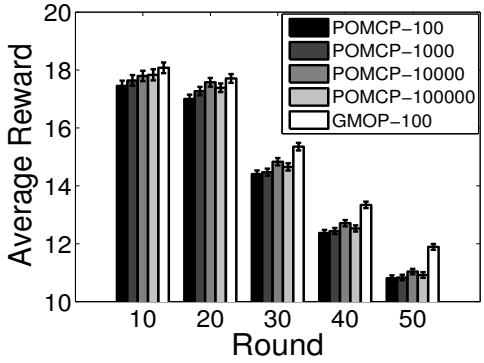
In this work I use Gibbs sampling to drive MCTS instead of the particle filter, as in the original POMCP algorithm [53]. In this way, the distribution of the samples is closer to the actual belief state. I now compare the performance of these two sampling techniques. The runtime of Gibbs sampling roughly increases linearly with $numSamples$; the runtime of the particle filter roughly increases linearly with the size of the particle filter (number



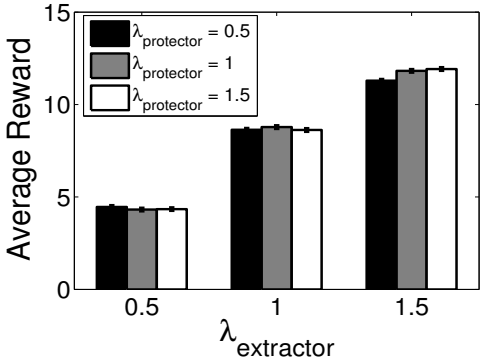
(a) MCTS convergence (FQR: $\lambda = 0.5$, 50 rounds)



(b) Different horizons (FQR: $\lambda = 0.5$, 50 rounds)

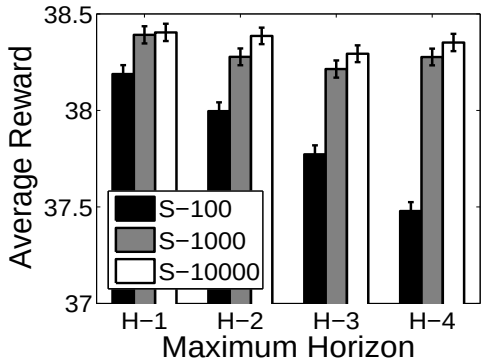


(c) GMOP vs POMCP in solution quality (FQR: $\lambda = 1.5$, $maxHorizon = 1$)

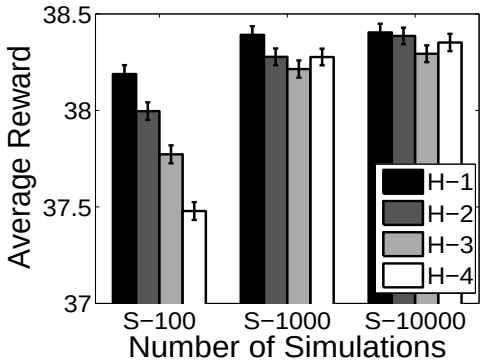


(d) Robustness (FQR, 50 rounds, $numSamples = 1000$, $maxHorizon = 2$)

Figure 5.2: Fictitious Quantal Response

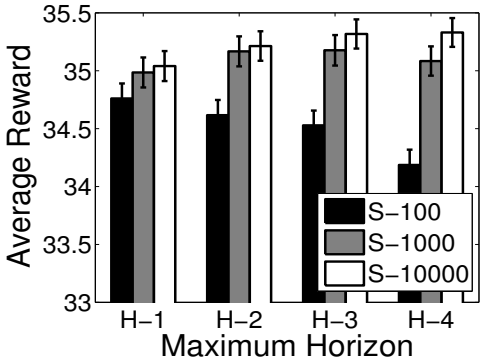


(a) MCTS convergence (FBR, 100 rounds)

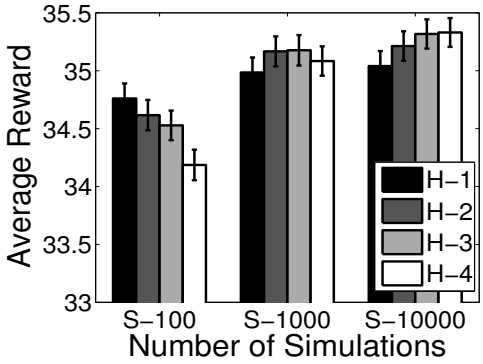


(b) Different horizons (FBR, 100 rounds)

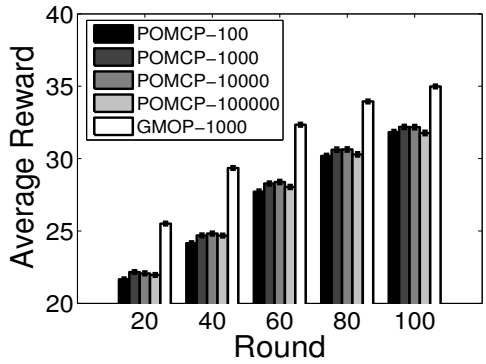
Figure 5.3: Continuous Utility Scenario



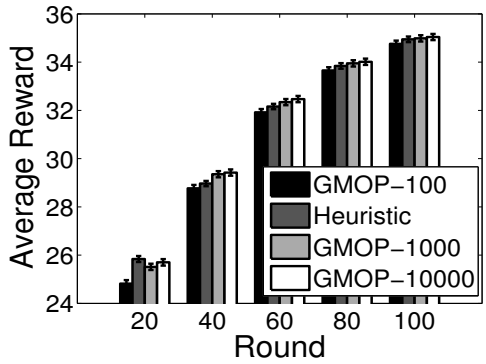
(a) MCTS convergence (FBR, 100 rounds)



(b) Different horizons (FBR, 100 rounds)



(c) GMOP vs POMCP in solution quality (FBR, $maxHorizon = 1$)



(d) GMOP vs heuristic in solution quality (FBR)

Figure 5.4: Fictitious Best Response

of particles). For a fair comparison, I fix the particle filter size as well as *numSamples* in Gibbs sampling.

For the FQR model, I set the particle filter size to be 100000 and *numSamples* in Gibbs sampling to be 100. The total runtimes are recorded in Table 5.2, where we see that the runtime of the GMOP algorithm is shorter than the runtime of POMCP as *numSamples* varies from 100 to 100000. However, Figure 5.2(c) demonstrates that the performance of the GMOP algorithm with 100 samples exceeds the performance of the POMCP algorithm regardless of the value of *numSamples*. This performance gap between GMOP and POMCP grows with time because the particle filter gives an increasingly worse approximation of the belief state as time evolves.

Table 5.2: GMOP vs POMCP in Runtime(s)

GMOP-100	POMCP-100	POMCP-1000	POMCP-10000	POMCP-100000
31.71	75.86	72.92	75.89	92.26

Fictitious Quantal Response ($\lambda = 1.5$), $maxHorizon = 1$

Table 5.3 and Figure 5.4(c) show the runtime and reward of GMOP with $numSamples = 1000$ vs POMCP with filter size 10000, for the FBR attacker. For the FBR attacker, we see the same pattern but with an even larger gap in solution quality. In the FBR attacker model, the particles are depleted much more quickly than in the FQR model so that more new particles must be added. However, these new particles do not follow the distribution induced by the current belief state, which is detrimental to the quality of the approximation of the belief state and thus leads to worse performance.

Table 5.3: GMOP vs POMCP in Runtime(s)

GMOP-1000	POMCP-100	POMCP-1000	POMCP-10000	POMCP-100000
83.48	224.35	240.83	257.40	282.71

Fictitious Best Response, $maxHorizon = 1$

5.6.1.4 Robustness

While the extension to unknown extractor can deal with the situation where the defender does not know the true value of λ that measures the attacker's rationality, I try to investigate here how the performance changes if the attacker's true value of λ is only slightly different from the defender's estimation. If the performance is very sensitive to the estimation of λ, we will have to include a lot attacker behavioral models with various λ in the extension where the attacker's behavioral model is unknown. However, if the performance is "robust" against the estimation of λ, it is not necessary to include a lot models in the extension. In this experiment, I allow the attacker's true value of λ to vary in a small scale—take values in $0.5, 1, 1.5$, and I allow the defender to estimate λ to be any of $0.5, 1, 1.5$, for a total of 9 combinations of the true λ and its estimate. Figure 5.2(d) presents the results of this experiment. It turns out that the defender only does slightly worse when she incorrectly estimates the attacker's true λ, which shows the "robustness" of the original framework.

5.6.1.5 Evaluation of the Advanced Sampling Technique in FBR Model

In Section 5.3.1, I proposed an advanced way to compute conditional probabilities when using Gibbs sampling in the FBR model. This technique is less computationally expensive than the general method. Table 5.4 compares the runtimes of the general sampling technique with the advanced sampling technique. As the number of rounds increases from 20 to 100, the total runtime of the advanced sampling technique increases linearly, implying that the sampling cost at each round is approximately the same. On the other hand, the total runtime of the general sampling technique increases with the square of

the number of rounds in the game, implying that the sampling cost is increasing linearly in each round.

Table 5.4: General vs Advanced Sampling in Runtime(s)

	20	40	60	80	100
General	51.77	209.31	469.80	835.15	1303.95
Advanced	43.83	62.04	77.24	92.77	108.67

Fictitious Best Response, $numSamples = 1000$, $maxHorizon = 1$

5.6.1.6 GMOP vs Myopic Planning Heuristic

The myopic heuristic trades solution quality for computational efficiency for a FBR attacker. Figure 5.4(d) compares the solution quality of the myopic planning heuristic versus GMOP, and Table 5.5 compares their total runtimes. For a fair comparison, I set $maxHorizon$ to be 1 in the GMOP algorithm. Figure 5.4(d) indicates that the heuristic gives better solutions than GMOP with $numSamples = 100$. However, the solution quality of the heuristic is worse than the one produced by GMOP when $numSamples$ equals 1000 or 10000. According to Table 5.5, the runtime of the myopic heuristic is much less than the runtime of GMOP.

Table 5.5: GMOP vs Heuristic in Runtime(s)

Heuristic	GMOP-100	GMOP-1000	GMOP-10000
0.49	8.38	83.48	689.87

Fictitious Best Response

5.6.2 Model Ensemble Evaluation

In this section, I will evaluate the performance of GMOP algorithm with the model ensemble idea. I use 6 different attacker models in the experiments. They are defined as:

- Model 1—Fictitious Quantal Response with $\lambda = 10$
- Model 2—Fictitious Quantal Response with $\lambda = 1$
- Model 3—Fictitious Quantal Response with the memory of the recent 20 rounds and $\lambda = 10$
- Model 4—Fictitious Quantal Response with the memory of the recent 20 rounds and $\lambda = 1$
- Model 5—Fictitious Quantal Response with exponentially reduced memory (exponential factor $= 0.9$)[5] and $\lambda = 10$
- Model 6—Fictitious Quantal Response with exponentially reduced memory (exponential factor $= 0.9$) and $\lambda = 1$

In this experiment, I try to compare the performance of the ensemble agent that takes into consideration all the 6 different models and the single agent that assumes the attacker has an exact behavioral model. Figure 5.5(a) shows the performance of those agents when the real attacker follows model 1. Similarly, Figure 5.5(b), Figure 5.5(c), Figure 5.5(d), Figure 5.5(e) and Figure 5.5(f) show the performance when the real attacker follows model 2, 3, 4, 5 and 6. The legend "Ensemble" represents the ensemble agent that takes into consideration all the 6 different models, while the legend "model i" represents the agent that assumes the attacker is of model i. Figures 5.5(a)/5.5(b)/5.5(c)/5.5(d)/5.5(e)/5.5(f) show how the performance of these 7 agents evolves as the game goes on. The observation is that as the game goes on, the performance of "Ensemble" comes very close to that of

[5]defender's action i rounds ago is of the weight 0.9^i such that more recent actions weigh more.

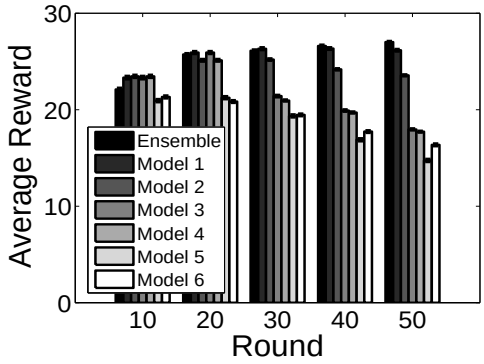
"Model i" when the real attacker follows model i, and outperforms the other single agents that assume the wrong attacker model. The ensemble agent gets a better idea as to the attacker's behavioral model as the game goes on, and thus the ensemble agent performs better than the single agent when the attacker model is unknown to the defender.

In Table 5.6, I compare the runtime of the ensemble agent and the single agent. We can see that using an ensemble agent only brings a little extra computational cost.

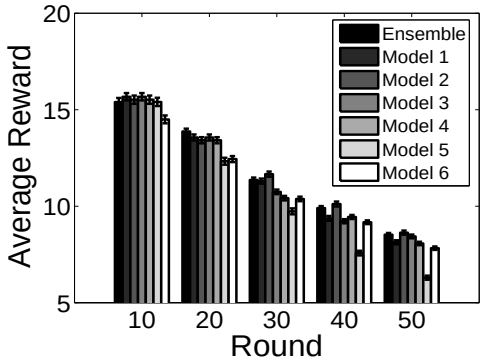
Table 5.6: Ensemble vs Single in Runtime(s)

Ensemble	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
3059.33	2912.09	2764.36	2925.66	2757.56	3802.97	2980.68

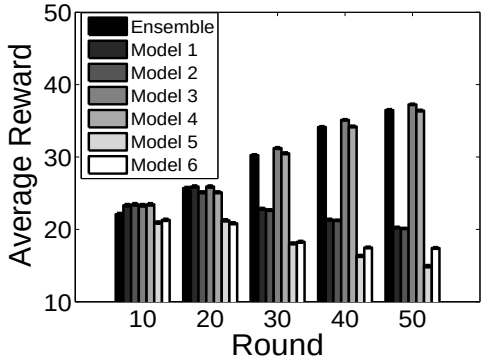
Deal with attacker of model 1, 50 rounds



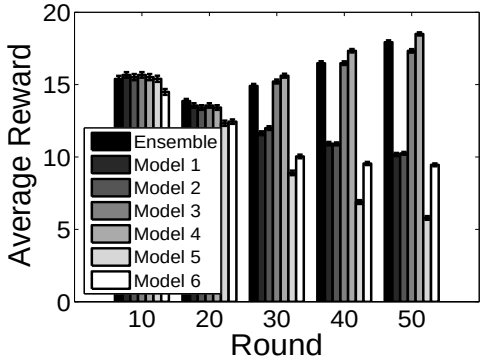
(a) Deal with attacker of model 1 (50 rounds)



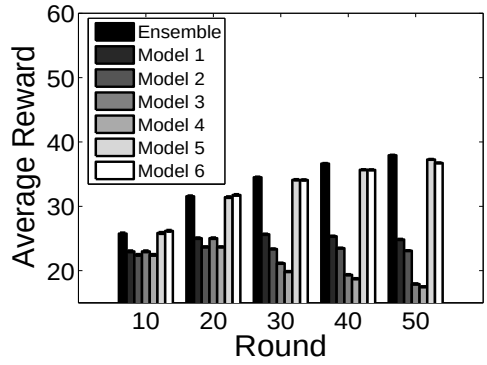
(b) Deal with attacker of model 2 (50 rounds)



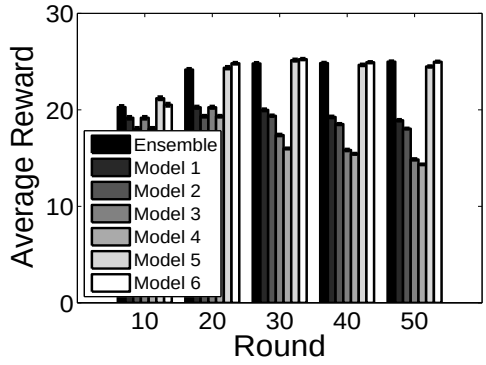
(c) Deal with attacker of model 3 (50 rounds)



(d) Deal with attacker of model 4 (50 rounds)



(e) Deal with attacker of model 5 (50 rounds)



(f) Deal with attacker of model 6 (50 rounds)

Figure 5.5: Model Ensemble

Chapter 6

Learning Attacker's Preference — Markovian Modeling

My work discussed in Chapter 5 assumes that defenders have knowledge of all poaching activities throughout the wildlife protected area. Unfortunately, given vast geographic areas for wildlife protection, defenders do not have knowledge of poaching activities in areas they do not protect. Thus, defenders are faced with the exploration-exploitation tradeoff — whether to protect the targets that are already known to have a lot of poaching activities or to explore the targets that haven't been protected for a long time. The work in this chapter aims to solve this exploration-exploitation tradeoff.

