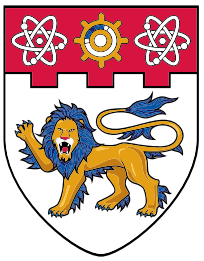


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

DEFENDING ON NETWORKS: APPLYING GAME THEORY TO PREVENT ILLEGAL ACTIVITIES IN STRUCTURED SECURITY DOMAINS

XINRUN WANG

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

NANYANG TECHNOLOGICAL UNIVERSITY

2019

DEFENDING ON NETWORKS: APPLYING GAME THEORY TO PREVENT ILLEGAL ACTIVITIES IN STRUCTURED SECURITY DOMAINS

XINRUN WANG

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

A thesis submitted to the Nanyang Technological University

in partial fulfillment of the requirement for the degree of

Doctor of Philosophy

2019

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

Xinrun Wang

July 11, 2019

....

Date Xinrun Wang

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

July 11, 2019

.....

Date

.....

Bo An

Authorship Attribution Statement

This thesis contains material from 3 papers published in the following peer-reviewed journals where I was the first and/or corresponding author.

Chapter 3 is published as **Xinrun Wang**, Qingyu Guo and Bo An. Stop nuclear smuggling through efficient container inspection. *Proceedings of the 16th International Joint Conference on Autonomous Agents and Multi-Agent Systems* (**AAMAS**), pp.669-677, 2017. The contributions of the co-authors are as follows:

- Prof. Bo An provided the initial idea of this work;
- Dr. Qingyu Guo and I investigated the problems and built the model;
- I justified the model, proposed and implemented the algorithms;
- I wrote the manuscript and Dr. Qingyu Guo revised the manuscript;
- Prof. Bo An provided critical comments to the paper.

Chapter 4 is published as **Xinrun Wang**, Bo An, Martin Strobel and Fookwai Kong. Catching Captain Jack: Efficient time and space dependent patrols to combat oil-siphoning in international waters. *Proceedings of the 32nd AAAI Conference on Artificial Intelligence* (**AAAI**), pp.208-215, 2018. The contributions of the co-authors are as follows:

- Prof. Bo An provided the initial idea of this work;
- Mr. Martin Strobel and I investigated the problems, built and justified the model;
- I proposed and implemented the algorithms and conducted the experiments;

- Mr. Fookwai Kong provided the data and provided useful comments;
- I wrote the manuscript and Prof. Bo An and Mr. Martin Strobel revised the paper.

Chapter 5 is published as **Xinrun Wang**, Bo An, Hau Chan. Who should pay the cost: A game-theoretic model for government subsidized investments to improve national cybersecurity. *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (**IJCAI**), accepted, 2019. The contributions of the co-authors are as follows:

- I proposed the problem and the model;
- Prof. Bo An provided critical comments to the model and I justified the model;
- I proposed the algorithms and wrote the manuscript;
- Prof. Bo An and Prof. Hau Chan provided insightful comments to the manuscript;
- Prof. Hau Chan and I revised the manuscript.

Xinrun Wang

July 11, 2019

....

Date Xinrun Wang

Abstract

The past decade witnesses the success of applying game theory to address security issues in domains such as protecting critical infrastructure and wildlife. However, the structured security domains such as interdicting illegal smuggling on networks, combating oil siphoning in international waters and preventing cyber attacks in cyber space are not well investigated. There are many critical challenges in these scenarios such as the scalability of the algorithm to solve realistic-sized problems, the long-term planning ability of the smugglers and the self-interested interdependent behaviors of users in cyber spaces. The previous works cannot be directly applied to solve these problems. Therefore, in this thesis, we provide models and solution approaches to several significant structured security domains.

First, we focus on preventing nuclear smuggling on container shipping networks through efficient container inspection. To stop the nuclear smuggling through container shipping networks, the U.S. government launched several initiatives to deploy advanced devices and inspection facilities in ports around the world. However, it is a significant challenge for the government to efficiently allocate the limited inspection resources to combat the nuclear smuggling activities due to the large volume of the containers imported to the U.S.. To address this problem, we propose a novel Container Inspection Model (CIM) based on Stackelberg game where the inspector determines her strategy first and the smuggler follows a Markov Decision Process (MDP) to determine his strategy after observing the inspector's strategy. Several important contributions are made to solve this problem including: i) a linear relaxation approximation which reformulates the problem into a bilinear optimization problem, ii) an algorithm based on the Multipleparametric Disaggregation Technique (MDT) to solve the bilinear program, and iii)

an iterative algorithm to further improve the scalability. Extensive experiments show that our algorithms outperform existing methods in the scalability significantly and can obtain a solution better than baselines, even under various uncertainties.

Second, we focus on preventing oil siphoning in international waters through efficient patrols. Pirate syndicates capturing tankers to siphon oil has become a serious security issue for maritime traffic. In response to the threat, coast guards and navies deploy patrol boats to protect international oil trade. However, given the vast area of the sea and the highly time and space dependent behaviors of both players, it remains a significant challenge to find efficient ways to deploy patrol resources. We provide four key contributions to address this challenge. First, we construct a Stackelberg model of the oil-siphoning problem; Second, we propose a compact formulation and a constraint generation algorithm to compute efficient strategies of security agencies; Third, to further improve the scalability, we propose an abstraction method to solve extremely large-scale games; Finally, extensive simulations and a detailed case study demonstrate that our approach achieves a dramatic improvement of scalability with modest influence on the solution quality and can scale up to realistic-sized problems.

Finally, we focus on preventing cyber attacks on interdependent users through government's subsidy. Cybersecurity is becoming one of the most critical concerns to the modern society due to the recent cyber attacks. Therefore, governments around the world launch various initiatives to improve the companies' investments to cybersecurity. However, the government has limited resources and needs to optimally allocate resources to companies. We introduce a game-theoretic model between the government, interdependent users (e.g., companies) and the cyber attacker. We investigate the three cases where the attacker can attack all, single and multiple users and propose a reverse convex formulation, an exact polynomial algorithm and a heuristic algorithm for the three cases, respectively. We extensively evaluate our model and algorithms on synthetic and real data. The results show that our model captures the interdependent behaviors of users and our heuristic algorithm converges efficiently under mild convergence criteria and outperforms the baselines with good robustness.

Acknowledgements

A long journey is done.

Pursuing a PhD degree is definitely the most important decision I have made in my past life. The four-year study teaches me how to be a professional researcher to tackle the problems, a grown-up to work with other people and a mature guy to get along with myself. I have learnt from many brilliant people, who taught, helped and inspired me. It is time to express my sincere gratitude to them.

First of all, I want to thank my supervisor Bo An. Without your supervision, this point can not be reached. Thank you for giving me the opportunity to pursue my PhD degree under your supervision. Four years ago, I joined your group without much knowledge about research. Thanks for your endless patience, insightful guidance and constant encouragement during every step of doing research. I am also deep impressed and motivated by your enthusiasm and profession to the research and those will be the fundamental things I will strive for in my future career. You reshapes my understanding about the research with your commitment to work on problems with impacts to the real-world and benefits to the society. Your critical comments always deepen my understanding on the research problems and the discussions with you always benefit me a lot. Not only on about research, I also learn a lot about the principles for the life: the way to dealing with failures because "rejection is a part of your academic life", the willingness to chase perfection and excellence because we always fight until the last minute of the conference deadlines and the love to Sichuan cuisine. Let me quote that "you set the pace and I will try to adjust to you". I will be forever grateful to your supervision and I hope I can let you be proud of me someday in the future.

I would like to thank my Thesis Advisory Committee (TAC) members: Gao Cong, Rongxing Lu, Yilin Mo and Xiaohui Bei. Those discussions help me to improve the

works and build a better vision of my research. Your guidance are valuable for me and I will always be thankful for your efforts in advising me.