The exploration-exploitation tradeoff here is different from that in the non-Bayesian stochastic multi-armed bandit problem [4]. In stochastic multi-armed bandit problems, the rewards of every arm are random variables with a stationary unknown distribution. However, in my problem, patrol affects attack activities — more patrol is likely to decrease attack activities and less patrol is likely to increase attack activities. Thus, the random variable distribution is changing depending on player's choice — more selection (patrol) leads to lower reward (less attack activities) and less selection (patrol) leads to higher reward (more attack activities). On the other hand, adversarial multi-armed bandit

problem [5] is also not an appropriate model for this domain. In adversarial multi-armed bandit problems, the reward can arbitrarily change while the attack activities in my problem are unlikely to change rapidly in a short period. This makes the adversarial multi-armed bandit model inappropriate for this domain.

In reality, how patrol affects attack activities would be reasonably assumed to follow a consistent pattern that can be learned from historical data (defenders' historical observations). I model this pattern as a Markov process and provide the following contributions in this work. First, I formulate the problem into a restless multi-armed bandit (RMAB) model to handle the limited observability challenge — defenders do not have observations for arms they do not activate (targets they do not protect). Second, I propose an EM based learning algorithm to learn the RMAB model from defenders' historical observations. Third, i use the solution concept of Whittle index policy to solve the RMAB model to plan for defenders' patrol strategies. However, indexability is required for the existence of Whittle index, so I provide two sufficient conditions for indexability and an algorithm to numerically evaluate indexability. Fourth, I propose a binary search based algorithm to find the Whittle index policy efficiently.

6.1 Model

6.1.1 Motivating Domains and their Properties

My work is mainly motivated by the domain of wildlife protection such as protecting endangered animals and fish stocks [12, 58]. Other motivating domains include police patrol to catch fare-evaders in a barrier-free transit system [61], border patrol [21, 22],

etc. The model I will describe in this work is based on the following assumptions about the nature of interactions between defenders and attackers in these domains. Except the frequent interactions between defenders (patrollers/police) and attackers (poachers/fare-evaders/smugglers), these domains share another two important properties: (i) patrol affects attacking activities (poaching/fare evasion/smuggling); (ii) limited/partial observability. I will next use the wildlife protection domain as the example to illustrate these two properties.

Poaching activity is a dynamic process affected by patrol. If patrollers patrol in a certain location frequently, it is very likely that the poachers poaching in this location will switch to other locations for poaching. On the other hand, if a location hasn't been patrolled for a long time, poachers may gradually notice that and switch to this location for poaching.

In the wildlife protection domain, both patrollers and poachers do not have perfect observation of their opponents' actions. This observation imperfection lies in two aspects: (i) limited observability — patrollers/poachers do not know what happens at locations they do not patrol/poach; (ii) partial observability — patrollers/poachers do not have perfect observation even at locations they patrol/poach — the location might be large (e.g., a $2km \times 2km$ area) so that it is possible that patrollers and poachers do not see each other even if they are at the same location.

These two properties make it extremely difficult for defenders to optimally plan their patrol strategies. For example, defenders may find a target with a large number of attack activities at the beginning so they may start to protect this target frequently. After a period of time, attack activities at this target may start to decrease due to the frequent

patrol. At this time, defenders have to decide whether to keep protecting this target (exploitation) or to switch to other targets (exploration). However, defenders do not have knowledge of attack activities at other targets at that moment, which makes this decision making extremely difficult for defenders.

Fortunately, the frequent interactions between defenders and attackers make it possible for defenders to learn the effect of patrol on attackers from the historical data. With this learned effect, defenders are able to estimate attack activities at targets they do not protect. Based on this concept, I model these domains as a restless multi-armed bandit problem and use the solution concept of Whittle index policy to plan for defenders' strategies.

6.1.2 Formal Model

I now formalize the story in Section 6.1.1 into a mathematical model that can be formulated as a restless multi-armed bandit problem. There are n targets that are indexed by $\mathbb{N} \triangleq \{1, \ldots, n\}$. Defenders have k patrol resources that can be deployed to these n targets. At every round, defenders choose k targets to protect. After that, defenders will have an observation of the number of attack activities for targets they protect, and no information for targets they do not protect. The objective for defenders is to decide which k targets to protect at every round to catch as many attackers as possible.

Due to the partial observability on defenders' side — defenders' observation of attack activities is not perfect even for targets they protect, I introduce a hidden variable attack intensity, which represents the true degree of attack intensity at a certain target. Clearly, this hidden variable attack intensity cannot directly be observed by defenders. Instead,

defenders' observation is a random variable conditioned on this hidden variable attack intensity, and the larger the attack intensity is, the more likely it is for defenders to observe more attack activities during their patrol.

I discretize the hidden variable attack intensity into n_s levels, denoted by $\mathbf{S} = \{0, 1, \ldots, n_s - 1\}$. Lower i represents lower attack intensity. For a certain target, its attack intensity transitions after every round. If this target is protected, attack intensity transitions according to a $n_s \times n_s$ transition matrix T^1; if this target is not protected, attack intensity transitions according to another $n_s \times n_s$ transition matrix T^0. The transition matrix represents how patrol affects attack intensity — T^1 tends to reduce attack intensity and T^0 tends to increase attack intensity. The randomness in the transition matrix models attackers' partial observability discussed in Section 6.1.1. Note that different targets may have different transition matrices because some targets may be more attractive to attackers (for example, some locations may have more animal resources in the wildlife protection domain) so that it is more difficult for attack intensity to go down and easier for attack intensity to go up.

I also discretize defenders' observations of attack activities into n_o levels, denoted by $\mathbf{O} = \{0, 1, \ldots, n_o - 1\}$. Lower i represents less attack activities defenders observe. Note that defenders will only have observation for targets they protect. A $n_s \times n_o$ observation matrix O determines how the observation depends on the hidden variable attack intensity. Generally, the larger the attack intensity is, the more likely it is for defenders to observe more attack activities during their patrol. Similar to transition matrices, different target may have different observation matrices.

While defenders get observations of attack activities during their patrol, they also receive rewards for that — arresting poachers/fare-evaders/smugglers bring benefit. Clearly, the reward defenders receive depends on their observation and I thus define the reward function $R(o), o \in \mathbf{O}$ — larger i leads to higher reward $R(i)$. For example, if $o = 0$ represents finding no attack activity and $o = 1$ represents finding attack activities, then $R(0) = 0$, $R(1) = 1$. Note that defenders only get rewards for targets they protect.

To summarize, for the targets defenders protect, defenders get an observation depending on its current attack intensity, get the reward associated with the observation, and then the attack intensity transitions according to T^1; for the targets defenders do not protect, defenders do not have any observation, get reward 0 and the attack intensity transitions according to T^0. Figure 6.1 demonstrates this process. In this model, the state discretization level n_s, observation discretization level n_o and reward function $R(o)$ are pre-specified by defenders; the transition matrices T^1 and T^0, observation matrix O and initial belief π can be learned from defenders' previous observations. I will briefly discuss the learning algorithm in Section 6.1.3. After those parameters are learned, this model is formulated into a restless multi-armed bandit model to plan for defenders' strategies.

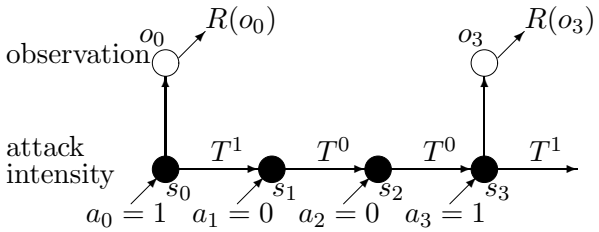


Figure 6.1: Model Illustration

6.1.3 Learning Model From Defenders' Previous Observations

Given defenders' action history $\{a_i\}$ and observation history $\{o_i\}$, my objective is to learn the transition matrices T^1 and T^0, observation matrix O and initial belief π. Due to the existence of hidden variables $\{s_i\}$, expectation-maximization (EM) algorithm is used for learning.

Expectation-maximization algorithm repeats the following steps until convergence:

1. compute $Q(\theta, \theta^d) = \sum_{z \in \mathcal{Z}} P(z|x; \theta^d) \log[P(x, z; \theta)]$
2. set $\theta^{(d+1)} = \arg\max_\theta Q(\theta, \theta^d)$

where z are latent variables, and are the hidden state sequence in my problem; x are observed data, and are the observation sequence in my problem; θ are the parameters to be estimated, and are the transition matrix T^1/T^0 (transition matrix when action $a = 1/0$), output matrix O and the initial hidden state distribution π.

$P(z|x; \theta^d) = \frac{P(x,z;\theta^d)}{P(x;\theta^d)}$, so we can write

$$\theta^{(d+1)} = \arg\max_\theta \sum_{z \in \mathcal{Z}} P(x, z; \theta^d) \log[P(x, z; \theta)]$$

Denote $\widehat{Q}(\theta, \theta^d) \triangleq \sum_{z \in \mathcal{Z}} P(x, z; \theta^d) \log[P(x, z; \theta)]$, so $\theta^{(d+1)} = \arg\max_\theta \widehat{Q}(\theta, \theta^d)$.

$$P(x, z; \theta) = \pi_{z_1} \prod_{t=1:a_t=1}^{T-1} T^1_{z_t z_{t+1}} \prod_{t=1:a_t=0}^{T-1} T^0_{z_t z_{t+1}} \prod_{t=1:a_t=1}^{T} O_{z_t x_t}$$

Taking the log gives us

$$\log P(x, z; \theta) = \log \pi_{z_1} + \sum_{t=1:a_t=1}^{T-1} \log T^1_{z_t z_{t+1}} + \sum_{t=1:a_t=0}^{T-1} \log T^0_{z_t z_{t+1}} + \sum_{t=1:a_t=1}^{T} \log O_{z_t x_t}$$

Then we have

$$\begin{aligned}\widehat{Q}(\theta,\theta^d) = &\sum_{z\in\mathcal{Z}} P(x,z;\theta^d)\log\pi_{z_1} \\ &+\sum_{z\in\mathcal{Z}}\sum_{t=1:a_t=1}^{T-1} P(x,z;\theta^d)\log T^1_{z_t z_{t+1}} \\ &+\sum_{z\in\mathcal{Z}}\sum_{t=1:a_t=0}^{T-1} P(x,z;\theta^d)\log T^0_{z_t z_{t+1}} \\ &+\sum_{z\in\mathcal{Z}}\sum_{t=1:a_t=1}^{T} P(x,z;\theta^d)\log O_{z_t x_t}\end{aligned}$$

We also have the constraints:

$$\begin{aligned}&\sum_{i=0}^{n_s-1}\pi_i = 1 \\ &\sum_{j=0}^{n_s-1} T^1_{ij} = 1, \forall i = 0,1,\ldots,n_s-1 \\ &\sum_{j=0}^{n_s-1} T^0_{ij} = 1, \forall i = 0,1,\ldots,n_s-1 \\ &\sum_{j=0}^{n_o-1} O^1_{ij} = 1, \forall i = 0,1,\ldots,n_s-1\end{aligned}$$

Using Lagrange multipliers we have:

$$\begin{aligned}\widehat{L}(\theta,\theta^d) = \widehat{Q}(\theta,\theta^d) &- \lambda_\pi(\sum_{i=0}^{n_s-1}\pi_i - 1) - \sum_{i=0}^{n_s-1}\lambda_{T_i^1}(\sum_{j=0}^{n_s-1} T^1_{ij} - 1) \\ &- \sum_{i=0}^{n_s-1}\lambda_{T_i^0}(\sum_{j=0}^{n_s-1} T^0_{ij} - 1) - \sum_{i=0}^{n_s-1}\lambda_{O_i}(\sum_{j=0}^{n_o-1} O_{ij} - 1)\end{aligned}$$

Take derivatives and set it to be 0, we get the update steps:

$$\begin{aligned}
\pi_i^{(d+1)} &= P(s_1 = i|x;\theta^d) \\
T_{ij}^{1(d+1)} &= \frac{\sum_{t=1:a_t=1}^{T-1} P(s_t = i, s_{t+1} = j|x;\theta^d)}{\sum_{t=1:a_t=1}^{T-1} P(s_t = i|x;\theta^d)} \\
T_{ij}^{0(d+1)} &= \frac{\sum_{t=1:a_t=0}^{T-1} P(s_t = i, s_{t+1} = j|x;\theta^d)}{\sum_{t=1:a_t=0}^{T-1} P(s_t = i|x;\theta^d)} \\
O_{ij}^{(d+1)} &= \frac{\sum_{t=1:a_t=1}^{T} P(s_t = i|x;\theta^d) I(o_t = j)}{\sum_{t=1:a_t=1}^{T} P(s_t = i|x;\theta^d)}
\end{aligned}$$

So we need to compute $P(s_t = i|x;\theta^d)$ and $P(s_t = i, s_{t+1} = j|x;\theta^d)$. It cam be computed through forward-backward algorithm.