During the past four years of research, I am grateful to have the privilege to collaborate with so many excellent researchers: Qingyu Guo, Martin Strobel, Fookwai Kong, Joachim Meyer, Hau Chan, Branislav Bošanský and Milind Tambe. I thank you for your guidance and insights in those projects, as well as the great patience of mentoring.

It was fortunate for me to share my PhD experience with a group of talented people: Haipeng Chen, Yanhai Xiong, Qingyu Guo, Mengchen Zhao, Zhen Wang, Jiarui Gan, Jiuchuan Jiang, Martin Strobel, Youzhi Zhang, Wanyuan Wang, Lei Feng, Xu He, Aye Phyu, Hongxin Wei, Wei Qiu, Rundong Wang, Jakub Černý and Runsheng Yu. Thank you all for those moments of laughs, happiness and celebrations and your generous help. It is an unforgettable memory of attending conferences, fighting for conference deadlines, having internal group meetings, going hiking and enjoying group treatments with you. Special thanks to Zhen Wang for his excellent guidance in my first year and to Qingyu Guo for his valuable help during the preparation of my first paper.

I also want to thank many of my precious friends, who are not around me but always with me. They are: Hongyu Feng, Bo Li, Hongchen Liu, Xinxin Liu, He Shi, Xuewei Wang, Rui Wang, Jiewei Yan, Pengfei Yang, Xiaoyue Yang, Weina Yu, Han Zhuang. Special thanks to Xiaoqing Yu for "when I doubted the world, you gave me the answer".

Lastly but most importantly, I want to thank my parents and my sisters for your unconditional love and supports. Thank you for always believing in me and sharing with my happy and tough time.

A new journey is beginning.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Security is one of the most important problems in the modern society, including the protection of critical infrastructures, i.e., buildings, coast ports and airports, large-scale public events and the patrolling of urban areas, international waters against pirates and nature reserve against poachers, interdiction of drug smuggling and nuclear smuggling and preventing the cyber attacks in cyber space. One of the major challenges in those problems is the insufficiency of the defending resources. The defending resources can be referred to the checkpoints, polices, inspection devices and facilities and patrollers. Therefore, it is important for the security agencies (i.e., defenders) in these domains to optimally or efficiently allocate the limited resources to combat the illegal activities. This is challenging mainly due to the complexity of the security domains and the strategic behaviors of the attacker because the attacker can take surveillance on the defender's daily executions and learn the patten of the defender's behaviors even before executing the attack. Therefore, a randomized allocation of the defending resources is required to avoid the exploitation of the attacker. Several other issues such as the attacker's bounded rationality and various uncertainties in the real world make the problem more difficult.

The past decade witnesses the celebrating success of applying game theory to tackle real-world security problems. Among them, one of the prominent models is called Stackelberg security game, which becomes the standard methodology in complicated security domain and the central role of several deployed systems protecting airports,

ports and wildlife [2–5]. Specifically, in a Stackelberg security game, there is a defender who protects valuable targets with limited resources and an attacker who attacks one or multiple targets. Instead of assuming that both players move simultaneously, in a Stackelberg security game, the defender commits to a randomized allocation strategy first and the attacker, after observing the randomized strategy through sufficient surveillances, launches his attack optimally. A vast of the effort is put to designing scalable algorithms to compute the optimal strategy of the defender in large-scale security domains [6–9]. Several works focus on study of the human behavioral modelling to predict the attacker's behaviors, by incorporating the well-studied theory in psychological literature prospect theory and quantal response equilibrium, which is applied in protecting ports and wildlife [4, 10]. A recent trend is moving the focus from physical security to cyber security [11] and the potential applications include packet selection and inspection [12], the placement of "honeypots" [13] and the allocation of cyber alerts[14].

The existing works of Stackelberg security game can be divided into two categories. The first category is the game without the structure of targets, where the targets only differ from their values and there is no geographical relations or connections between targets. The second category is the game with the structure of targets such as the security game on transportation networks [8], terrorist networks [15] and nature reserve with spatial features [5]. For example, Wang et al. consider to detect the terrorist plots on the terrorist network where the defender can randomly select the terrorists to monitor and the attackers have to form a connected subgraph to execute the coordinated attacks [15].

Along with the direction of structured security game, in this thesis, we extend the Stackelberg security game methodologies to three other specific scenarios: i) the container inspection problem to prevent nuclear smuggling on container shipping networks, ii) the patrol problem to prevent oil siphoning in the international waters and iii) the government subsidy assignment problem to prevent cyber attacks on cyber users' connection networks. In the rest part of this chapter, we present a brief review of the problems, the models and the contributions made in this thesis.

1.1 Preventing Nuclear Smuggling

The first part of this thesis focuses on preventing nuclear smuggling on container shipping network through efficient container inspection. Container has been a critical method for smugglers and terrorists to smuggle illegal goods including guns, weapons of mass destruction (WMD), dirty bombs and nuclear materials from one country to another. To prevent the activities of nuclear smuggling, the U.S. government has deployed various initiatives at both domestic and international ports, such as Container Security Initiative (CSI) [16], Megaports initiative [17] and Secure Freight Initiative (SFT) [18], which deploy devices and staffs to inspect containers at ports with inspection devices and facilities and security officers. However, due to the large volume of containers imported into the U.S. per year, about 17.5 million [19], only a small proportion of containers (less than 20%) can be inspected carefully with the sophisticated facilities such as radiation and spectroscopic portal monitors, by the inspector at ports, while other containers are only under the fast and simple non-intrusive inspection, which cannot provide reliable information to detect nuclear material if the material is shielded and under some threshold. Therefore, it is extremely important for the inspector to combat the nuclear smuggling activities through efficiently allocating the limited inspection resources over shipping lines.

What makes the case even worse is that the sophisticated smuggler can observe the inspector's strategy at ports through extensive observations and choose the optimal shipping lines to reduce the risk of being interdicted. Therefore, the following reasons make the problem even more difficult: i) the smuggler can take a number of containers, rather than put all material into one container, to smuggle illegal material through several shipping lines, which causes an exponentially large number of smuggler's action space; ii) the smuggler will transport the illegal containers sequentially, rather than at the same time, to reduce the risk of being interdicted by the inspector; therefore, a long-term and sequential smuggling plan are preferred, making the smuggler's decision process even more difficult to infer; iii) in order to quickly respond to emergencies after the interdiction of illegal containers, the inspection process needs to operate under different modes such as normal and emergent modes; and iv) the smuggler's decision-making process

is largely influenced by the inspector's strategy, which makes the computation of the inspector's strategy become a large-scale non-convex optimization.

There are several previous works which consider the container inspection problem. Some research focused on applying the optimization method to design more efficient inspection apparatus for more reliable inspection results [20, 21]. Some works proposed advanced inspection processes to take advantage of the limited resources at ports to speed up [22] while others applied game theoretical methods to help the inspector to select which containers to be inspected [23–25]. However, previous works do not consider the strategic behaviors of the smuggler and the allocation of the inspection resources may not be efficient to combat the nuclear smuggling.

Different from previous works, we tackle these challenges of efficiently combating sophisticated nuclear smuggling on the shipping network novelly and several key contributions are made. We propose a realistic container inspection model (CIM) where the inspector allocates the inspection resource to shipping lines to conduct the inspection. If the inspector interdicts an illegal, an emergency condition is triggered where additional inspection resources are added. The smuggler is assumed to know the inspection strategy and makes a long-term plan to smuggle the illegal containers, where the smuggler's decision-making process of is modeled as a Markov Decision Process (MDP).

We formulate the inspector's optimization problem as a non-convex program with exponentially many constraints. We propose several novel approaches to solve the non-convex program, including a linear relaxation approximation which reformulates the problem into a bilinear optimization problem with the guarantee of the solution quality, an algorithm based on the Multipleparametric Disaggregation Technique (MDT) [26] to solve the bilinear program, and a novel iterative method to incrementally add constraints into consideration to further improve the scalability. We conduct extensive experiments on both synthetic and real shipping networks and show that our approaches can scale up to realistic-sized problems and the solution outperforms the baselines robustly.