Let $\alpha_i(t) = P(o_1 = x_1, \ldots, o_t = x_t, s_t = i;\theta)$. It can be computed recursively:

$$\alpha_i(1) = \begin{cases} \pi_i O_{ix_1}, & a_1 = 1, \\ \pi_i, & a_1 = 0. \end{cases}$$

$$\alpha_j(t+1) = \begin{cases} O_{jx_{t+1}} \sum_{i=0}^{n_s-1} \alpha_i(t) T_{ij}^1, & a_t = 1, a_{t+1} = 1, \\ O_{jx_{t+1}} \sum_{i=0}^{n_s-1} \alpha_i(t) T_{ij}^0, & a_t = 0, a_{t+1} = 1, \\ \sum_{i=0}^{n_s-1} \alpha_i(t) T_{ij}^1, & a_t = 1, a_{t+1} = 0, \\ \sum_{i=0}^{n_s-1} \alpha_i(t) T_{ij}^0, & a_t = 0, a_{t+1} = 0. \end{cases}$$

Define $\beta_i(t) = P(o_{t+1} = x_{t+1}, \ldots, o_T = x_T | s_t = i; \theta)$. It can also be computed recursively:

$$\beta_i(T) = 1$$

$$\beta_i(t) = \begin{cases} \sum_{j=0}^{n_s-1} \beta_j(t+1) T_{ij}^1 O_{jx_{t+1}}, & a_t = 1, a_{t+1} = 1, \\ \sum_{j=0}^{n_s-1} \beta_j(t+1) T_{ij}^0 O_{jx_{t+1}}, & a_t = 0, a_{t+1} = 1, \\ \sum_{j=0}^{n_s-1} \beta_j(t+1) T_{ij}^1, & a_t = 1, a_{t+1} = 0, \\ \sum_{j=0}^{n_s-1} \beta_j(t+1) T_{ij}^0, & a_t = 0, a_{t+1} = 0. \end{cases}$$

so we have:

$$P(s_t = i | x; \theta) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=0}^{n_s-1} \alpha_j(t)\beta_j(t)}$$

$$P(s_t = i, s_{t+1} = j | x; \theta) = \begin{cases} \frac{\alpha_i(t) T_{ij}^1 \beta_j(t+1) O_{jx_{t+1}}}{\sum_{k=0}^{n_s-1} \alpha_k(t)\beta_k(t)}, & a_t = 1, a_{t+1} = 1, \\ \frac{\alpha_i(t) T_{ij}^0 \beta_j(t+1) O_{jx_{t+1}}}{\sum_{k=0}^{n_s-1} \alpha_k(t)\beta_k(t)}, & a_t = 0, a_{t+1} = 1, \\ \frac{\alpha_i(t) T_{ij}^1 \beta_j(t+1)}{\sum_{k=0}^{n_s-1} \alpha_k(t)\beta_k(t)}, & a_t = 1, a_{t+1} = 0, \\ \frac{\alpha_i(t) T_{ij}^0 \beta_j(t+1)}{\sum_{k=0}^{n_s-1} \alpha_k(t)\beta_k(t)}, & a_t = 0, a_{t+1} = 0. \end{cases}$$

6.2 Restless Bandit for Planning

In this section, I will formulate the model discussed in Section 6.1.2 as a restless multi-armed bandit problem and plan defenders' strategies using the solution concept of Whittle index policy.

6.2.1 Restless Bandit Formulation

It is straightforward to formulation the model discussed in Section 6.1.2 into a restless multi-armed bandit problem. Every target is viewed as an arm and defenders choose k arms to activate (k targets to protect) at every round. Consider a single arm (target), it is associated with n_s (hidden) states, n_o observations, $n_s \times n_s$ transition matrices T^1 and T^0, $n_s \times n_o$ observation matrix O and reward function $R(o), o \in \mathbf{O}$ as is described in Section 6.1.2. For the arm defenders activate, defenders get an observation, get reward associated with the observation, and the state transitions according to T^1. Note that defenders' observation is not the state. Instead, it is a random variable conditioned on the state, and reveals some information about the state. For the arms defenders do not activate, defenders do not have any observation, get reward 0 and the state transitions according to T^0.

Since defenders can not directly observe the state, defenders maintain a belief b of the states for each target, based on which defenders make decisions. The belief is updated according to the Bayesian rules. The following equation shows the belief update when defenders protect this target ($a = 1$) and get observation o or defenders do not protect this target ($a = 0$).

$$b'(s') = \begin{cases} \eta \sum_{s \in \mathbf{S}} b(s) O_{so} T^1_{ss'}, & a = 1 \\ \sum_{s \in \mathbf{S}} b(s) T^0_{ss'}, & a = 0, \end{cases} \tag{6.1}$$

where η is the normalization factor. When defenders do not protect this target ($a = 0$), defenders do not have any observation, so their belief is updated according to the

state transition rule; When defenders protect this target ($a = 1$), their belief is firstly updated according to their observation o ($b_{new}(s) = \eta b(s) O_{so}$ according to Bayes' rule), and then the new belief is then updated according to the state transition rule: $b'(s') = \sum_{s \in \mathbf{S}} b_{new}(s) T^1_{ss'} = \sum_{s \in \mathbf{S}} \eta b(s) O_{so} T^1_{ss'} = \eta \sum_{s \in \mathbf{S}} b(s) O_{so} T^1_{ss'}$

I now present the mathematical definition of Whittle index for this problem. Denote $V_m(b)$ to be the value function for belief state b with subsidy m; $V_m(b; a = 0)$ to be the value function for belief state b with subsidy m and defenders take passive action; $V_m(b; a = 1)$ to be the value function for belief state b with subsidy m and defenders take active action. The following equations show these value functions:

$$
\begin{aligned}
V_m(b; a = 0) &= m + \beta V_m(b_{a=0}) \\
V_m(b; a = 1) &= \sum_{s \in \mathbf{S}} b(s) \sum_{o \in \mathbf{O}} O_{so} R(o) \\
&\quad + \beta \sum_{o \in \mathbf{O}} \sum_{s \in \mathbf{S}} b(s) O_{so} V_m(b^o_{a=1}) \\
V_m(b) &= \max\{V_m(b; a = 0), V_m(b; a = 1)\}
\end{aligned}
$$

When defenders take passive action, they get the immediate reward m and the β-discounted future reward — value function at new belief $b_{a=0}$, which is updated from b according to the case $a = 0$ in Equation 6.1. When defenders take active action, they get the expected immediate reward $\sum_{s \in \mathbf{S}} b(s) \sum_{o \in \mathbf{O}} O_{so} R(o)$ and the β-discounted future reward. The future reward is composed of different observation cases — $\sum_{s \in \mathbf{S}} b(s) O_{so}$ is defenders' probability to have observation o at belief state b, and $V_m(b^o_{a=1})$ is the value function at new belief $b^o_{a=1}$ that is updated from b according to the case $a = 1$ with

observation o in Equation 6.1. The value function $V_m(b)$ is the maximum of $V_m(b; a = 0)$ and $V_m(b; a = 1)$. Whittle index $I(b)$ of belief state b is then defined to be :

$$I(b) \triangleq \inf_m \{m : V_m(b; a = 0) \geq V_m(b : a = 1)\}$$

The passive action set $\Phi(m)$, which is the set of belief states for which passive action is the optimal action given subsidy m is then defined to be:

$$\Phi(m) \triangleq \{b : V_m(b; a = 0) \geq V_m(b : a = 1)\}$$

6.2.2 Sufficient Conditions for Indexability

In this section, I provide two sufficient conditions for indexability when $n_o = 2$ and $n_s = 2$. Denote the transition matrices to be T^0 and T^1, observation matrix to be O. Clearly in this problem, $O_{11} > O_{01}$, $O_{00} > O_{10}$ (higher attack intensity leads to higher probability to see attack activities when patrolling); $T^1_{11} > T^1_{01}$, $T^1_{00} > T^1_{10}$; $T^0_{11} > T^0_{01}$, $T^0_{00} > T^0_{10}$ (positively correlated arms).

Define $\alpha \triangleq \max\{T^0_{11} - T^0_{01}, T^1_{11} - T^1_{01}\}$. Since it is a two-state problem with $\mathbf{S} = \{0, 1\}$, I use one variable x to represent the belief state: $x \triangleq b(s = 1)$, which is the probability of being in state 1.

Define $\Gamma_1(x) = xT^1_{11} + (1-x)T^1_{01}$, which is the belief for the next round if the belief for the current round is x and the active action is taken. Similarly, $\Gamma_0(x) = xT^0_{11} + (1-x)T^0_{01}$, which is the belief for the next round if the belief for the current round is x and the passive action is taken.

I present below two theorems demonstrating two sufficient conditions for indexability. The proof is in Appendix B.

Theorem 6.2.1. *When $\beta \leq 0.5$, the process is indexable, i.e., for any belief x, if $V_m(x; a = 0) \geq V_m(x; a = 1)$, then $V_{m'}(x; a = 0) \geq V_{m'}(x; a = 1)$, $\forall m' \geq m$*

Theorem 6.2.2. *When $\alpha\beta \leq 0.5$ and $\Gamma_1(1) \leq \Gamma_0(0)$, the process is indexable, i.e., for any belief x, if $V_m(x; a = 0) \geq V_m(x; a = 1)$, then $V_{m'}(x; a = 0) \geq V_{m'}(x; a = 1)$, $\forall m' \geq m$*

6.2.3 Numerical Evaluation of Indexability

For problems other than those that have been proved to be indexable in Section 6.2.2, we can numerically evaluate their indexability. I first provide the following proposition.

Proposition 6.2.3. *If $m < R(0) - \beta\frac{R(n_o-1)-R(0)}{1-\beta}$, $\Phi(m) = \emptyset$; if $m > R(n_o - 1)$, $\Phi(m)$ is the whole belief state space.*

Proof. If $m < R(0) - \beta\frac{R(n_o-1)-R(0)}{1-\beta}$, denote $V_m(b; a = 0) = m + \beta W_0$; $V_m(b; a = 1) = R(o)+\beta W_1$, where W_1 and W_0 represent the maximum future reward. Since $W_0 \leq \frac{R(n_o-1)}{1-\beta}$ (achieving reward $R(n_o - 1)$ at every round), $W_1 \geq \frac{R(0)}{1-\beta}$ (achieving reward $R(0)$ at every round), $R(o) \geq R(0)$, we have $V_m(b; a = 1) - V_m(b; a = 0) = R(o) - m + \beta(W_1 - W_0) \geq R(0) - m + \beta\frac{R(0)-R(n_o-1)}{1-\beta} > 0$. Thus, being active is always the optimal action for any state so that $\Phi(m) = \emptyset$.

If $m > R(n_o-1)$, then the strategy of always being passive dominates other strategies so $\Phi(m)$ is the whole belief state space. □

Thus, we only need to determine whether the set $\Phi(m)$ monotonically increases for $m \subseteq [R(0) - \beta \frac{R(n_o - 1) - R(0)}{1-\beta}, R(n_o - 1)]$. Numerically, we can discretize this limited m range and then evaluate if $\Phi(m)$ monotonically increases with the increase of discretized m. Given the subsidy m, $\Phi(m)$ can be determined by solving a special POMDP model whose conditional observation probability is dependent on start state and action. I will discuss the algorithm in detail in Section 6.3. This algorithm returns a set D which contains n_s-length vectors $d_1, d_2, \ldots, d_{|D|}$. Every vector d_i is associated with an optimal action e_i. Given the belief b, the optimal action is determined by $a^{opt} = e_i$, $i = \arg\max_j b^T d_j$. Thus, $\Phi(m) = \bigcup_{i:e_i=0} \{b : b^T d_i \geq b^T d_j, \forall j\}$.

Given $m_0 < m_1$, my aim is to check whether $\Phi(m_0) \subseteq \Phi(m_1)$. Use the superscript 0 or 1 for set D, vector d, action e to distinguish between the returned solutions with subsidy m_0 and m_1. The following mixed-integer linear program (MILP) can be used to determine whether $\Phi(m_0) \subseteq \Phi(m_1)$.

$$
\begin{aligned}
\min_{b,z^0,z^1,\xi^0,\xi^1} \quad & \sum_{i=1}^{|D^0|} z_i^0 e_i^0 - \sum_{i=1}^{|D^1|} z_i^1 e_i^1 \\
s.t. \quad & b_i \in [0,1], \forall i \in \mathbf{S}, \qquad \sum_{i \in \mathbf{S}} b_i = 1 \\
& z_i^0 \in \{0,1\}, \forall i \in \{1,2,\ldots,|D^0|\}, \qquad \sum_i z_i^0 = 1 \\
& b^T d_i^0 \leq \xi^0, \forall i \in \{1,2,\ldots,|D^0|\} \\
& \xi^0 \leq b^T d_i^0 + M(1 - z_i^0), \forall i \in \{1,2,\ldots,|D^0|\} \\
& z_i^1 \in \{0,1\}, \forall i \in \{1,2,\ldots,|D^1|\} \qquad \sum_i z_i^1 = 1 \\
& b^T d_i^1 \leq \xi^1, \forall i \in \{1,2,\ldots,|D^1|\} \\
& \xi^1 \leq b^T d_i^1 + M(1 - z_i^1), \forall i \in \{1,2,\ldots,|D^1|\}
\end{aligned}
$$

If the result of the above MILP is 0 or 1, $\Phi(m_0) \subseteq \Phi(m_1)$. In the MILP, M is a given large number, b is the belief state, $z_i^{0/1}$ is a binary variable that indicates whether $b^T d_i^{0/1} \geq b^T d_j^{0/1}, \forall j$ (1 indicates yes and 0 indicates no), $\xi^{0/1}$ is an auxiliary variable that equals $\max_i b^T d_i^{0/1}$, $\sum_{i=1}^{|D^{0/1}|} z_i^{0/1} e_i^{0/1}$ is the optimal action for the problem with subsidy $m_{0/1}$. If the result of this MILP is 0 or 1, it means that there does not exist a belief b under which the optimal action for the problem with subsidy m_0 is passive (0) and the optimal action for the problem with subsidy m_1 is active (1). This means $\Phi(m_0) \subseteq \Phi(m_1)$.

6.2.4 Computation of Whittle Index Policy

Given the indexability, Whittle index can be found by doing a binary search within the range $m \subseteq [R(0) - \beta\frac{R(n_o-1)-R(0)}{1-\beta}, R(n_o-1)]$. Given the upper bound ub and lower bound lb, the problem with middle point $\frac{lb+ub}{2}$ as passive subsidy is sent to the special POMDP solver to find the optimal action for the current belief. If the optimal action is active, then the Whittle index is greater than the middle point so $lb \leftarrow \frac{lb+ub}{2}$; or else $ub \leftarrow \frac{lb+ub}{2}$. This binary search algorithm can find Whittle index with arbitrary precision. Naively, we can compute the Whittle index policy by computing the ε-precision indices of all arms and then picking the k arms with the highest indices.

However, since we are actually only interested in which k arms have the highest Whittle index and we do not care what exactly their indices are, we can do better than this naive method, which is demonstrated in Algorithm 8.