1.2 Preventing Oil Siphoning

The second part of this thesis focuses on preventing oil siphoning in international waters through efficient patrols. Pirate syndicates capturing tankers to siphon oil has become one of the critical issues in international waters. The estimated worldwide economic damage due to piracy reaches from $5 to $12 billion a year and a major part of the attacks occur in the South China Sea (SCS) [27, 28]. To combat the threat of piracy in the SCS, Singapore, Malaysia and Indonesia plan to establish patrols in the region [29]. However, given the vast area of the SCS, the huge number of merchant ships passing through and the limited number of patrol boats, the question of how to efficiently deploy patrol resources is extremely challenging to the security agencies. One of the most challenging issues is that the problem is highly time and space dependent because both the defender and the attacker take paths as their strategies where the number of possible paths, i.e., pure strategies, for both players are huge.

To address this problem, we construct a Stackelberg Model of the Oil-Siphoning problem (SMOS), where both players take time-dependent paths on the grid as their strategies, based on the incident reports from actual attacks as well as special reports conducted by maritime authorities to make the model realistic and computable. In order to compute the efficient defender's patrol strategy, we propose a compact formulation, which considers the probability that a zone is reached by the patrol boats rather than the probability that a patrol path is selected, and a constraint generation (CG) algorithm, which avoids the exponentially increasing number of pure strategies of the defender and the attacker, respectively.

To further improve the scalability, we propose an abstraction method to solve the extremely large-scale problem, which exploits intrinsic properties of the defender's strategy space to reduce the size of the game and makes a tradeoff between scalability and optimality. Specifically, we abstract the original grid into a small grid and compute the optimal defender's strategy on the small grid and then map the defender's strategy into the original grid. Finally, we evaluate our approaches through extensive simulations and a detailed case study with real ship traffic data. The results demonstrate that our approaches can scale up to realistic-sized problems with modest influence on the

solution quality. Our solution significantly outperforms existing methods, and is robust enough against the uncertainties in the real world. Our abstraction method provides a flexible framework to trade-off between the scalability and the solution quality.

1.3 Preventing Cyber Attacks

The third part of this thesis focuses on preventing cyber attacks on interdependent users through government's subsidy. In recent decades, cybersecurity has become one of the most important issues of the modern society because most companies, organizations and even governments build their services upon computer systems, and cyber attacks can cause extremely high loss to them. For instance, a single cybersecurity incident brings an average loss of $1.3 million for large businesses and $117,000 for small and medium businesses in North America [30]. The 2017 WannaCry ransomware attack affected more than 300,000 computers across 150 countries with a total damage $4 billion [31]. It is obviously clear that from these recent cyber attacks, businesses are not doing enough to protect themselves and/or do not have sufficient awareness of cyber threats. In fact, 80% of over 400 global companies do not know where their sensitive data is located and how to secure it, and 60% of them do not protect their privileged accounts adequately [32]. Because the companies and organizations are interdependent in cyber space, the cyber attacks could propagate from one company to others.

To combat these issues, the governments around the world have launched various initiatives to improve the national cybersecurity level. For example, the UK government, one of the pioneering countries on national cybersecurity practice, launched Cyber Essentials scheme in 2014 and provided millions of subsidies, i.e., grants and vouchers, for businesses to boost their cybersecurity in 2015 and 2016 [33, 34]. The government's goal is to allocate subsidies to companies so as to maximize the national cybersecurity level. However, the government subsidy budget is limited and the highly interdependence of self-interested cyber companies, whose investment actions/decisions in protection will be largely influenced by the investment decisions of their neighboring companies due the spreadability of cyber attacks in the cyber space, creates challenges for the government to optimally assign the subsidies to companies. In this work, we

use game-theoretic methodologies to analyze the interactions between companies and a cyber attacker and help the government to improve the national cybersecurity.

To address this problem, we formulate our cybersecurity setting as a game played between the government, many users and an attacker. The government moves first by determining the allocation of subsidy to users, then the users decide whether or not to invest in cybersecurity to protect themselves and the attacker decides the users to attack, simultaneously. We consider the Stackelberg equilibrium between the leader, i.e., the government, and the followers, i.e., users and the attacker, and the Nash equilibrium between followers. This model extends both Stackelberg security game and interdependent security game by adding interdependence between users of the Stackelberg security game and a leader role, i.e., the government, into the model.

We comprehensively investigate three cases where the attacker can attack all users, a single user and multiple users theoretically and propose a linear reverse convex program for the case with all-user attacks, an exact polynomial algorithm for the case with single-user attacks and a heuristic algorithm based on best-response-gradient dynamics for the case with multiple-user attacks to compute the government's allocation strategy. Finally, we extensively evaluate our model and the heuristic algorithm on synthetic and real data to demonstrate the reasonability of model and the performance of our algorithms.

1.4 Thesis Overview

The organization of this thesis is as follows. Chapter 2 discusses the background material and related works of this thesis. Chapter 3 considers the problem of preventing nuclear smuggling on container shipping network through efficient container inspection. Chapter 4 considers the problem of preventing oil siphoning in international waters through efficient patrols. Chapter 5 considers the problem of preventing cyber attacks on interdependent users through government's subsidy. Chapter 6 concludes this thesis and provides future directions.

Chapter 2

Background

This chapter reviews the background knowledge which supports this thesis with fundamental theorems, models and algorithms. The topics discussed in this thesis fall into the research area called security games which applies game-theoretic methodologies and large-scale optimization techniques to optimize the allocation of limited resources to protect valuable targets in various real-world security domains.

We begin with the basic models and theorems in game theory in Section 2.1, which include the normal-form game and the canonical Nash equilibrium and the standard solution concept in security games Stackelberg equilibrium. We then discuss a special case called zero-sum in Section 2.1.2, where we introduce the well-known minimax theorem, which plays the fundamental role to exactly compute the Nash equilibrium in zero-sum game using the widely used methods, such as constraint generation, column generation and double oracle introduced in Section 2.1.2.2. After that, we extend the discussion to general-sum game in Section 2.1.3, which is much more difficult to exactly compute the Nash equilibrium in these game. Therefore, we introduce the widely used best-response-gradient dynamics to approximate the Nash equilibrium.

We then discuss the security game in Section 2.2, which is a special area of game theory. In Section 2.2.1, we present the basic assumptions and formulations in Stackelberg security game and an overview of the previous works in security game and we

discuss the interdependent security game in 2.2.2. As we leverage the optimization techniques to compute the solution concepts of our model, we also present a brief overview of nonconvex optimization in Section 2.3.

2.1 Game Theory

Game theory aims to help people to understand conflictive or cooperative situations in which rational decision-makers interact [35, 36]. The most famous representation in game theory is the *normal form game*, also named as *strategic form game*.

Definition 2.1. A normal form game is formed by

- the set of players $\mathcal{N} = \{1, 2, \ldots, N\}$,
- the set of *actions* or *pure strategies* $\mathcal{A} = \times_{i \in \mathcal{N}} \mathcal{A}_i$ where $\mathcal{A}_i$ is the set of actions for each player $i \in \mathcal{N}$,
- the payoff function $u_i : \mathcal{A} \to \mathbb{R}$ for each player $i \in \mathcal{N}$.

	c_1	c_2
r_1	2,1	4,0
r_2	1,0	3,1

TABLE 2.1: An example, from [1], of two-player two-action normal form game where the first value is for the row player and the second is for the column player.