In Algorithm 8, A is Whittle index policy to be returned and is set to be $\emptyset$ at the beginning. S is the set of arms that we have not known whether belong to A or not and is set to be the whole set of arms at the beginning. Before it finds top-k arms (the loop between Line 4 and Line 21), it tests all the arms in S about their optimal action with subsidy $\frac{ub+lb}{2}$. If the optimal action is 1, it means this arm's index is higher than $\frac{ub+lb}{2}$ and we add it to S_1; if the optimal action is 0, it means this arm's index is lower than $\frac{ub+lb}{2}$ and we add it to S_0 (Lines $6-13$). At this moment, we know that all arms in S_1 have higher indices than all arms in S_0. If there is enough space in A to include all arms in S_1, we add S_1 to A, remove them from S and set the upper bound to be $\frac{ub+lb}{2}$ because we already know that S_1 belongs to Whittle index policy set and all the rest arms have

Algorithm 8 Algorithm to Compute Whittle Index Policy

function FINDWHITTLEINDEXPOLICY
  $lb \leftarrow R(0) - \beta\frac{R(n_o-1)-R(0)}{1-\beta}, ub \leftarrow R(n_o-1)$
  $A \leftarrow \emptyset$, $S \leftarrow \{1, 2, \ldots, n\}$
  while $|A| < k$ **do**
    $S_1 \leftarrow \emptyset$, $S_0 \leftarrow \emptyset$
    for $i \in S$ **do**
      $a^{opt} \leftarrow$ POMDPSOLVE(P_i, $\frac{lb+ub}{2}$)
      if $a^{opt} = 1$ **then**
        $S_1 \leftarrow S_1 \bigcup \{i\}$
      else
        $S_0 \leftarrow S_0 \bigcup \{i\}$
      end if
    end for
    if $|S_1| \leq k - |A|$ **then**
      $A \leftarrow A \bigcup S_1$, $S \leftarrow S - S_1$
      $ub \leftarrow \frac{lb+ub}{2}$
    else
      $S \leftarrow S - S_0$
      $lb \leftarrow \frac{lb+ub}{2}$
    end if
  end while
  return A
end function

the index lower than $\frac{ub+lb}{2}$ (Lines $14-16$). If there is not enough space in A, we remove S_0 from S and set the lower bound to be $\frac{ub+lb}{2}$ because we already know that S_0 does not belong to Whittle index policy set and all the rest arms have the index higher than $\frac{ub+lb}{2}$ (Lines $17-19$).

6.3 Computation of Passive Action Set

In this section, I will discuss the algorithm to compute the passive action set $\Phi(m)$ with the subsidy m. This problem can be viewed as solving a special POMDP model whose conditional observation probability is dependent on start state and action while the conditional observation probability is dependent on end state and action in standard POMDPs. Figure 6.2 demonstrates the difference. The left figure represents special POMDPs and the right figure represents standard POMDPs. In both cases, the original state is s, the agent takes action a, and the state transitions to s' according to $P(s'|s,a)$. However, the observation o the agent get during this process is dependent on s and a in my special POMDPs; while it depends on s' and a in standard POMDPs.

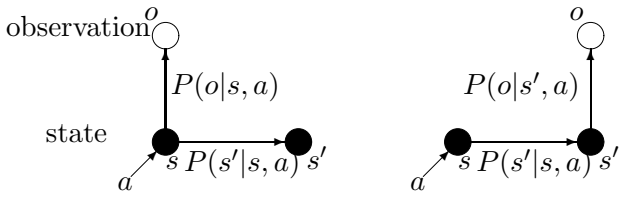


Figure 6.2: Special POMDPs vs Standard POMDPs

Despite this difference, the solution concept of value iteration algorithm in standard POMDPs can be used to solve my special POMDP formulations with appropriate modifications. I will discuss the special POMDP formulation for my problem in Section 6.3.1 and present the modified value iteration algorithm in Section 6.3.2.

6.3.1 Special POMDP Formulation

The special POMDP formulation for my problem is straightforward.

state space The state space is $\mathbf{S} = \{0, 1, \ldots, n_s - 1\}$.

action space The action space is $\mathbf{A} = \{0, 1\}$, where $a = 0$ represents passive action (do not protect) and $a = 1$ represents active action (protect).

observation space The observation space is $\mathbf{O} = \{-1, 0, 1, \ldots, n_o - 1\}$. It adds a "fake" observation $o = -1$ to represent no observation when taking action $a = 0$. It's called "fake" because defenders have probability 1 to observe $o = -1$ no matter what the state is when they take action $a = 0$, so this observation does not provide any information. When defenders take action $a = 1$, they may observe observations $\mathbf{O} \backslash \{-1\}$.

conditional transition probability The conditional transition probability $P(s'|s, a)$ is defined to be: $P(s' = j|s = i, a = 1) = T_{ij}^1$ and $P(s' = j|s = i, a = 0) = T_{ij}^0$.

conditional observation probability The conditional observation probability $P(o|s, a)$ is defined to be $P(o = -1|s, a = 0) = 1, \forall s \in \mathbf{S}$; $P(o = j|s = i, a = 1) = O_{ij}$. Note that the conditional observation probability here is dependent on the start state s and action a, while it depends on end state s' and action a in standard POMDP models. Intuitively, defenders' observation of attack activities today depends on the attack intensity today, not the transitioned attack intensity tomorrow.

reward function The reward function R is

$$R\left(s, s', a, o\right) = \begin{cases} 0, & a = 0, \\ R(o), & a = 1. \end{cases}$$

With the transition probability and observation probability, $R(s, a)$ can be computed. Note that this formulation is also slightly different due to the different definition of observation probability.

$$R(s, a) = \sum_{s' \in \mathbf{S}} P(s'|s, a) \sum_{o \in \mathbf{O}} P(o|s, a) R(s, s', a, o)$$

6.3.2 Value Iteration for My Special POMDP

Different from standard POMDP formulation, the belief update in the special POMDP formulation is

$$b'(s') = \frac{\sum_{s \in \mathbf{S}} b(s) P(o|s, a) P(s'|s, a)}{P(o|b, a)} \tag{6.2}$$

where

$$P(o|b, a) = \sum_{s' \in \mathbf{S}} \sum_{s \in \mathbf{S}} b(s) P(o|s, a) P(s'|s, a) = \sum_{s \in \mathbf{S}} b(s) P(o|s, a)$$

Note that the belief update process is also consistent with that in Equation 6.1. Similar to standard POMDP formulation, we have the value function

$$V'(b) = \max_{a \in \mathbf{A}} \left(\sum_{s \in \mathbf{S}} b(s) R(s, a) + \beta \sum_{o \in \mathbf{O}} P(o|b, a) V(b_a^o) \right)$$

which can be broken up to simpler combinations of other value functions:

$$V'(b) = \max_{a \in \mathbf{A}} V_a(b)$$
$$V_a(b) = \sum_{o \in \mathbf{O}} V_a^o(b)$$
$$V_a^o(b) = \frac{\sum_{s \in \mathbf{S}} b(s) R(s,a)}{|\mathbf{O}|} + \beta P(o|b,a) V(b_a^o)$$

All the value functions can be represented as $V(b) = \max_{\alpha \in \mathbf{D}} b \cdot \alpha$ since the update process maintains this property, so we only need to update the set D when updating the value function. The set D is updated according to the following process:

$$D' = \text{purge}\left(\bigcup_{a \in \mathbf{A}} D_a\right)$$
$$D_a = \text{purge}\left(\bigoplus_{o \in \mathbf{O}} D_a^o\right)$$
$$D_a^o = \text{purge}\left(\{\tau(\alpha, a, o) | \alpha \in D\}\right)$$

where $\tau(\alpha, a, o)$ is the $|D|$-vector given by

$$\tau(\alpha, a, o)(s) = (1/|\mathbf{O}|) R(s,a) + \beta P(o|s,a) \sum_{s' \in \mathbf{S}} \alpha(s') P(s'|s,a)$$

and purge$(\cdot)$ takes a set of vectors and reduces it to its unique minimum form (remove redundant vectors that are dominated by other vectors in the set). $\bigoplus$ represents the cross sum of two sets of vectors: $A \bigoplus B = \{\alpha + \beta | \alpha \in A, \beta \in B\}$.

The update of D' and D_a is intuitive, so I briefly explain the update of D_a^{o1} here:

$$
\begin{aligned}
P(o|b,a)V(b_a^o) &= P(o|b,a)\max_{\alpha\in D}\sum_{s'\in\mathbf{S}}\alpha(s')P(s'|b,a,o) \\
&= P(o|b,a)\max_{\alpha\in D}\sum_{s'\in\mathbf{S}}\alpha(s')\frac{\sum_{s\in\mathbf{S}}b(s)P(o|s,a)P(s'|s,a)}{P(o|b,a)} \\
&= \max_{\alpha\in D}\sum_{s'\in\mathbf{S}}\alpha(s')\sum_{s\in\mathbf{S}}b(s)P(o|s,a)P(s'|s,a) \\
&= \max_{\alpha\in D}\sum_{s\in\mathbf{S}}b(s)\cdot\left(P(o|s,a)\sum_{s'\in\mathbf{S}}\alpha(s')P(s'|s,a)\right)
\end{aligned}
$$

Here, $P(s'|b,a,o)$ is the belief of state s' in the next round when the belief in the current round is b, the agent takes action a and get the observation o, which is the $b(s')$ in Equation 6.2.

6.4 Planning from POMDP View

I have discussed in Section 6.3.1 that every single target can be modeled as a special POMDP model. Given that, we can combine these POMDP models at all targets to form a special POMDP model that describe the whole problem, and solving this special POMDP model leads to defenders' *exact* optimal strategy. Use the superscript i to denote target i. Generally, the POMDP model for the whole problem is the cross product of the single-target POMDP models at all targets with the constraint that only k targets are protected at every round.

state space The state space is $\mathbf{S} = \mathbf{S^1} \times \mathbf{S^2} \times \ldots \times \mathbf{S^n}$. Denote $s = (s^1, s^2, \ldots, s^n)$

[1] Actually the only difference of value iteration algorithm for the special POMDP formulation compared with that for the standard POMDP formulation is the different update of D_a^o.

action space The action space is $\mathbf{A} = \{(a^1, a^2, \ldots, a^n) | a^j \in \{0, 1\}, \forall j \in \mathbb{N}, \sum_{j \in \mathbb{N}} a^j = k\}$, which represents that only k targets can be protected at a round. Denote $a = (a^1, a^2, \ldots, a^n)$

observation space The observation space is $\mathbf{O} = \mathbf{O^1} \times \mathbf{O^2} \times \ldots \times \mathbf{O^n}$. Denote $o = (o^1, o^2, \ldots, o^n)$

conditional transition probability The conditional transition probability is $P(s'|s, a) = \prod_{j \in \mathbb{N}} P^j(s'^j | s^j, a^j)$.

conditional observation probability The conditional observation probability is $P(o|s, a) = \prod_{j \in \mathbb{N}} P^j(o^j | s^j, a^j)$.

reward function The reward function is $R(s, s', a, o) = \sum_{j \in \mathbb{N}} R(s^j, s'^j, a^j, o^j)$

Naively, the modified value iteration algorithm discussed in Section 6.3.2 can be used to solve this special POMDP formulation. However, this POMDP formulation suffers from curse of dimensionality — the problem size increases exponentially with the number of targets. Thus, the computational cost of value iteration algorithm will soon become unaffordable as the problem size grows.

Silver and Veness [53] have proposed POMCP algorithm, which provides high quality solutions and is scalable to large POMDPs. The POMCP algorithm only requires a simulator of the problem so it also applies to my special POMDPs. At a high level, the POMCP algorithm is composed of two parts: (i) it uses a particle filter to maintain an approximation of the belief state; (ii) it draw state samples from the particle filter and then use MCTS to simulate what will happen next to find the best action. It uses a particle filter to approximate the belief state because it is even computationally impossible in many problems to update belief state due to the extreme large size of the state space.

However, in my problem, the all-target POMDP model is the cross product of the single-target POMDP models at all targets. The single-state POMDP model is small so that it is computationally inexpensive to maintain its belief state. Thus, we can easily sample the state s^i at target i from its belief state and then compose them together to get the state sample $s = (s^1, s^2, \ldots, s^n)$ for the all-target POMDP model.

The details of MCTS in POMDP are available in [53] so I omit it here. Although the POMCP algorithm shows better scalability than the exact POMDP algorithm, its scalability is also limited because the action space and observation space are also exponential with k in my problem. Consider the problem instance of $n = 10$, $k = 3$ and $n_o = 2$, the number of actions is $\binom{10}{3} = \frac{10*9*8}{1*2*3} = 120$ and the number of observations is $\binom{10}{3} * 2^3 = 960$. Since actions and observations are the branches in the MCTS, the tree size will soon become extremely large when planning more rounds ahead. This leads to two problems: (i) it will soon run out of memory when planning more rounds ahead; (ii) a huge number of state samples is needed to establish the convergence. Thus, the POMCP algorithm only applies to problem instances with small k. The experimental evaluation shows that the POMCP algorithm is unable to plan 3 horizons forward (runs out of memory) for the problem instance of $n = 10$, $k = 3$ and $n_o = 2$. It means that for large problem instances, the POMCP algorithm is reduced to myopic policy (only look one round ahead when planning).

6.5 Experimental Evaluation

In this section, I will firstly evaluate the Whittle Index Policy in Section 6.5.1 and then evaluate the RMAB model in Section 6.5.2. The performance is evaluated in terms of the cumulative reward received within the first 20 rounds with discounting factor $\beta = 0.9$. All results are averaged over 500 simulation runs.

6.5.1 Evaluation of Whittle Index Policy

I will compare the Whittle Index policy with four baseline algorithms:

Random: The defenders randomly choose k targets to protect at every round.

Myopic: The defenders choose k targets with the highest immediate reward to protect at every round.

Exact POMDP: The defenders uses the modified value iteration algorithm to solve the special POMDP problem discussed in Section 6.4 to plan for patrol strategies at every round. Note that it only works for small-scale problems and is the *exact* optimal patrol strategy defenders may take

POMCP: The defenders uses POMCP algorithm to solve the special POMDP problem discussed in Section 6.4 to plan for patrol strategies at every round.

The computation of Whittle Index policy and exact POMDP algorithm involve solving special POMDPs using the modified value iteration algorithm as is discussed in Section 6.3.2. I implement the modified value iteration algorithm by modifying the POMDP solver written by Anthony R. Cassandra[2]. The detailed algorithm I use for value iteration is the incremental pruning algorithm [10].

[2]http://pomdp.org/code/

There are two parameters in the POMCP algorithm: the number of state samples and the depth of the tree, i.e., the number of rounds we look ahead when planning. With the increase of the number of state samples, the performance of the POMCP algorithm improves; while the runtime also increases at the same time. Thus, for a fair comparison, during my experiment, I choose the number of state samples so that its runtime is similar to that of Whittle index policy. For the depth of the tree, I choose the one with the largest cumulative reward.

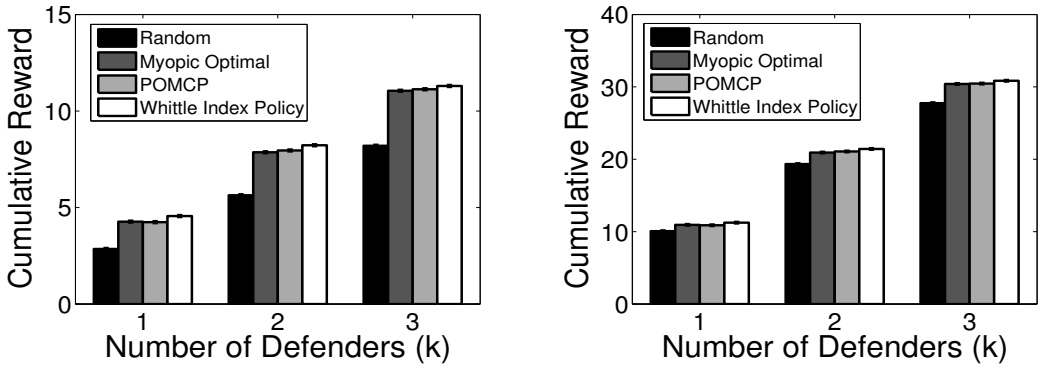
Small Scale: Compare with Exact POMDP Algorithm I then evaluate these five planning algorithms in a small problem instance with $n = 2$, $k = 1$, $n_s = 2$ and $n_o = 2$. The result is shown in Table 6.1. From the table, we can see that my Whittle index policy and POMCP algorithm perform very close to the optimal Exact POMDP solution and are much better than the myopic optimal policy and random policy, demonstrating their high solution quality.

Table 6.1: Planning Algorithm Evaluation in Solution Quality for Small-scale Problem Instances

Random	Myopic Optimal	POMCP	Exact POMDP	Whittle Index
2.6534	3.1384	3.1694	3.1798	3.1740

Large Scale: I then evaluate my planning algorithms in a larger problem instance with $n = 10$. Figure 6.3(a) shows the solution quality comparison when $n_s = 2$ and $n_o = 2$. The x-axis shows the number of defenders (k) and the y-axis shows the cumulative reward. From this figure, we can see that Whittle index policy performs better than the POMCP algorithm and myopic optimal policy, and all of these three algorithms perform much better than the random policy. One thing to note is that the POMCP algorithm shows poor scalability with regard to k — it is unable to plan 3 horizons forward (runs out of

memory) with $k = 3$. Figure 6.3(b) shows the solution quality comparison when $n_s = 3$ and $n_o = 3$, and demonstrates similar patterns as Figure 6.3(a).



(a) $n_s = 2, n_o = 2$ (b) $n_s = 3, n_o = 3$

Figure 6.3: Planning Algorithm Evaluation in Solution Quality for Large-scale Problem Instances

An Example When Myopic Policy Fails We can see from Figures 6.3(a) and 6.3(b) that the myopic policy performs only slightly worse compared with the Whittle index policy. Here I provide an example where the myopic policy performs significantly worse. Consider the case with 2 targets and 1 defender.

For target 0:

$$T^0 = \begin{bmatrix} 0.95 & 0.05 \\ 0.05 & 0.95 \end{bmatrix} T^1 = \begin{bmatrix} 0.99 & 0.01 \\ 0.1 & 0.9 \end{bmatrix} O = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}$$

For target 1:

$$T^0 = \begin{bmatrix} 0.4 & 0.6 \\ 0.1 & 0.9 \end{bmatrix} \quad T^1 = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \quad O = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix}$$

Figure 6.4 shows the performance of different algorithms. In this case, the myopic policy performs similar to the random policy, and is much worse compared with Whittle Index policy.

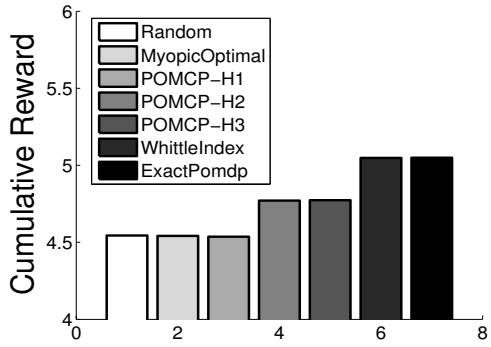


Figure 6.4: Example when Myopic Policy Fails

Runtime Analysis of Whittle Index Policy: Figure 6.5 analyzes the runtime of Whittle index policy. The x-axis shows the number of targets (n) and the y-axis shows the average runtime. From the figure, we can see that the runtime increases linearly with the number of targets. This is because Whittle index policy reduces an n-dimensional problem to n 1-dimensional problems so that the complexity is linear with n. Another observation is that the number of defenders (k) does not affect the runtime a lot for a given n.

6.5.2 Evaluation of RMAB Modeling

In this section, I will compare my RMAB model with the algorithms (UCB, SWUCB, EXP3) used in [21] with a group of simulated attackers. The performance is evaluated in terms of the cumulative reward received within the first 20 rounds after several rounds learning ($\beta = 0.9$).

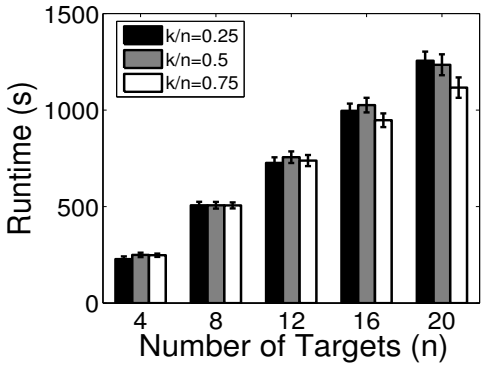
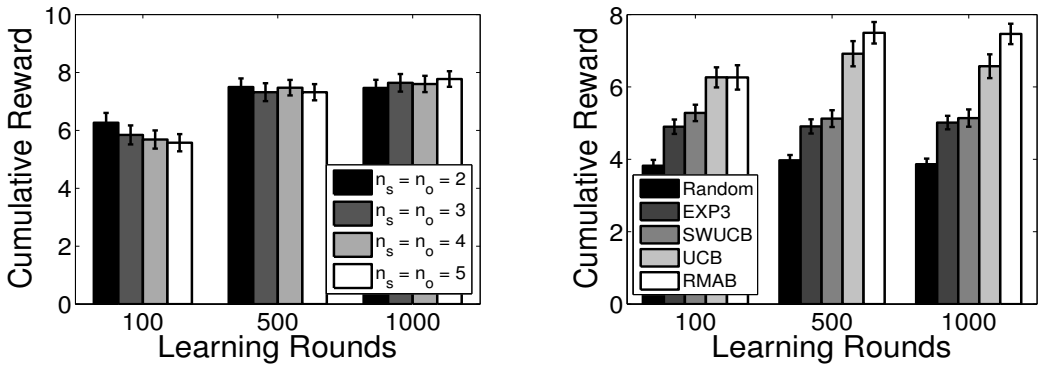


Figure 6.5: Runtime Analysis of Whittle Index Policy: $n_s = 2, n_o = 2$

Figure 6.6(a) demonstrates how the performance changes with different learning rounds and $n_s(n_o)$. It shows that when learning rounds is smaller (100), the model with $n_s = n_o = 2$ performs the best. This is because the model with higher $n_s(n_o)$ suffers overfitting with limited data at this time. When the data is relatively abundant (learning rounds = 1000), the model with higher $n_s(n_o)$ performs better. However, we noticed that the difference is not significantly large.

Figure 6.6(b) shows that comparison of my RMAB model with the Random/UCB/ SWUCB/EXP3 algorithms. When learning rounds is smaller (100), my RMAB model performs similar to UCB algorithm, and is better than other algorithms. When learning rounds becomes larger, my RMAB model shows significant advantage over other algorithms.



(a) RMAB

(b) $n_s = 2, n_o = 2$

Figure 6.6: Evaluation of RMAB Modeling

Chapter 7

Conclusion

7.1 Contributions

My contributions include addressing uncertainty in attackers' preference using robust and learning approaches. My first contribution develops an algorithm to efficiently compute the robust strategy against risk-aware attackers in SSGs. My second contribution models the preference as payoffs and focuses on learning the payoffs and then planning accordingly in green security domains. My third contribution models the preference as markovian process that transits according to defender's strategies to handle the exploration-exploitation tradeoff in these domains.

Robust Strategy against Risk-aware Attackers in SSGs My first contribution focuses on handling attacker's risk preference uncertainty in security games with robust approaches. I computes a robust defender strategy that optimizes the worst case against risk-aware attackers with uncertainty in the degree of risk awareness [1], i.e., it provides a solution quality guarantee for the defender no matter how risk-aware the attacker is.

To develop the robust strategy, I firstly build a robust SSG framework against an attacker with uncertainty in level of risk awareness. Second, building on previous work on SSGs in mixed-integer programs, I propose a novel mixed-integer bilinear programming problem (MIBLP), and find that it only finds locally optimal solutions. While the MIBLP formulation is also unable to scale up, it provides key intuition for my new algorithm. This new algorithm, BeRRA (**B**inary search based **R**obust algorithm against **R**isk-**A**ware attackers) is my third contribution, and it finds globally ϵ-optimal solutions by solving $\mathcal{O}(n \log(\frac{1}{\epsilon}) \log(\frac{1}{\delta}))$ linear feasibility problems. The key idea of the BeRRA algorithm is to reduce the problem from maximizing the reward with a given number of resources to minimizing the number of resources needed to achieve a given reward. This transformation allows BeRRA to scale up via the removal of the bilinear terms and integer variables as well as the utilization of key theoretical properties that prove correspondence of its potential "attack sets" [20] with that of the maximin strategy. Finally, I also show that the defender does not need to consider attacker's risk attitude in zero-sum games. The experimental results show the solution quality and runtime advantages of my robust model and BeRRA algorithm.

Learning Attacker's Preference — Payoff Modeling My second contribution focuses on learning attacker's payoffs in green security domains where there are frequent interactions between the defender and the attacker. I model attacker's preference as payoffs and the frequent interactions give the defender the opportunity to learn the attacker's payoffs by observing the attacker's actions. Motivated by this, my work develops the model and algorithm for the defender to learn target values from attacker's actions and then uses this information to better plan her strategy.

I model these interactions between the defender and the attacker as a repeated game. I then adopt a fixed model for the attacker's behavior and recast this repeated game as a partially observable Markov decision process (POMDP). However, my POMDP formulation has an exponential number of states, making current POMDP solvers like ZMDP [54] and APPL [25] infeasible in terms of computational cost. Silver and Veness [53] have proposed the POMCP algorithm which achieves a high level of performance in large POMDPs. It uses particle filtering to maintain an approximation of the belief state of the agent, and then uses Monte Carlo Tree Search (MCTS) for online planning. However, the particle filter is only an approximation of the belief state. By appealing to the special properties of my POMDP, I propose the GMOP algorithm (**G**ibbs sampling based **M**CTS **O**nline **P**lanning) which draws samples directly from the exact belief state using Gibbs sampling and then runs MCTS for online planning. My algorithm provides higher solution quality than the POMCP algorithm. Additionally, for a specific subclass of my game with an attacker who plays a best response against the defender's empirical distribution, and a uniform penalty of being seized across all targets, I provide an advanced sampling technique to speed up the GMOP algorithm along with a heuristic that trades off solution quality for lower computational cost. Moreover, I explore the case of continuous utilities where my original POMDP formulation becomes a continuous-state POMDP, which is generally difficult to solve. However, the special properties in the specific subclass of game mentioned above make possible the extension of the GMOP algorithm to continuous utilities. Finally, I explore the more realistic scenario where the defender is not only uncertain about the distribution of resources, but also uncertain about the attacker's

behavioral model. I address this challenge by extending my POMDP formulation and the GMOP algorithm.

There are two assumptions in this model: (i) both the defender and the attacker are able to observe their opponent's actions even if they are protecting/attacking different targets; (ii) the defender knows the attacker's behavioral model (or several candidate behavioral models). These two assumptions may not hold in some real-world domains. Thus in response, I have the third contribution that do not need these two assumptions.

Learning Attacker's Preference — Markovian Modeling My second contribution assumes that defenders have knowledge of all poaching activities throughout the wildlife protected area. Unfortunately, given vast geographic areas for wildlife protection, defenders do not have knowledge of poaching activities in areas they do not protect. My third contribution then relaxes this assumption by modeling attacker's preference as Markovian processes.

I assume that how patrol affects attack activities can be assumed to follow a consistent pattern that can be learned from historical data (defenders' historical observations). I model this pattern as a Markovian process and provide the following contributions in this work. First, I formulate the problem into a restless multi-armed bandit (RMAB) model to handle the limited observability challenge — defenders do not have observations for arms they do not activate (targets they do not protect). Second, I propose an EM based learning algorithm to learn the RMAB model from defenders' historical observations. Third, I use the solution concept of Whittle index policy to solve the RMAB model to plan for defenders' patrol strategies. However, indexability is required for the existence of Whittle index, so I provide two sufficient conditions for indexability and an algorithm

to numerically evaluate indexability. Fourth, I propose a binary search based algorithm to find the Whittle index policy efficiently.

In this contribution, it does not assume full observability or attacker's exact behavioral model. Instead, it assumes that the attacker's behavior follows certain pattern that does not change rapidly over time since all the planning algorithm is based on the adversary model learned from attacker's previous actions. In addition, since the planning algorithm fully trusts the learned adversary model, it assumes that the learned adversary model is correct. One possible extension is to include some robustness in the learned adversary model.

To summarize, since my thesis have two different contributions on learning, and they are also related to the green security game modeling [12, 58], I show in the following table the comparison between these models.

Table 7.1: Comparison Between Different Models

	AAMAS'14	AAMAS'16	Green Security Game
Strategy	pure strategy	pure strategy	mixed strategy
Observability by defender	full observability	limited observability	full observability
Observability by attacker	full observability	limited observability	full observability
assumption on behavioral models	QR	no	SUQR

These models have their advantages and disadvantages, and their combination may lead to possible directions for future work. For example, although the AAMAS'14 model is unable to handle the limited observability challenge and it assumes for attacker's behavioral model, it provides a more precise prediction result for attacker's preference. Similarly, although the mixed strategy setting in the green security game modeling has its limitations in the real world, its combination with my model might be possible to improve the performance.

7.2 Future Work

My thesis has discussed the algorithm to handle attacker's preference with robust and learning approaches. However, these two methods are all passively responding to attacker's preferences. Thus, an interesting question to ask is — is it possible for the defender to "manipulate" attacker's actions utilizing their preferences? An intuitive idea is that the defender may "manipulate" the attacker's actions by "manipulating" attacker's penalties in domains where the attacker's penalties are determined by the defender. For example, in the domain of protecting fish, some areas may be more "important" so that the defender may set the penalties higher in these areas. Therefore, it becomes a challenge for the defender as to how to set the penalties optimally to get a higher reward?