Table 2.1 shows the payoff matrix of a simple two-player two-action normal form game. To analyse the possible outcomes of the game, we also need to determine other additional information such as the sequence of the player's moves and the reasonability of the players. In the simultaneous-move case, each player assumes that other players are perfect rational and choose the optimal action to maximize their own payoffs. For the game displayed in Table 2.1, the column player can reason that the row player will always play r_1 because the payoff of r_1 is always higher than r_2 no matter what the row player plays. Therefore, the row player will play c_1 to maximize his own payoff. The reached outcome $\langle r_1, c_1 \rangle$ is an equilibrium of the game where no player want to deviate unilaterally.

	c_1	c_2
r_1	1,-1	-1,1
r_2	-1,1	1,-1

TABLE 2.2: An example of normal form game with no pure strategy equilibrium.

The equilibrium with pure strategies does not exist in all games. An example without any pure strategy equilibrium is shown in Table 2.2. In the generalization of the equilibrium, we allow the player to randomize his action, which is called *mixed strategy*. The mixed strategy $\sigma_i \in \Delta_i$ is a probability distribution over all actions $\mathcal{A}_i$ where Δ_i is the set of all mixed strategies and $\sigma_i(a_j)$ is the probability that player i will play $a_j \in \mathcal{A}_i$. We denote the strategy profile of all players as $\boldsymbol{\sigma} = \langle \sigma_i, \sigma_{-i} \rangle$ where σ_{-i} is the strategy profile of the players except player i. The orthodoxical solution concept in simultaneous-move and non-cooperative game is *Nash equilibrium* (Definition 2.2) where no player has incentive to deviate by playing a different (mixed) strategy [37] . A celebrating result proved by Nash is that every normal form game with finite actions for players has at least a mixed strategy Nash equilibrium [37].

Definition 2.2. A strategy profile $\boldsymbol{\sigma}^*$ forms a Nash equilibrium if and only if

$$U_i(\sigma_i^*, \sigma_{-i}^*) \geq U_i(\sigma_i, \sigma_{-i}^*) \quad \forall i \in \mathcal{N}, \sigma_i \in \Delta_i. \tag{2.1}$$

where $U_i(\sigma_i, \sigma_{-i})$ is the utility, i.e., expected payoff, of the strategy profile $\langle \sigma_i, \sigma_{-i} \rangle$.

2.1.1 Stackelberg Game

Different from the simultaneous-move game, Stackelberg game, also called leader-follower game, assumes that players determine their actions or strategies sequentially [38]. Generally, there is a leader and a follower in the Stackelberg game, i.e., $\mathcal{N} = \{l, f\}$. The leader first commits to a mixed strategy σ_l and then the follower can observe the strategy committed by the leader and determine his strategy σ_f. The observability of the defender's strategy to the attacker seems a disadvantage against the defender, however, the commitment of the defender is proved to be a first-mover advantage to him [39]. For the example in Table 2.1, take the row player as the leader.

If the leader commit to r_2 and the column (i.e., follower) will choose c_2 to maximize his payoff. Therefore, the leader obtains 3, which is higher than the payoff in the Nash equilibrium payoff 2.

The typical solution concept in Stackelberg game is the *Strong Stackelberg Equilibrium (SSE)*. We define the follower's response function $g : \Delta_l \to \Delta_f$ which determines the follower's strategy given the leader's strategy as input. In SSE, the response function returns the follower's best response to the leader's strategy. When there are multiple best responses for the follower, the follower is assumed to break ties in favor of the leader. The formal definition of SSE is presented in Definition 2.3.

Definition 2.3. A strategy profile $\langle \sigma_l^*, g(\sigma_l^*) \rangle$ forms an SSE if and only if

- $U_l(\sigma_l^*, g(\sigma_l^*)) \geq U_l(\sigma_l, g(\sigma_l)) \quad \forall \sigma_l \in \Delta_l$,
- $U_f(\sigma_l, g(\sigma_l)) \geq U_f(\sigma_l, \sigma_f) \quad \forall \sigma_l \in \Delta_l, \sigma_f \in \Delta_f$,
- $U_l(\sigma_l, g(\sigma_l)) \geq U_l(\sigma_l, \sigma_f) \quad \forall \sigma_l \in \Delta_l, \sigma_f \in BR(\sigma_l)$.

$BR(\sigma_l) = \arg\max_{\sigma_f \in \Delta_f} U_f(\sigma_l, \sigma_f)$ is the set of the follower's best response strategies.

2.1.2 Zero-sum Game

Two-player zero-sum game is a special case of the game and closely related to the works presented in this thesis. In zero-sum games, the summation of the two players' payoffs are always zero, i.e., the two players are completely competitive. Zero-sum games have many nice properties for theoretical analysis and computation. One of the significant properties is that the Nash equilibrium, maximin and minimax are equivalent in zero-sum game, as well as the Stackelberg equilibrium. Therefore, we can compute the leader's optimal strategy in zero-sum games with the maximin linear program:

$$\max_{\sigma_l} \quad U_l \tag{2.2a}$$

$$\text{s.t.} \quad U_l \leq \sum_{a_l \in A_l} \sigma_l(a_l) u_l(a_l, a_f) \quad \forall a_f \in \mathcal{A}_f \tag{2.2b}$$

$$\sum_{a_l \in \mathcal{A}_l} \sigma_l(a_l) = 1 \tag{2.2c}$$

$$\sigma_l(a_l) \geq 0. \tag{2.2d}$$

2.1.2.1 Minimax Theorem

In this section, we present the minimax theorem, which provides a useful tool for us to analyse the double oracle algorithm introduced in Section 2.1.2.2. The minimax theorem, in particular the von Neumann's minimax theorem, states that for two-player game with finite pure strategies, the player's values for minimax equilibrium is equal to the values for maximin equilibrium.

Theorem 2.4 (von Neumann's minimax theorem)**.** *For two-player zero-sum game with* $N = \{l, f\}$ *and finite actions for each player, we have*

$$\max_{\sigma_l} \min_{\sigma_f} U_l(\sigma_l, \sigma_f) = \min_{\sigma_f} \max_{\sigma_l} U_l(\sigma_l, \sigma_f) = v. \tag{2.3}$$

2.1.2.2 Double Oracle

The LP (2.2) can be solved in time polynomial with the size of the payoff matrix. However, in many complicated models. The number of pure strategies of players can be huge. Even in the simplest case of the security game where the defender assigns K resources to protect N targets, the number of pure strategies of the defender is C_N^K, which grows exponentially with both K and N. Double oracle algorithm is proposed to tackle the scalability problem of solve large-scale zero-sum game, especially in security games.

The basic idea of double oracle algorithm is to solve a restricted game with restricted strategy spaces of players, rather than the full game, and then incrementally enlarge the strategy spaces until the minimax equilibrium of the full game is obtained. The double oracle algorithm is depicted in Algorithm 1. The algorithm starts from a restricted game where only few random pure strategies for players, i.e., $\mathcal{A}'_l$ and $\mathcal{A}'_f$, are considered. The restricted game is solved by LP (2.2), denoted as *CoreLP*, obtaining the (Nash) equilibrium profile $\langle \sigma_l, \sigma_f \rangle$ of the restricted game. We note that the equilibrium for the

Algorithm 1: Double Oracle Framework

```
Initialize A'_l with random leader actions;
Initialize A'_f with random follower actions;
repeat
    ⟨σ_l, σ_f⟩ ← CoreLP(A'_l, A'_f);
    A'_l ← A'_l ∪ {LO(σ_f)};
    A'_f ← A'_f ∪ {FO(σ_l)};
until convergence;
return ⟨σ_l, σ_f⟩.
```

restricted game may not be the equilibrium for the original game. Thus, two oracles are called to solve the best responses for both players in $\mathcal{A}_l$ and $\mathcal{A}_f$, denoted by LO and FO, respectively. The best responses, i.e., new pure strategies, are added to $\mathcal{A}'_l$ and $\mathcal{A}'_f$. The procedure is repeated until convergence.