One other possible direction for future work is to combine the payoff uncertainty and attacker's risk attitude uncertainty together. Previous research [19] proposes the algorithm to compute the robust strategy against payoff uncertainty and my work [51] discusses the algorithm to compute the robust strategy against risk attitude uncertainty. Considering the fact that these two uncertainties may exist at the same time in real-world applications, a necessary next step is to develop an algorithm that can handle these two uncertainties at the same time.

Another possible direction for future work concerns with further improvements of the model for green security domains. My current model is based on a set of simplifying assumptions about the domain. For example, it discretizes the forest into a couple of grids and views each grid as a single target. In this way, it ignores the geometric relations between different grids — the poaching activities at neighboring grids may be correlated.

Moreover, it assumes that the defender protects a grid at every round. However, the defender may take a patrol route that goes across several targets at every round. A necessary next step is to take those domain features into account.

Bibliography

[1] Michele Aghassi and Dimitris Bertsimas. Robust game theory. *Mathematical Programming*, 2006.

[2] David J Agnew, John Pearce, Ganapathiraju Pramod, Tom Peatman, Reg Watson, John R Beddington, and Tony J Pitcher. Estimating the worldwide extent of illegal fishing. *PLoS One*, 2009.

[3] PS Ansell, Kevin D Glazebrook, José Niño-Mora, and M O'Keeffe. Whittle's index policy for a multi-class queueing system with convex holding costs. *Mathematical Methods of Operations Research*, 2003.

[4] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 2002.

[5] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*. IEEE, 1995.

[6] Jonathan F Bard. *Practical bilevel optimization: algorithms and applications*. Springer, 1998.

[7] Michael K Block and Vernon E Gerety. Some experimental evidence on differences between student and prisoner reactions to monetary penalties and risk. *The Journal of Legal Studies*, 1995.

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

[9] George Casella and Edward I George. Explaining the gibbs sampler. *The American Statistician*, 1992.

[10] Anthony Cassandra, Michael L Littman, and Nevin L Zhang. Incremental pruning: A simple, fast, exact method for partially observable markov decision processes. In *UAI*, 1997.

[11] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *EC*, 2006.

[12] Fei Fang, Peter Stone, and Milind Tambe. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *IJCAI*, 2015.

[13] KD Glazebrook, D Ruiz-Hernandez, and C Kirkbride. Some indexable families of restless bandit problems. *Advances in Applied Probability*, 2006.

[14] Jeffrey Grogger. Certainty vs. severity of punishment. *Economic Inquiry*, 1991.

[15] William B Haskell, Debarun Kar, Fei Fang, Milind Tambe, Sam Cheung, and Lt Elizabeth Denicola. Robust protection of fisheries with compass. 2014.

[16] Maria Hauck and NA Sweijd. A case study of abalone poaching in south africa and its impact on fisheries management. *ICES Journal of Marine Science: Journal du Conseil*, 1999.

[17] Bruce Hoffman. The modern terrorist mindset: Tactics, targets and technologies. *Center for the Study of Terrorism and Political Violence, St. Andrews University. As of April*, 1997.

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

[19] Christopher Kiekintveld, Towhidul Islam, and Vladik Kreinovich. Security games with interval uncertainty. In *AAMAS*, 2013.

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

[21] Richard Klíma, Christopher Kiekintveld, and Viliam Lisỳ. Online learning methods for border patrol resource allocation. In *Decision and Game Theory for Security*. Springer, 2014.

[22] Richard Klıma, Viliam Lisỳ, and Christopher Kiekintveld. Combining online learning and equilibrium computation in security games. 2015.

[23] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML*. 2006.

[24] Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Complexity of computing optimal stackelberg strategies in security resource allocation games. In *AAAI*, 2010.

[25] Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008.

[26] Joshua Letchford, Vincent Conitzer, and Kamesh Munagala. Learning and approximating the optimal strategy to commit to. In *Algorithmic Game Theory*. Springer, 2009.

[27] Keqin Liu and Qing Zhao. Indexability of restless bandit problems and optimality of whittle index for dynamic multichannel access. *Information Theory, IEEE Transactions on*, 2010.

[28] Keqin Liu, Qing Zhao, and Bhaskar Krishnamachari. Dynamic multichannel access with imperfect channel state detection. *Signal Processing, IEEE Transactions on*, 2010.

[29] Stéphanie Manel, Pierre Berthier, and Gordon Luikart. Detecting wildlife poaching: identifying the origin of individuals with bayesian assignment tests and multilocus genotypes. *Conservation Biology*, 2002.

[30] Janusz Marecki, Gerry Tesauro, and Richard Segal. Playing repeated stackelberg games with unknown opponents. In *AAMAS*, 2012.

[31] Richard D McKelvey and Thomas R Palfrey. Quantal response equilibria for normal form games. *Games and economic behavior*, 1995.

[32] Richard D McKelvey and Thomas R Palfrey. Quantal response equilibria for extensive form games. *Experimental economics*, 1998.

[33] Andrew R Morral and Brian A Jackson. *Understanding the role of deterrence in counterterrorism security*. Rand Corporation, 2014.

[34] Thanh Nguyen, Albert Jiang, and Milind Tambe. Stop the compartmentalization: Unified robust algorithms for handling uncertainties in security games. In *AAMAS*, 2014.

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

[36] Thanh Hong Nguyen, Amulya Yadav, Bo An, Milind Tambe, and Craig Boutilier. Regret-based optimization and preference elicitation for stackelberg security games with uncertainty. In *AAAI*, 2014.

[37] Jose Nino-Mora. Restless bandits, partial conservation laws and indexability. *Advances in Applied Probability*, 2001.

[38] Jerome Le Ny, Munther Dahleh, and Eric Feron. Multi-uav dynamic routing with partial observations using restless bandit allocation indices. In *American Control Conference*. IEEE, 2008.

[39] United States. Defense Science Board. Task Force on Preventing and Defending Against Clandestine Nuclear Attack. *Report of the Defense Science Board Task Force on Preventing and Defending Against Clandestine Nuclear Attack*. Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, 2004.

[40] National Research Council (US). Committee on Science and Technology for Countering Terrorism. Panel on Transportation. *Deterrence, protection, and preparation: the new transportation security imperative*. Number 270. Transportation Research Board, 2002.

[41] Christos H Papadimitriou and John N Tsitsiklis. The complexity of optimal queuing network control. *Mathematics of Operations Research*, 1999.

[42] Praveen Paruchuri, Jonathan P. Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. Playing games with security: An efficient exact algorithm for bayesian stackelberg games. In *AAMAS*, 2008.

[43] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[44] Peter Phillips. The preferred risk habitat of al-qa'ida terrorists. *European Journal of Economics, Finance and Administrative Sciences*, 2010.

[45] Peter Phillips. The preferred risk habitat of al-qa'ida terrorists. *European Journal of Economics, Finance and Administrative Sciences*, 2010.

[46] Peter J Phillips. Applying modern portfolio theory to the analysis of terrorism. computing the set of attack method combinations from which the rational terrorist group will choose in order to maximise injuries and fatalities. *Defence and Peace Economics*, 2009.

[47] Peter J Phillips. Applying modern portfolio theory to the analysis of terrorism. computing the set of attack method combinations from which the rational terrorist group will choose in order to maximise injuries and fatalities. *Defence and Peace Economics*, 2009.

[48] Peter J Phillips. The end of al-qa'ida: rationality, survivability and risk aversion. *International Journal of Economic Sciences*, 2013.

[49] Peter J Phillips. The end of al-qa'ida: rationality, survivability and risk aversion. *International Journal of Economic Sciences*, 2013.

[50] Yundi Qian, William B. Haskell, Albert Xin Jiang, and Milind Tambe. Online planning for optimal protector strategies in resource conservation games. In *AAMAS*, 2014.

[51] Yundi Qian, William B. Haskell, and Milind Tambe. Robust strategy against unknown risk-averse attackers in security games. In *AAMAS*, 2015.

[52] Yundi Qian, Chao Zhang, Bhaskar Krishnamachari, and Milind Tambe. Restless poachers: Handling exploration-exploitation tradeoffs in security domains. In *AAMAS*, 2016.

[53] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *NIPS*, 2010.

[54] Trey Smith. *Probabilistic Planning for Robotic Exploration*. PhD thesis, Carnegie Mellon University, 2007.

[55] Bernhard Von Stengel and Shmuel Zamir. Leadership with commitment to mixed strategies. 2004.

[56] Richard R Weber and Gideon Weiss. On an index policy for restless bandits. *Journal of Applied Probability*, 1990.

[57] Peter Whittle. Restless bandits: Activity allocation in a changing world. *Journal of applied probability*, 1988.

[58] Rong Yang, Benjamin Ford, Milind Tambe, and Andrew Lemieux. Adaptive resource allocation for wildlife protection against illegal poachers. In *AAMAS*, 2014.

[59] Rong Yang, Christopher Kiekintveld, Fernando Ordonez, Milind Tambe, and Richard John. Improving resource allocation strategy against human adversaries in security games. In *IJCAI*, 2011.

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

[61] Zhengyu Yin, Albert Jiang, Matthew Johnson, Milind Tambe, Christopher Kiekintveld, Kevin Leyton-Brown, Tuomas Sandholm, and John Sullivan. Trusts: Scheduling randomized patrols for fare inspection in transit systems. In *IAAI*, 2012.

[62] Zhengyu Yin and Milind Tambe. A unified method for handling discrete and continuous uncertainty in bayesian stackelberg games. In *AAMAS*, 2012.

Appendix A

An Example of BeRRA Algorithm

Consider the example below with 3 targets and 1 resource.

Table A.1: Example of BeRRA Algorithm

Target	U_d^u	U_d^c	U_a^u	U_a^c
t_0	-26	39	18	-14
t_1	-25	15	25	-27
t_2	-39	39	30	-24

In Algorithm 1, the upper bound for defender's reward is 39 and the lower bound is -39 at the beginning. Then it tries to find whether the reward 0 (the midpoint of upper bound and lower bound) is achievable with 1 resource. Algorithm 3 computes the maximin strategy $\mathbf{c^{max}}$ to achieve the reward 0 for the defender is $c_0^{max} = 0.4, c_1^{max} = 0.625, c_2^{max} = 0.5$ and the possible attack set under the maximin strategy only contains t_0, so $S_p(\mathbf{c^{max}}) = \{t_0\}$ and $S_i(\mathbf{c^{max}}) = \{t_1, t_2\}$. Then it uses binary search in Algorithm 4 to reduce the coverage probability for targets in $S_i(\mathbf{c^{max}})$ and get the optimal strategy $\mathbf{c^{opt}}$, which is $c_0^{opt} = 0.4, c_1^{opt} = 0.380769, c_2^{opt} = 0.459269$. It returns back to Algorithm 1 and Algorithm 1 computes that the minimum number of resources needed to achieve the reward 0 is 1.240038, which is larger than the resource we have, which is 1. Therefore, the reward 0 is not achievable for the defender with 1 resource and the upper bound of

defender's reward is then set to be 0. This process goes on and on and until the upper bound and the lower bound become closer enough.

Appendix B

Proof for Indexability

I am going to prove Theorem 6.2.1 and Theorem 6.2.2 demonstrating two sufficient conditions for indexability. Consider the case with $n_o = 2$ and $n_s = 2$.

Transition matrix:

$$T^0 = \begin{pmatrix} T^0_{00} & T^0_{01} \\ T^0_{10} & T^0_{11} \end{pmatrix}$$

$$T^1 = \begin{pmatrix} T^1_{00} & T^1_{01} \\ T^1_{10} & T^1_{11} \end{pmatrix}$$

Observation matrix:

$$O = \begin{pmatrix} O_{00} & O_{01} \\ O_{10} & O_{11} \end{pmatrix}$$

In this problem, $O_{11} > O_{01}$, $O_{00} > O_{10}$ (higher attack intensity leads to higher probability to see attack activities when patrolling); $T^1_{11} > T^1_{01}$, $T^1_{00} > T^1_{10}$; $T^0_{11} > T^0_{01}$, $T^0_{00} > T^0_{10}$ (positively correlated arms).

Define $\alpha \triangleq \max\{T^0_{11} - T^0_{01}, T^1_{11} - T^1_{01}\}$.

Since it is a two-state problem with $\mathbf{S} = \{0, 1\}$, I use one variable x to represent the belief state: $x \triangleq b(s = 1)$, which is the probability of being in state 1.

Define $\Gamma_1(x) = xT_{11}^1 + (1-x)T_{01}^1$, which is the belief for the next round if the belief for the current round is x and the active action is taken. Similarly, $\Gamma_0(x) = xT_{11}^0 + (1-x)T_{01}^0$, which is the belief for the next round if the belief for the current round is x and the passive action is taken. We have $\Gamma_1(x_2) - \Gamma_1(x_1) = (T_{11}^1 - T_{01}^1)(x_2 - x_1) \leq \alpha(x_2 - x_1), \forall x_2 > x_1$ and $\Gamma_0(x_2) - \Gamma_0(x_1) = (T_{11}^0 - T_{01}^0)(x_2 - x_1) \leq \alpha(x_2 - x_1), \forall x_2 > x_1$ according to the definition of α.

We normalize the reward function $R(o)$ so that $R(0) = 0$ and $R(1) = 1$.

B.1 Preliminaries

Further expand the value function presented in thesis, we have:

$$
\begin{aligned}
V_m(x; a = 0) &= m + \beta V_m(\Gamma_0(x)) \\
V_m(x; a = 1) &= xO_{11} + (1 - x)O_{01} \\
&\quad + \beta[xO_{11} + (1 - x)O_{01}]V_m(\Gamma_1(\frac{xO_{11}}{xO_{11} + (1 - x)O_{01}})) \\
&\quad + \beta[xO_{10} + (1 - x)O_{00}]V_m(\Gamma_1(\frac{xO_{10}}{xO_{10} + (1 - x)O_{00}})) \\
V_m(x) &= \max\{V_m(x; a = 0), V_m(x; a = 1)\}
\end{aligned}
$$

where $V_m(x)$ is the value function for belief state x with subsidy m, $V_m(x; a = 0)$ is the value function for belief state x with subsidy m and defenders take passive action,

and $V_m(x; a = 1)$ is the value function for belief state x with subsidy m and defenders take active action.

Define $V_m^t(x)$ to be the value function for state x with subsidy m when there are t rounds left. $V_m^t(x; a = 0)$ and $V_m^t(x; a = 1)$ are also defined similarly. Clearly,

$$\lim_{t \to +\infty} V_m^t(x) = V_m(x)$$

$$\lim_{t \to +\infty} V_m^t(x; a = 0) = V_m(x; a = 0)$$

$$\lim_{t \to +\infty} V_m^t(x; a = 1) = V_m(x; a = 1)$$

Proposition B.1.1. *The value function for the last round is:* $V_m^1(x) = \max\{m, xO_{11} + (1 - x)O_{01}\}$

Proof. It can be easily computed by assuming $V_m^0(x) = 0, \forall x \in [0, 1]$. □

Figure B.1 shows an example of $V_m^1(x)$ with $m = 0.2$, $O_{11} = 0.4$, $O_{01} = 0.1$

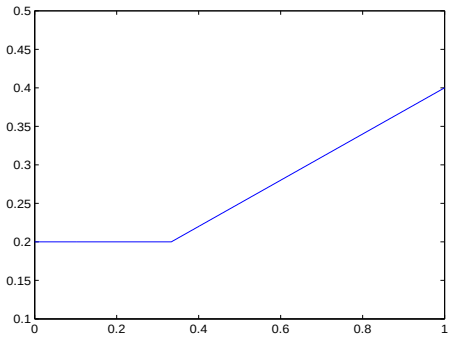


Figure B.1: An example of $V_m^1(x)$

My proof below is based on mathematical induction — I first prove the conclusion holds true for $V_m^1(x)$; then prove it holds true for $V_m^{t+1}(x)$ if it holds true for $V_m^t(x)$. Then it holds true for $V_m^\infty(x)$, in other words, the conclusion holds true for $V_m(x)$.

B.2 Proof of Theorem 6.2.1

To prove Theorem 6.2.1, I first prove a couple of lemmas.

Lemma B.2.1. *$V_{m'}(x) \geq V_m(x)$, $\forall m' \geq m, \forall x$*

Proof. Clearly, $V_{m'}^1(x) \geq V_m^1(x)$, $\forall m' \geq m, \forall x$

If $V_{m'}^t(x) \geq V_m^t(x)$, $\forall m' \geq m, \forall x$, since $V_{m'}^{t+1}(x; a = 0) \geq V_m^{t+1}(x; a = 0)$ and $V_{m'}^{t+1}(x; a = 1) \geq V_m^{t+1}(x; a = 1)$, we have $V_{m'}^{t+1}(x) \geq V_m^{t+1}(x)$, $\forall m' \geq m, \forall x$ □

Lemma B.2.2. *$V_{m'}(x) - V_m(x) \leq \frac{m'-m}{1-\beta}, \forall m' \geq m, \forall x$*

Proof. Clearly, $V_{m'}^1(x) - V_m^1(x) \leq m' - m \leq \frac{m'-m}{1-\beta}, \forall m' \geq m, \forall x$

If $V_{m'}^t(x) - V_m^t(x) \leq \frac{m'-m}{1-\beta}, \forall m' \geq m, \forall x$, we have:

$V_{m'}^{t+1}(x; a = 0) - V_m^{t+1}(x; a = 0) = m' + \beta V_{m'}^t(\Gamma_0(x)) - m - \beta V_m^t(\Gamma_0(x)) \leq (m' - m) + \beta \frac{m'-m}{1-\beta} = \frac{m'-m}{1-\beta}$

$V_{m'}^{t+1}(x; a = 1) - V_m^{t+1}(x; a = 1) \leq \beta \frac{m'-m}{1-\beta} \leq \frac{m'-m}{1-\beta}$

so $V_{m'}^{t+1}(x) - V_m^{t+1}(x) \leq \frac{m'-m}{1-\beta}, \forall m' \geq m, \forall x$ □

Now we are ready to prove Theorem 6.2.1.

Theorem (6.2.1)**.** *When $\beta \leq 0.5$, the process is indexable, i.e., for any belief x, if $V_m(x; a = 0) \geq V_m(x; a = 1)$, then $V_{m'}(x; a = 0) \geq V_{m'}(x; a = 1)$, $\forall m' \geq m$*

Proof. According to Lemma B.2.1: $V_{m'}(x;a=0) - V_m(x;a=0) = m' + \beta V_{m'}(\Gamma_0(x)) - m - \beta V_m(\Gamma_0(x)) \geq m' - m$

According to Lemma B.2.2: $V_{m'}(x;a=1) - V_m(x;a=1) \leq \beta\frac{m'-m}{1-\beta} \leq m' - m$

so $V_{m'}(x;a=0) - V_m(x;a=0) \geq V_{m'}(x;a=1) - V_m(x;a=1)$, therefore $V_{m'}(x;a=0) \geq V_{m'}(x;a=1)$ □

B.3 Proof of Theorem 6.2.2

To prove Theorem 6.2.2, we first prove the following lemma.

Lemma B.3.1. *When $\alpha\beta \leq 0.5$, we have:*

- *the value function $V_m(x)$ is an increasing function with x*
- *the optimal policy is a threshold policy with smaller x leads to $a=0$ and larger x leads to $a=1$*
- $V_m(x_2) - V_m(x_1) \leq \frac{(x_2-x_1)(O_{11}-O_{01})}{1-\alpha\beta}, \forall x_2 > x_1$

Proof. Obviously, it holds true for $t=1$.

Suppose it holds true for $V_m^t(x)$, i.e., $V_m^t(x)$ is an increasing function with x, the optimal policy is a threshold policy with smaller x leads to $a=0$ and larger x leads to $a=1$, and $V_m^t(x_2) - V_m^t(x_1) \leq \frac{(x_2-x_1)(O_{11}-O_{01})}{1-\alpha\beta}, \forall x_2 > x_1$, consider $V_m^{t+1}(x)$.

(i) We first prove that the optimal policy is a threshold policy for $V_m^{t+1}(x)$ with smaller x leads to $a=0$ and larger x leads to $a=1$. It is equivalent to prove that:

if $V_m^{t+1}(x_2;a=0) \geq V_m^{t+1}(x_2;a=1)$, then $V_m^{t+1}(x_1;a=0) \geq V_m^{t+1}(x_1;a=1), \forall x_1 < x_2$

if $V_m^{t+1}(x_1; a=1) \geq V_m^{t+1}(x_1; a=0)$, then $V_m^{t+1}(x_2; a=1) \geq V_m^{t+1}(x_2; a=0), \forall x_1 < x_2$

These conclusions are correct if $V_m^{t+1}(x_2; a=1) - V_m^{t+1}(x_1; a=1) \geq V_m^{t+1}(x_2; a=0) - V_m^{t+1}(x_1; a=0), \forall x_2 > x_1$.

$$
\begin{aligned}
& V_m^{t+1}(x_2; a=0) - V_m^{t+1}(x_1; a=0) \\
=& m + \beta V_m^t(\Gamma_0(x_2)) - m - \beta V_m^t(\Gamma_0(x_1)) \\
=& \beta(V_m^t(\Gamma_0(x_2)) - V_m^t(\Gamma_0(x_1))) \\
\leq& \beta \frac{(\Gamma_0(x_2) - \Gamma_0(x_1))(O_{11} - O_{01})}{1 - \alpha\beta} \\
\leq& \alpha\beta \frac{(x_2 - x_1)(O_{11} - O_{01})}{1 - \alpha\beta} \\
\leq& (x_2 - x_1)(O_{11} - O_{01})
\end{aligned}
$$

$$
\begin{aligned}
& V_m^{t+1}(x_2; a=1) - V_m^{t+1}(x_1; a=1) \\
=& (O_{11} - O_{01})(x_2 - x_1) \\
& + \beta[x_2 O_{11} + (1 - x_2) O_{01}] V_m^t(\Gamma_1(\frac{x_2 O_{11}}{x_2 O_{11} + (1 - x_2) O_{01}})) \\
& + \beta[x_2 O_{10} + (1 - x_2) O_{00}] V_m^t(\Gamma_1(\frac{x_2 O_{10}}{x_2 O_{10} + (1 - x_2) O_{00}})) \\
& - \beta[x_1 O_{11} + (1 - x_1) O_{01}] V_m^t(\Gamma_1(\frac{x_1 O_{11}}{x_1 O_{11} + (1 - x_1) O_{01}})) \\
& - \beta[x_1 O_{10} + (1 - x_1) O_{00}] V_m^t(\Gamma_1(\frac{x_1 O_{10}}{x_1 O_{10} + (1 - x_1) O_{00}})) \\
\geq& (O_{11} - O_{01})(x_2 - x_1)
\end{aligned}
$$

The last inequality is due to the increasing property of $V_m^t(x)$.

So we have $V_m^{t+1}(x_2; a = 1) - V_m^{t+1}(x_1; a = 1) \geq V_m^{t+1}(x_2; a = 0) - V_m^{t+1}(x_1; a = 0), \forall x_2 > x_1$, and thus the optimal policy is a threshold policy with smaller x leads to $a = 0$ and larger x leads to $a = 1$ for $V_m^{t+1}(x)$.

(ii) We next prove that $V_m^{t+1}(x)$ is an increasing function with the increase of x. This is true because $V_m^{t+1}(x; a = 1)$ and $V_m^{t+1}(x; a = 0)$ are all increasing functions (because $V_m^t(x)$ is an increasing function) so that their maximization is also an increasing function.

(iii) Last we prove that $V_m^{t+1}(x_2) - V_m^{t+1}(x_1) \leq \frac{(x_2 - x_1)(O_{11} - O_{01})}{1 - \alpha\beta}, \forall x_2 > x_1$. This can be proved by proving $V_m^{t+1}(x_2; a = 0) - V_m^{t+1}(x_1; a = 0) \leq \frac{(x_2 - x_1)(O_{11} - O_{01})}{1 - \alpha\beta}, \forall x_2 > x_1$ and

$V_m^{t+1}(x_2; a=1) - V_m^{t+1}(x_1; a=1) \leq \frac{(x_2-x_1)(O_{11}-O_{01})}{1-\alpha\beta}, \forall x_2 > x_1$. The former one when $a=0$ has been proved in (i). We will next prove the latter one when $a=1$.

$$
\begin{aligned}
& V_m^{t+1}(x_2; a=1) - V_m^{t+1}(x_1; a=1) \\
=& (O_{11}-O_{01})(x_2-x_1) \\
& + \beta[x_2 O_{11} + (1-x_2)O_{01}] V_m^t(\Gamma_1(\frac{x_2 O_{11}}{x_2 O_{11} + (1-x_2)O_{01}})) \\
& + \beta[x_2 O_{10} + (1-x_2)O_{00}] V_m^t(\Gamma_1(\frac{x_2 O_{10}}{x_2 O_{10} + (1-x_2)O_{00}})) \\
& - \beta[x_1 O_{11} + (1-x_1)O_{01}] V_m^t(\Gamma_1(\frac{x_1 O_{11}}{x_1 O_{11} + (1-x_1)O_{01}})) \\
& - \beta[x_1 O_{10} + (1-x_1)O_{00}] V_m^t(\Gamma_1(\frac{x_1 O_{10}}{x_1 O_{10} + (1-x_1)O_{00}})) \\
=& (O_{11}-O_{01})(x_2-x_1) \\
& + \beta[x_1 O_{11} + (1-x_1)O_{01}](V_m^t(\Gamma_1(\frac{x_2 O_{11}}{x_2 O_{11} + (1-x_2)O_{01}})) - V_m^t(\Gamma_1(\frac{x_1 O_{11}}{x_1 O_{11} + (1-x_1)O_{01}}))) \\
& + \beta[x_2 O_{10} + (1-x_2)O_{00}](V_m^t(\Gamma_1(\frac{x_2 O_{10}}{x_2 O_{10} + (1-x_2)O_{00}})) - V_m^t(\Gamma_1(\frac{x_1 O_{10}}{x_1 O_{10} + (1-x_1)O_{00}}))) \\
& + \beta(x_2-x_1)(O_{11}-O_{01})(V_m^t(\Gamma_1(\frac{x_2 O_{11}}{x_2 O_{11} + (1-x_2)O_{01}})) - V_m^t(\Gamma_1(\frac{x_1 O_{10}}{x_1 O_{10} + (1-x_1)O_{00}}))) \\
\leq& (O_{11}-O_{01})(x_2-x_1) + \beta\frac{O_{11}-O_{01}}{1-\alpha\beta}\alpha(\frac{x_2 O_{11}(x_1 O_{11} + (1-x_1)O_{01})}{x_2 O_{11} + (1-x_2)O_{01}} - x_1 O_{11}) \\
& + \beta\frac{O_{11}-O_{01}}{1-\alpha\beta}\alpha(x_2 O_{10} - \frac{x_1 O_{10}(x_2 O_{10} + (1-x_2)O_{00})}{x_1 O_{10} + (1-x_1)O_{00}}) \\
& + \beta(x_2-x_1)(O_{11}-O_{01})\frac{(O_{11}-O_{01})}{(1-\alpha\beta)}\alpha\frac{x_2 O_{11}(1-x_1)O_{00} - x_1 O_{10}(1-x_2)O_{01}}{(x_2 O_{11} + (1-x_2)O_{01})(x_1 O_{10} + (1-x_1)O_{00})} \\
=& (O_{11}-O_{01})(x_2-x_1)(1 + \frac{\alpha\beta}{1-\alpha\beta}) = \frac{(O_{11}-O_{01})(x_2-x_1)}{1-\alpha\beta}
\end{aligned}
$$

□

During the proof of Lemma B.3.1, we have the following result, which will be used in proving Lemma B.3.3.