The convergence criterion usually is that the best responses already returned by both oracles are already existing in the restricted strategy spaces. As the pure strategy spaces of both players are assumed to be finite, the algorithm will terminated in finite iterations. The obtained strategy profile is guaranteed to be the equilibrium of the original game by the minimax theorem [40]. In the worst case, the algorithm has to enumerate all pure strategies for both players before the termination, however, in practice, there are only a few pure strategies with positive probabilities to be executed and the algorithm can terminate after a few iterations. Most of the works in the security focus on designing efficient oracles to compute the best responses for both players, which is the computationally costly part of the algorithm [7, 41].

2.1.3 General-sum Game

Previously, we introduce the zero-sum game and the related methods to compute the equilibrium. In this section, we extend the discussion to general-sum game, which is a more general case of the game. Though we can find the equilibrium in zero-sum games in polynomial time, it is PPAD-complete to find the mixed Nash equilibrium in the general-sum game [42]. The most known algorithm to compute the Nash equilibrium in two-player general-sum game is Lemke-Howson algorithm [43]. We refer readers to [44] for more details of the algorithm.

2.2 Security Game

In this section, we present two lines of research to apply game theory to security domains. One is the well-known Stackelberg security game with the algorithmic contributions and deployed applications to the real world and the other is interdependent security game which is more suitable to model the domains where users are interdependent and self-interested, such as cyber security domains.

2.2.1 Stackelberg Security Game

The critical challenge in most of the security domains is that the defender does not have sufficient resources, such as patrollers, inspection devices and subsidies. Besides, when allocating the limited resources, the defender has to consider the attacker's surveillance of the allocating strategy, as well as the adversarial behaviors. Security game leverages the game-theoretic methodology to guide the security agencies to allocate the security resources. The Stackelberg security game (SSG) is the simplest and most widely investigated model applied to various security domains. Basically, the SSG is the game between a defender who wants to protect a set of targets with limited and insufficient resources and an attacker who want to attack one or more targets. The targets can be public infrastructure, ports and air ports, groups of people and vessels, areas with endangered animals and even computer companies. The pure strategy of the defender is a fixed and specific allocation of the resource and the attacker's pure strategy is a set of targets to be attacked. The mixed strategies of both players follow the definition of mixed strategy in Section 2.1. This basic model is the backbone of many other complicated models where the players' strategies change from one to another and the targets may have specific structures. SSG adopts the Stackelberg game to model the interaction between the players, where the defender moves first and commits to a mixed strategy and the attacker can fully observe the defender's mixed strategy[1] and take best responses. The solution concept considered in SSG is the Strong Stackelberg Equilibrium (SSE) (Definition 2.3).

[1]The attacker's full observability is justified by the unlimited surveillance of the realizations of pure strategies sampled from the committed strategy before the attacker takes attacks.

The SSG model was extensively studied and played the central role of several deployed applications to security domains in the past decade. Inspired by the seminal work [1] which studies the computational complexity of the optimal commitment in Stackelberg games, the DOBSS algorithm is proposed to compute the SSE in Bayesian security games by mixed-integer linear programs [6]. DOBSS becomes the core of the deployed ARMOR system to schedule the allocation of checkpoints at exits of the LAX airport [3]. Since then, a vast amount of efforts are put into designing scalable algorithms to address large-scale real-world security problems. Some of them are the ERASER algorithm which exploits the payoff structure of security games [45], and the ASPEN algorithm which applies the branch and price framework to tackle large-scale security games with allocation constraints [46]. Those approaches are in many deployed systems, such as IRIS to assist the Federal Air Marshal Service (FAMS) with randomly scheduling air marshals on flights [47] and GUARDS to protect the airport on a nation scale [48].

2.2.1.1 Robustness and Human Behavior Modelling in Stackelberg Security Games

The deployment of the SSG based system tests the reasonability of the model, where researchers find critical issues of the model, which leads that the defender's strategy computed by the system may not be efficient against the attacker. Several of them are i) the uncertainties of attacker's payoff structure, i.e., the defender may not know the payoff of the attacker exactly, ii) the execution and observation uncertainties regarding the defender's and attacker's abilities, i.e., the defender may not correctly execute the strategy he commits to and the attacker may not perfectly know the defender's strategy and iii) more importantly, the bounded rationality of the attacker's behaviors.

To remedy these issues, Pita et al. [49] propose the algorithms based on the MAXMIN method to address the execution and observation uncertainties and Nguyen et al. [50] propose the minimax regret, instead of MAXMIN-based solution, as the robustness measure for dealing with the uncertainties. An et al. consider the cost of the

attacker's surveillance and propose a Markov decision process for the attack's decision-making process. Jiang et al. [52] study the uncertainty during the execution of the defender's strategy (i.e., interruptions of the patrols) and model the uncertainty using the Markov Decision Processes, which build the core of the TRUST system deployed in LA metro rail system to schedule the patrol for fare inspection [53].

Human behavior modelling is an important technique to exploit the attacker's bounded rationality. The works on human behavior modelling in security games are related to two fundamental theory from psychological literature: Prospect Theory [54] and Quantal Response Equilibrium (QRE) [55]. Yang et al. introduce the Quantal Response (QR) model into the security game context [56] and efficient algorithms to compute the QRE in security games are proposed in [57]. As a result, PROTECT system based on QRE solution is deployed to protect the ports of the United States [4]. Furthermore, Nguyen et al. integrate the subjective utility (SU) function into QR model and the resulting SUQR model outperforms the QR model to predict the attacker's behaviors as it captures the fact that humans put more weight on the probability of a success attack in their decision making process, rather than considering the expected utility [58]. The SUQR model is applied in PAWS to schedule the ranger's patrols to protect the wildlife reservation against poachers [10, 59].

2.2.1.2 Stackelberg Security Game on Structured Domains

The works discussed previously are aiming to tackle the explosive growth of the defender's strategy space, where the targets in these works do not possess any structures, i.e., the targets are only different by their values and their spatial or logical structures are ignored. However, a lot of security domains can be abstracted as networks or grids, such as preventing illegal nuclear smuggling on the container shipping networks, interdicting the oil siphoning boats through patrols and combating the cyber risks on the connection network of companies. On one hand, the structures of these domains makes the players' strategy spaces become extremely huge because the players' strategies become a subgraph, a path even a flow on the network. On the other hand, researchers can

design efficient algorithms to exploit the special structure to scale the algorithm even better.

Network interdiction problem is well-studied in the current literature [60], where many objectives are studied, including maximizing the shortest path and minimizing the network flow, on the given weighted or unweighted, directed or undirected graph [61–64]. Other works focused on the *stochastic network interdiction* where the interdicting action succeeds with some known probability [65]. Recently, some works from security game focused on the randomized resource allocation to interdict the escape path of the attacker or illegal network flow of drug smuggling [7, 8, 66]. In these works, the double oracle is widely used to eliminate the obstacles of exponential growth of the players' strategy space, which is also adopted by some other structured security games [15, 41].

Instead of using the double oracle algorithm, there are several works which focus on using abstraction techniques to reduce the strategy space size in structured security game[67–69]. Typically, abstraction techniques exploit action space similarity to build a simpler problem and solve it. After that, the solution of the simple problem will be mapped to a solution to the original problem. The abstraction can be lossy or lossless depending on the problem. For example, Basak et al. build a small graph with only key nodes on the original graph to compute the defender's patrol strategy on the small graph and then mapping the solution, i.e., paths from key nodes, to the original by selecting the shortest path between key nodes on the original graph [68].