Remark B.3.2. *When $\alpha\beta \leq 0.5$, we have:*

- $V_m^t(x_2; a=0) - V_m^t(x_1; a=0) \leq (x_2 - x_1)(O_{11} - O_{01}), \forall x_2 > x_1, \forall t$
- $(O_{11} - O_{01})(x_2 - x_1) \leq V_m^t(x_2; a=1) - V_m^t(x_1; a=1) \leq \frac{(O_{11}-O_{01})(x_2-x_1)}{1-\alpha\beta}, \forall x_2 > x_1, \forall t$

Lemma B.3.3. *When $\alpha\beta \leq 0.5$ and $\Gamma_1(1) \leq \Gamma_0(0)$, we have:*

- $V_{m'}(x_1) - V_m(x_1) \geq V_{m'}(x_2) - V_m(x_2), \forall x_1 \leq x_2$
- $V_{m'}(0) - V_m(0) - V_{m'}(1) + V_m(1) \leq m' - m$
- $V_{m'}(x; a=0) - V_m(x; a=0) \geq V_{m'}(x; a=1) - V_m(x; a=1), \forall x$

Proof. Consider the following sets of conclusions:

- $V_{m'}^t(x_1) - V_m^t(x_1) \geq V_{m'}^t(x_2) - V_m^t(x_2), \forall x_1 \leq x_2$
- $V_{m'}^t(0) - V_m^t(0) - V_{m'}^t(1) + V_m^t(1) \leq m' - m$
- $V_{m'}^{t+1}(x; a=0) - V_m^{t+1}(x; a=0) \geq V_{m'}^{t+1}(x; a=1) - V_m^{t+1}(x; a=1), \forall x$

Obviously, it holds true for $t = 0$. Suppose it holds true for t. Consider $t + 1$:

(i) We first prove $V_{m'}^{t+1}(x_1) - V_m^{t+1}(x_1) - V_{m'}^{t+1}(x_2) + V_m^{t+1}(x_2) \geq 0$

$$
\begin{aligned}
& V_{m'}^{t+1}(x_1) - V_m^{t+1}(x_1) - V_{m'}^{t+1}(x_2) + V_m^{t+1}(x_2) \\
=& \max\{V_{m'}^{t+1}(x_1; a=0), V_{m'}^{t+1}(x_1; a=1)\} \\
& - \max\{V_m^{t+1}(x_1; a=0), V_m^{t+1}(x_1; a=1)\} \\
& - \max\{V_{m'}^{t+1}(x_2; a=0), V_{m'}^{t+1}(x_2; a=1)\} \\
& + \max\{V_m^{t+1}(x_2; a=0), V_m^{t+1}(x_2; a=1)\}
\end{aligned}
$$

- Case 1: $V_m^{t+1}(x_1; a=0) \geq V_m^{t+1}(x_1; a=1)$ and $V_{m'}^{t+1}(x_2; a=0) \geq V_{m'}^{t+1}(x_2; a=1)$

$$
\begin{aligned}
& V_{m'}^{t+1}(x_1) - V_m^{t+1}(x_1) - V_{m'}^{t+1}(x_2) + V_m^{t+1}(x_2) \\
\geq& V_{m'}^{t+1}(x_1; a=0) - V_m^{t+1}(x_1; a=0) - V_{m'}^{t+1}(x_2; a=0) + V_m^{t+1}(x_2; a=0) \\
=& m' + \beta V_{m'}^t(\Gamma_0(x_1)) - m - \beta V_m^t(\Gamma_0(x_1)) - m' - \beta V_{m'}^t(\Gamma_0(x_2)) + m + \beta V_m^t(\Gamma_0(x_2)) \\
=& \beta(V_{m'}^t(\Gamma_0(x_1)) - V_m^t(\Gamma_0(x_1)) - V_{m'}^t(\Gamma_0(x_2)) + V_m^t(\Gamma_0(x_2))) \geq 0
\end{aligned}
$$

- Case 2: $V_m^{t+1}(x_1; a=0) \geq V_m^{t+1}(x_1; a=1)$ and $V_{m'}^{t+1}(x_2; a=0) < V_{m'}^{t+1}(x_2; a=1)$

$$
\begin{aligned}
& V_{m'}^{t+1}(x_1) - V_m^{t+1}(x_1) - V_{m'}^{t+1}(x_2) + V_m^{t+1}(x_2) \\
\geq& V_{m'}^{t+1}(x_1; a=0) - V_m^{t+1}(x_1; a=0) - V_{m'}^{t+1}(x_2; a=1) + V_m^{t+1}(x_2; a=1) \\
\geq& V_{m'}^{t+1}(x_1; a=0) - V_m^{t+1}(x_1; a=0) - V_{m'}^{t+1}(x_2; a=0) + V_m^{t+1}(x_2; a=0) \\
\geq& 0
\end{aligned}
$$

- Case 3: $V_m^{t+1}(x_1; a=0) < V_m^{t+1}(x_1; a=1)$ and $V_{m'}^{t+1}(x_2; a=0) \geq V_{m'}^{t+1}(x_2; a=1)$

$$
\begin{aligned}
& V_{m'}^{t+1}(x_1) - V_m^{t+1}(x_1) - V_{m'}^{t+1}(x_2) + V_m^{t+1}(x_2) \\
\geq & V_{m'}^{t+1}(x_1; a=0) - V_m^{t+1}(x_1; a=1) - V_{m'}^{t+1}(x_2; a=0) + V_m^{t+1}(x_2; a=1) \\
\geq & -(x_2 - x_1)(O_{11} - O_{01}) + (O_{11} - O_{01})(x_2 - x_1) = 0
\end{aligned}
$$

The last inequality is based on Remark 1.

- Case 4: $V_m^{t+1}(x_1; a=0) < V_m^{t+1}(x_1; a=1)$ and $V_{m'}^{t+1}(x_2; a=0) < V_{m'}^{t+1}(x_2; a=1)$

$$
\begin{aligned}
& V_{m'}^{t+1}(x_1) - V_m^{t+1}(x_1) - V_{m'}^{t+1}(x_2) + V_m^{t+1}(x_2) \\
\geq & V_{m'}^{t+1}(x_1; a=1) - V_m^{t+1}(x_1; a=1) - V_{m'}^{t+1}(x_2; a=1) + V_m^{t+1}(x_2; a=1) \\
= & \beta[x_2 O_{11} + (1-x_2)O_{01}](V_{m'}^t(\Gamma_1(\frac{x_1 O_{11}}{x_1 O_{11} + (1-x_1)O_{01}})) - V_m^t(\Gamma_1(\frac{x_1 O_{11}}{x_1 O_{11} + (1-x_1)O_{01}})) \\
& - V_{m'}^t(\Gamma_1(\frac{x_2 O_{11}}{x_2 O_{11} + (1-x_2)O_{01}})) + V_m^t(\Gamma_1(\frac{x_2 O_{11}}{x_2 O_{11} + (1-x_2)O_{01}}))) \\
& + \beta[x_2 O_{10} + (1-x_2)O_{00}](V_{m'}^t(\Gamma_1(\frac{x_1 O_{10}}{x_1 O_{10} + (1-x_1)O_{00}})) - V_m^t(\Gamma_1(\frac{x_1 O_{10}}{x_1 O_{10} + (1-x_1)O_{00}})) \\
& - V_{m'}^t(\Gamma_1(\frac{x_2 O_{10}}{x_2 O_{10} + (1-x_2)O_{00}})) + V_m^t(\Gamma_1(\frac{x_2 O_{10}}{x_2 O_{10} + (1-x_2)O_{00}}))) \\
& + \beta(x_2 - x_1)(O_{11} - O_{01})(V_{m'}^t(\Gamma_1(\frac{x_1 O_{10}}{x_1 O_{10} + (1-x_1)O_{00}})) - V_m^t(\Gamma_1(\frac{x_1 O_{10}}{x_1 O_{10} + (1-x_1)O_{00}})) \\
& - V_{m'}^t(\Gamma_1(\frac{x_1 O_{11}}{x_1 O_{11} + (1-x_1)O_{01}})) + V_m^t(\Gamma_1(\frac{x_1 O_{11}}{x_1 O_{11} + (1-x_1)O_{01}}))) \\
\geq & 0
\end{aligned}
$$

(ii) We then prove that $V_{m'}^{t+1}(0) - V_m^{t+1}(0) - V_{m'}^{t+1}(1) + V_m^{t+1}(1) \leq m' - m$

$$
\begin{aligned}
& V_{m'}^{t+1}(0) - V_m^{t+1}(0) - V_{m'}^{t+1}(1) + V_m^{t+1}(1) \\
=& \max\{m' + \beta V_{m'}^t(\Gamma_0(0)), O_{01} + \beta V_{m'}^t(\Gamma_1(0))\} \\
& - \max\{m + \beta V_m^t(\Gamma_0(0)), O_{01} + \beta V_m^t(\Gamma_1(0))\} \\
& - \max\{m' + \beta V_{m'}^t(\Gamma_0(1)), O_{11} + \beta V_{m'}^t(\Gamma_1(1))\} \\
& + \max\{m + \beta V_m^t(\Gamma_0(1)), O_{11} + \beta V_m^t(\Gamma_1(1))\}
\end{aligned}
$$

- Case 1: $m' + \beta V_{m'}^t(\Gamma_0(0)) \geq O_{01} + \beta V_{m'}^t(\Gamma_1(0))$ and $m + \beta V_m^t(\Gamma_0(1)) \geq O_{11} + \beta V_m^t(\Gamma_1(1))$

$$
\begin{aligned}
& V_{m'}^{t+1}(0) - V_m^{t+1}(0) - V_{m'}^{t+1}(1) + V_m^{t+1}(1) \\
\leq& m' + \beta V_{m'}^t(\Gamma_0(0)) - m - \beta V_m^t(\Gamma_0(0)) - m' - \beta V_{m'}^t(\Gamma_0(1)) + m + \beta V_m^t(\Gamma_0(1)) \\
=& \beta(V_{m'}^t(\Gamma_0(0)) - V_m^t(\Gamma_0(0)) - V_{m'}^t(\Gamma_0(1)) + V_m^t(\Gamma_0(1))) \\
\leq& \beta(V_{m'}^t(0) - V_m^t(0) - V_{m'}^t(1) + V_m^t(1)) \leq m' - m
\end{aligned}
$$

- Case 2: $m' + \beta V^t_{m'}(\Gamma_0(0)) \geq O_{01} + \beta V^t_{m'}(\Gamma_1(0))$ and $m + \beta V^t_m(\Gamma_0(1)) < O_{11} + \beta V^t_m(\Gamma_1(1))$

$$
\begin{aligned}
& V^{t+1}_{m'}(0) - V^{t+1}_m(0) - V^{t+1}_{m'}(1) + V^{t+1}_m(1) \\
\leq & m' + \beta V^t_{m'}(\Gamma_0(0)) - m - \beta V^t_m(\Gamma_0(0)) - O_{11} - \beta V^t_{m'}(\Gamma_1(1)) + O_{11} + \beta V^t_m(\Gamma_1(1)) \\
= & m' - m + \beta(V^t_{m'}(\Gamma_0(0)) - V^t_m(\Gamma_0(0)) - V^t_{m'}(\Gamma_1(1)) + V^t_m(\Gamma_1(1))) \leq m' - m
\end{aligned}
$$

- Case 3: $m' + \beta V^t_{m'}(\Gamma_0(0)) < O_{01} + \beta V^t_{m'}(\Gamma_1(0))$ and $m + \beta V^t_m(\Gamma_0(1)) \geq O_{11} + \beta V^t_m(\Gamma_1(1))$

  This case is impossible. This case means $V^{t+1}_{m'}(0; a = 0) < V^{t+1}_{m'}(0; a = 1)$ and $V^{t+1}_m(1; a = 0) \geq V^{t+1}_m(1; a = 1)$. $V^{t+1}_{m'}(0; a = 0) < V^{t+1}_{m'}(0; a = 1)$ leads to $V^{t+1}_{m'}(1; a = 0) < V^{t+1}_{m'}(1; a = 1)$ since it is a threshold policy for $V^{t+1}_{m'}(x)$ with smaller x leads to $a = 0$ and higher x leads to $a = 1$. Add $V^{t+1}_m(1; a = 0) \geq V^{t+1}_m(1; a = 1)$ and $V^{t+1}_{m'}(1; a = 0) < V^{t+1}_{m'}(1; a = 1)$ together contradicts the condition that $V_{m'}(x; a = 0) - V_m(x; a = 0) \geq V_{m'}(x; a = 1) - V_m(x; a = 1), \forall x$

- Case 4: $m' + \beta V^t_{m'}(\Gamma_0(0)) < O_{01} + \beta V^t_{m'}(\Gamma_1(0))$ and $m + \beta V^t_m(\Gamma_0(1)) < O_{11} + \beta V^t_m(\Gamma_1(1))$

$$
\begin{aligned}
& V^{t+1}_{m'}(0) - V^{t+1}_m(0) - V^{t+1}_{m'}(1) + V^{t+1}_m(1) \\
\leq & O_{01} + \beta V^t_{m'}(\Gamma_1(0)) - O_{01} - \beta V^t_m(\Gamma_1(0)) - O_{11} - \beta V^t_{m'}(\Gamma_1(1)) + O_{11} + \beta V^t_m(\Gamma_1(1)) \\
= & \beta(V^t_{m'}(\Gamma_1(0)) - V^t_m(\Gamma_1(0)) - V^t_{m'}(\Gamma_1(1)) + V^t_m(\Gamma_1(1))) \leq m' - m
\end{aligned}
$$

(iii) We finally prove that $V_{m'}^{t+2}(x; a = 0) - V_m^{t+2}(x; a = 0) \geq V_{m'}^{t+2}(x; a = 1) - V_m^{t+2}(x; a = 1), \forall x$

$$
\begin{aligned}
& V_{m'}^{t+2}(x; a=0) - V_m^{t+2}(x; a=0) - V_{m'}^{t+2}(x; a=1) + V_m^{t+2}(x; a=1) \\
=& m' + \beta V_{m'}^{t+1}(\Gamma_0(x)) - m - \beta V_m^{t+1}(\Gamma_0(x)) \\
& - xO_{11} - (1-x)O_{01} - \beta[xO_{11} + (1-x)O_{01}]V_{m'}^{t+1}(\Gamma_1(\frac{xO_{11}}{xO_{11} + (1-x)O_{01}})) \\
& - \beta[xO_{10} + (1-x)O_{00}]V_{m'}^{t+1}(\Gamma_1(\frac{xO_{10}}{xO_{10} + (1-x)O_{00}})) \\
& + xO_{11} + (1-x)O_{01} + \beta[xO_{11} + (1-x)O_{01}]V_m^{t+1}(\Gamma_1(\frac{xO_{11}}{xO_{11} + (1-x)O_{01}})) \\
& + \beta[xO_{10} + (1-x)O_{00}]V_m^{t+1}(\Gamma_1(\frac{xO_{10}}{xO_{10} + (1-x)O_{00}})) \\
=& m' - m + \beta(V_{m'}^{t+1}(\Gamma_0(x)) - V_m^{t+1}(\Gamma_0(x))) \\
& - \beta[xO_{11} + (1-x)O_{01}](V_{m'}^{t+1}(\Gamma_1(\frac{xO_{11}}{xO_{11} + (1-x)O_{01}})) - V_m^{t+1}(\Gamma_1(\frac{xO_{11}}{xO_{11} + (1-x)O_{01}}))) \\
& - \beta[xO_{10} + (1-x)O_{00}](V_{m'}^{t+1}(\Gamma_1(\frac{xO_{10}}{xO_{10} + (1-x)O_{00}})) - V_m^{t+1}(\Gamma_1(\frac{xO_{10}}{xO_{10} + (1-x)O_{00}}))) \\
\geq& m' - m + \beta(V_{m'}^{t+1}(1) - V_m^{t+1}(1) - V_{m'}^{t+1}(0) + V_m^{t+1}(0)) \\
\geq& m' - m + \beta(m - m') \geq 0
\end{aligned}
$$

□

Now we are ready to prove Theorem 6.2.2.

Theorem (6.2.2)**.** *When $\alpha\beta \leq 0.5$ and $\Gamma_1(1) \leq \Gamma_0(0)$, the process is indexable, i.e., for any belief x, if $V_m(x; a = 0) \geq V_m(x; a = 1)$, then $V_{m'}(x; a = 0) \geq V_{m'}(x; a = 1)$, $\forall m' \geq m$*

Proof. It follows from Lemma B.3.3 that when $\alpha\beta \leq 0.5$ and $\Gamma_1(1) \leq \Gamma_0(0)$, $V_{m'}(x; a = 0) - V_m(x; a = 0) \geq V_{m'}(x; a = 1) - V_m(x; a = 1), \forall x$. So if $V_m(x; a = 0) \geq V_m(x; a = 1)$, then $V_{m'}(x; a = 0) \geq V_{m'}(x; a = 1)$, $\forall m' \geq m$. □