2.2.2 Interdependent Security Game

A parallel and even earlier line of research for security problem is interdependent security game (IDS) [70]. IDS focus on the environment of self-interested interdependent decision-makers, which differs from the Stackelberg security game. Originally introduced by Kunreuther and Heal in [70], IDS is applied to analyse the airline baggage security in [71] where airlines need to choose whether to invest on additional complementary equipment to screen passengers' bags and check for hazards that could cause damage to their passengers, planes or buildings. Another application of IDS model is

screening the container on the container shipping network [72]. IDS focus on computing the Nash equilibrium between users. An important results proved in [73] is that there exists a pure strategy Nash equilibrium in IDS which can be computed in time polynomially with the number of users. However, it is NP-complete to find all pure strategy Nash equilibrium in IDS [73]. An important extension to IDS is interdependent defense game [74–76], where instead of only considering the users, an attacker is introduced who may strategically attack the users.

2.3 Nonconvex Optimization

As security game is finally formulated as an optimization problem, we introduce some basic knowledge of optimizations. Particularly, we focus on two kinds of nonconvex optimization problem, i.e., bilinear optimization and linear reverse convex optimization.

2.3.1 Bilinear Optimization

Continuous and mixed integer bilinear problems are often raised in applications such as pooling problem [77, 78], concave piecewise linear network flow problems [79] and multi-item dynamic pricing problems [80]. Bilinear program, in its general form, can be formulated as

$$\max \ \boldsymbol{x}^\top Q^\top \boldsymbol{y} + \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}^\top \boldsymbol{y} \tag{2.4a}$$

$$\text{s.t.} \ A\boldsymbol{x} \leq \boldsymbol{a}, B^\top \boldsymbol{y} \leq \boldsymbol{b}, \tag{2.4b}$$

$$\boldsymbol{x} \geq \boldsymbol{0}, \boldsymbol{y} \geq \boldsymbol{0} \tag{2.4c}$$

where $\mathcal{X} = \{\boldsymbol{x} | A\boldsymbol{x} \leq \boldsymbol{a}, \boldsymbol{x} \geq \boldsymbol{0}\}$ and $\mathcal{Y} = \{\boldsymbol{y} | B^\top \boldsymbol{y} \leq \boldsymbol{b}, \boldsymbol{y} \geq \boldsymbol{0}\}$ are assumed to be non-empty and bounded polyhedral sets. A common solution methodology is to construct polyhedral relaxations using envelopes of bilinear terms [81] within a spatial branch-and-bound framework [82]. Most of works fall in either building tighter relaxations [83, 84] or proposing new branching strategies [85]. Recently, a new reformulation technique is proposed in [26], which represents the bilinear terms with a set

of linear terms and constraints and finally transforms the bilinear programming into a mixed integer linear program (MILP).

2.3.2 Linear Reverse Convex Optimization

We now briefly introduce another kind of nonconvex problems, linear reverse convex optimization. The problem is formally formulated:

$$\min \ \boldsymbol{c}^\top \boldsymbol{x} \tag{2.5a}$$

$$\text{s.t.} \ A\boldsymbol{x} \leq \boldsymbol{b} \tag{2.5b}$$

$$h(\boldsymbol{x}) \geq 0 \tag{2.5c}$$

$$\boldsymbol{x} \geq 0 \tag{2.5d}$$

where $h : \mathbb{R}^n \to \mathbb{R}$ is a convex function. The problem can be understood as a linear program with additional reverse convex constraints, which is often encountered in engineering design problems [86] and economic managements [87]. We refer the reader to [88] for more details of this kind of problems.

Chapter 3

Preventing Nuclear Smuggling on Container Shipping Networks through Efficient Container Inspection

Container has been a critical method for smugglers and terrorists to smuggle illegal goods from one country to another. To prevent the activities of nuclear smuggling, the U.S. government has deployed various initiatives at both domestic and international ports to deploy devices and staffs to inspect containers at ports with inspection devices and facilities and security officers. However, due to the large volume of containers imported into the U.S. per year, only a small proportion of containers (less than 20%) can be inspected carefully with the sophisticated facilities by the inspector at ports, while other containers are only under the fast and simple non-intrusive inspection[1] which cannot detect nuclear material reliably if the material is shielded and under some threshold. Therefore, the inspector need to efficiently allocate the inspection resource to combat nuclear smuggling. However, the following reasons make the problem difficult: i) the smuggler can take a number of containers, rather than put all material into one container, to smuggle illegal material through several shipping lines, which causes an exponentially large number of smuggler's action space; ii) the smuggler will transport the illegal containers sequentially, rather than at the same time, to reduce the risk of being

[1]Non-intrusive inspection uses X-rays or gamma rays to scan containers and creates images of the contents in containers without opening them to identify anomalies among other goods.

interdicted by the inspector; therefore, a long-term and sequential smuggling plan are preferred, making the smuggler's decision process even more difficult to infer; iii) in order to quickly respond to emergencies after the interdiction of illegal containers, the inspection process needs to operate under different modes such as normal and emergent modes; and iv) the smuggler's decision-making process is largely influenced by the inspector's strategy, which makes the computation of the inspector's strategy become a large-scale non-convex optimization.

Container inspection has been investigated from different perspectives [20, 22–25]. There are some works focusing on designing more efficient inspection devices using the optimization methods for more reliable inspection results [20, 21]. Bakshi et al. investigated the influences to the ports if all containers are inspected by the inspector [22]. Bakır adopted the Stackelberg game to investigate the resource allocation problem of limited security resources to routes to prevent the smuggling activities in cargo containers [23]. Haphuriwat et al. investigated the proportion of containers to be inspected to deter the smuggling activities where containers are inspected uniformly [25]. However, only a small percentage of the enormous containers can be inspected at ports during the operation and both the smuggler and the inspector can take strategic behaviors to maximize their utility.

Network interdiction problem is well-studied in the current literature [60], where many objectives are studied, including maximizing the shortest path and minimizing the network flow, on the given weighted or unweighted, directed or undirected graph [61–64]. Other works focused on the *stochastic network interdiction* where the interdicting action succeeds with some known probability [65]. Recently, some works from security game focused on the randomized resource allocation to interdict the escape path of the attacker or illegal network flow of drug smuggling [7, 8, 66]. Unfortunately, none of them considers sophisticated smuggling activities and sequential decisions, and the game is assumed to be one-shot.

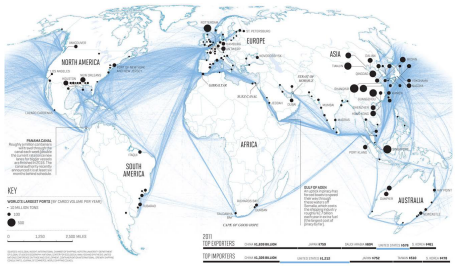
Several recent works in security games modeled the sequential decision process of the attacker as a Markov decision process (MDP) to deal with the long-term planning of the security problem [51, 89]. An et al. studied the adversary's sequential surveillances of the pure strategies sampled from the defender's random strategy before executing

the attacks where the number of states of the attacker's MDP is linear to the horizon of the time and the number of defender's actions [51]; Zhao et al. computed the optimal thresholds of email filtering systems of different users to prevent the long-term sequential phishing attacks where the number of the states in the attacker's MDP is linear to the targets' number. The problems in both works can be categorized in scenarios of protecting targets where the attacker needs to choose targets to attack [89]. Therefore, the dynamic programming algorithms can solve the MDPs in both works efficiently. While for defending on the networks, both the state space and action space of the attacker's MDP are exponentially large, which requires novel and efficient algorithms.

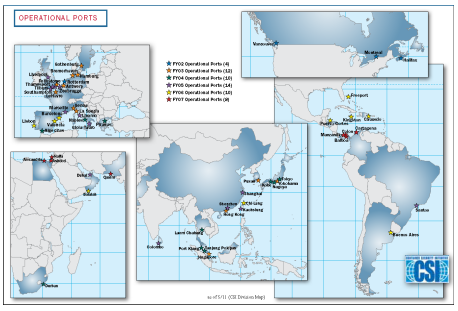
First, we build a novel container inspection model (CIM) where the inspector allocates the inspection resource to shipping lines to conduct the inspection while the smuggler makes sequential plans after observing the inspector's strategy, and the decision-making process of the smuggler is modeled as a Markov Decision Process (MDP). Second, we formulate the optimization problem of the inspector's strategy as a nonconvex program with exponentially many constraints. Third, we propose several novel approaches to solve the nonconvex problem, including a linear relaxation approximation which reformulates the problem into a bilinear optimization problem with the guarantee of the solution quality, an algorithm based on the Multipleparametric Disaggregation Technique (MDT) [26] to transform the bilinear terms into linear terms, and a novel iterative method to incrementally add constraints into consideration. Finally, we conduct extensive experiments on both synthetic and real shipping networks and show that our algorithms outperform existing methods in the scalability significantly and can obtain a robust solution better than baselines.

3.1 Motivation Scenario

While our model can be applied to various illegal container smuggling scenarios, we take the nuclear smuggling as an example to motivate the model in this section. More than 34 million containers are shipped all around the world per year [19] through the container shipping network displayed in Figure 3.1a. To combat nuclear smuggling on the maritime container shipping network, the U.S. government has launched various



(a) Container shipping networks.



(b) Ports of CSI.

(c) Gamma-ray inspection for NII.

(d) MRDIS.

FIGURE 3.1: Nuclear smuggling and its prevention.

international initiatives, including Container Security Initiative (CSI), Megaport Initiative (MI) and Security Fright Initiative (SFI) to inspect containers at ports. Figure 3.1b shows the 58 CSI ports around the world, which cover more than 80% of containers imported into the U.S. [16]. Due to the large volume of the containers imported to the U.S., most containers are under the fast and simple non-intrusive inspection (NII) by using the system shown in Figure 3.1c. However, if the smuggler divides the nuclear material into small amounts[1] and shields the nuclear material by other goods in the container, NII cannot provide reliable information to help the inspector to detect the material.

Therefore, the U.S. government deployed advanced facilities such as Mobile Radiation Detection and Identification System (MRDIS) as displayed in Figure 3.1d, which is a more sensitive facility to the nuclear material. However, as the inspection by MRDIS takes longer time than NII, only less than 20 percent of the containers can be inspected by MRDIS [17]. To quickly respond to the emergency after the interdiction of illegal containers, the government has developed emergency plans by adding more devices and

[1]Different from the illegal goods such as cigarettes or guns, where large volume of these goods are of concern, small amount of nuclear material is still a critical threats to the security of the country. For example, 25kg of HEU or 8kg of plutonium could be used to build nuclear weapons [90].

officers to inspect the containers. For example, seven different agencies of the Jamaica government will coordinate to quickly respond to the radioactive case in its emergency plan [91].

As we mentioned before, to avoid the interdiction by NII, the smuggler would divide the nuclear material into small units (e.g., 1 kg) and shield the material with other goods in the container [92]. Furthermore, the sophisticated smuggler would choose many shipping lines after observing the inspector's strategy and interrupt the smuggling plan when the inspection is under the emergency mode. Thus, a sequential plan is more beneficial to the smuggler, which brings more difficulties to interdict the illegal containers. Therefore, it is an extremely challenging task for the inspector to efficiently allocate the limited reliable inspection resources, e.g., MRDIS, to the shipping line and respond to the emergency after an interdiction of the illegal containers. In this chapter, we are going to compute the efficient allocation of the limited inspection resources to preventing the smuggling activities of the nuclear material through containers after knowing the number of illegal containers being transported by the smuggler.

3.2 Container Inspection Model

In this section, we now introduce the Container Inspection Model (CIM) where the inspector allocates the inspection resource to shipping lines to conduct the inspection while the smuggler makes sequential plans after observing the inspector's strategy, and the decision-making process of the smuggler is modeled as a Markov Decision Process (MDP).

The maritime shipping network in the CIM model is represented as a tuple $\mathcal{N} = \langle \mathcal{L}, \mathcal{P} \rangle$ where $\mathcal{L}$ is the set of shipping lines and $\mathcal{P}$ is the set of ports. We use $\alpha : \mathcal{L} \times \mathcal{P} \to \{0, 1\}$ to denote whether the ports is visited by the shipping lines or not, where $\alpha_{lp} = 1$ if port $p \in \mathcal{P}$ is visited by the shipping line $l \in \mathcal{L}$ and $\alpha_{lp} = 0$ otherwise. The container flow on the shipping line is denoted by $\mathbf{f} = \langle f_l \rangle$ where f_l is the average number of containers transported through the shipping line $l \in \mathcal{L}$ in the given time period. To incorporate the sequential decisions of both inspector and the smuggler, we

take τ as the unit period of the time to ship an illegal container to the target port[1] and the smuggler can make his decisions at $\{0, \tau, ..., t \cdot \tau, ...\}$ where t is the *time step*.

The smuggler strategically ships m ($m \leq |\mathcal{L}|$) illegal containers over different shipping lines where the inspector can estimate the value of m by the nuclear material lost by the institutions and governments. In line with existing works in security games [7, 8, 15, 66], we assume that the objective of the inspector is minimizing the smuggler's utility. We assume that if an illegal container is interdicted by the inspector, the payoff for both players are zero. We denote the payoffs of the smuggler and the inspector when an illegal container is successfully smuggled to the target ports through shipping line l as u_l^a and u_l^d, respectively and note that $u_l^d = -u_l^a$. W.l.o.g., we assume u_l^a larger than 0 for all shipping lines[2].

Inspector's Strategy: The inspector determines her allocation of the limited inspection resources at port by deciding the proportion of containers being inspected for shipping lines visiting each ports. To make the model more realistic, extra inspection resources will be put when illegal containers are interdicted by the inspector [91]. However, those resources cannot last for a long time because the cost is high for the government. Therefore, we assume that there are two modes of the inspection process: *normal mode* and *emergency mode* with additional resources. We use $\Theta = \{normal, emergency\}$ to denote the two modes and the transitions between the two modes are as follows, also illustrated in Figure 3.2: if the inspector interdicts an illegal container at the normal mode at t, the emergency mode is triggered at $t + 1$ and this mode will continue for only one time step if there is no illegal container interdicted during this time step. Otherwise, the emergency mode will continue until there is no illegal container interdicted and transits to normal mode.

We use n_p to denote the capability of port $p \in \mathcal{P}$ which is the largest number of containers which can be inspected at port p in a time step. The ports' capabilities

[1] Generally, the time of transporting a container from the original port to the target port varies from one shipping line to another, from one week to a month. For the conveniences of the modeling and analysis, we take the average time of the transportation as the value of τ. We note that this can be easily relaxed to incorporate more realistic analysis.

[2] To maximize his utility, the smuggler will prefer to smuggler as much material as possible in an illegal without triggering the NII alarms. Therefore, we assume that all containers are including the maximum amount of nuclear material and the payoff of transporting a container to the target ports depends on the prices on the black market where the smuggler sells the nuclear material.

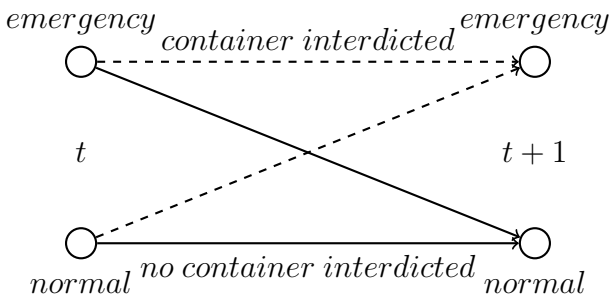


FIGURE 3.2: Mode transition.

are constant in both normal and emergency mode, as the inspector will fully take all the devices and facilities to inspect the containers. We denote the allocation of the inspection resource as $C : \Theta \times \mathcal{P} \times \mathcal{L} \to [0, 1]$ where C^{θ}_{pl} is the percentage of the containers on l being inspected at the port p at the mode θ, and $C^{\theta}_{pl} = 0$ if the line l does not visit the port p, i.e., $\alpha_{lp} = 0$. An allocation of the inspection resource C must satisfy the following constraints:

$$\sum_{l \in \mathcal{L}} C^{\theta}_{pl} f_l \leq n_p \quad \text{for all} \quad p \in \mathcal{P}, \theta \in \Theta. \tag{3.1}$$

The additional emergency inspection resource is denoted as n^e, i.e., the largest number of containers being inspected by the additional emergency resources in a time step. We use $\mathbf{c} : \mathcal{L} \to [0, 1]$ to denote the inspection strategy of using the emergency resources in the emergency mode where c_l denotes the percentage of the containers on the line l being inspected by the emergency inspection resources. The valid allocation $\mathbf{c}$ must satisfy:

$$\sum_{l \in \mathcal{L}} c_l f_l \leq n^e \tag{3.2}$$

We denote the percentage of containers being inspected by the inspector's strategy in different modes as $X : \Theta \times \mathcal{L} \to [0, 1]$:

$$X^{\theta}_{l} = \sum_{p \in \mathcal{P}} C^{\theta}_{pl} + c_l \mathbb{I}\{\theta = emergency\} \quad \text{for all} \quad l \in \mathcal{L}, \tag{3.3}$$

where X^{θ}_{l} denotes the percentage of containers on the shipping line l being inspected at mode θ and $\mathbb{I}\{\theta = emergency\}$ is the indicator function where if $\mathbb{I}\{\theta = emergency\} =$

1 if θ is emergency and 0 otherwise[1].

Smuggler's strategy: One one hand, the sequential plans are more beneficial to the smuggler to avoid the risk of being interdicted, compared with the plan of immediately transporting all illegal containers. On the other hand, the smuggler will want to transport all illegal containers as soon as possible because the longer the smuggling process lasts, the probability of detecting by the local security agencies is higher. Therefore, we use the discount factor γ to model this kind of the smuggler's behaviors, i.e., the payoff of successfully shipping an illegal container at the time step t is discounted by γ^t. We model the long-term smuggler's decision process as a Markov Decision Process (MDP) parameterized by a tuple $(\mathcal{S}, \mathcal{A}, T, R, \pi)$. $\mathcal{S}$ is the set of states of the MDP and each state $s \in \mathcal{S}$ is determined by $s = \langle t, \theta, \tilde{m} \rangle$ where t is the time step, θ is the inspector's mode at state s and $\tilde{m}$ is the number of remaining illegal containers to be smuggled. The process of the smuggling activities is started with $s_0 = \langle 0, normal, m \rangle$ and the terminal states are $s = \langle *, *, 0 \rangle$, i.e., all illegal containers are transported. The set of all terminal states is denoted as $\mathcal{S}^T \subset \mathcal{S}$. $\mathcal{A}$ is the set of actions of the smuggler. Specifically, $\mathbf{a} = \langle a_l \rangle$ is a valid allocation of the illegal containers to shipping lines where $a_l = 1$ means that there is a container transported through the shipping line l and $a_l = 0$ if there is no container transported on line l[2]. We assume that there is no interdependence between the shipping line, i.e., the probabilities of being interdicted by the inspector are independent for different shipping lines. We denote the set of all valid actions at the state $s = \langle t, \theta, \tilde{m} \rangle$ as $\mathcal{A}_s \subseteq \mathcal{A}$:

$$\mathcal{A}_s = \{\mathbf{a} \in \{0,1\}^{|\mathcal{L}|} : \sum\nolimits_{l \in \mathcal{L}} a_l \leq \tilde{m}\} \quad \forall s \in \mathcal{S}.$$

We note that $\mathcal{A} = \mathcal{A}_{s_0}$ and use $\mathbf{a}^s$ to represent the specific action at s when necessary.

When the smuggler takes the action $\mathbf{a} \in \mathcal{A}_s$ at the state $s = \langle t, \theta, \tilde{m} \rangle$, there are two states which can be probably reached: $s' = \langle t+1, normal, \tilde{m}' \rangle$ and $s' = \langle t +$

[1] As the reliable inspection resources are very limited and the inspection results can be sent to any ports, we assume that the container is inspected by the reliable inspection resources at most once when being transported through the shipping line. Even the inspection can be failed with some probability, the inspector would like to inspector the container which is never being inspected before.

[2] In the normal case, the shipping companies would pack the goods for customer at the container fright station. To avoid the case where two or more units of nuclear materials are packed into one container, the smuggler would only transport at most one unit through a shipping line at a time step.

$1, emergency, \tilde{m}'\rangle$ such that $\tilde{m}' = \tilde{m} - \sum_{l\in\mathcal{L}} a_l$. Given the inspector's strategy X, we use $T(s, \mathbf{a}, s')$ to represent the probability of reaching $s' = \langle t+1, \theta', \tilde{m}'\rangle$ from $s = \langle t, \theta, \tilde{m}\rangle$ by taking $\mathbf{a} \in \mathcal{A}_s$:

$$T(s, \mathbf{a}, s') = \begin{cases} 1 - \Phi(\theta, X, \mathbf{a}) & \theta' = emergency; \\ \Phi(\theta, X, \mathbf{a}) & \theta' = normal. \end{cases} \tag{3.4}$$

where $\Phi(\theta, X, \mathbf{a}) = \prod_{l\in\mathcal{L}}(1 - X_l^\theta)^{a_l}$ is the probability that there is no illegal container is interdicted by the inspector at mode θ. $R(s, \mathbf{a}, s')$ is the reward of the smuggler by taking action $\mathbf{a}$ at state s and finally reaching state s', noting that the discount factor is taken into account:

$$R(s, \mathbf{a}, s') = \begin{cases} \gamma^t \frac{\sum_{l\in\mathcal{L}}(1 - X_l^\theta - \Phi(\theta, X, \mathbf{a}))a_l u_l^a}{1-\Phi(\theta, X, \mathbf{a})} & \theta' = emergency; \\ \gamma^t \sum_{l\in\mathcal{L}} a_l u_l^a & \theta' = normal. \end{cases} \tag{3.5}$$

The reward for $\theta' = emergency$ is calculated by the fact that the expected utilities of taking action $\mathbf{a}$ at state s equal to $\gamma^t \cdot \sum_{l\in\mathcal{L}}[(1 - X_l^\theta) \cdot a_l u_l^a]$.

We denote the policy of the smuggler as $\pi : \mathcal{S} \to \mathcal{A}$ where $\pi(s)$ gives the action $\mathbf{a} \in \mathcal{A}_s$ taken by the smuggler at state s. We denote the value function of the policy as $V^\pi : \mathcal{S} \to \mathbb{R}$ where $V^\pi(s)$ represents the expected utility that when the current state is s and the smuggler follows policy π in subsequent states.

Utility and Equilibrium: Given the strategy profile of both the inspector and the smuggler $\langle X, \pi\rangle$, the expected utility of the smuggler $U_a(X, \pi)$ is defined as: $U_a(X, \pi) = V^\pi(s_0)$; Given that the game is zero-sum, the expected utility of the inspector is $U_d(X, \pi) = -U_a(X, \pi)$. We take the Stackelberg equilibrium, which is the standard solution concept in security game [2, 39, 93]. Formally, the Stackelberg equilibrium is the strategy profile $\langle \mathbf{X}^*, \pi^*\rangle$ such that:

1. $U_a(X^*, \pi^*) \geq U_a(X^*, \pi)$ for any other policy π,
2. $U_d(X^*, \pi^*) \geq U_d(X, \pi)$ for any other strategy X where π is the best response policy against X.

3.3 Solution Approach

In this section, we introduce the approach developed in this chapter to compute the Stackelberg equilibrium of the model efficiently. First, we propose a non-convex formulation of the problem based on the fact that the smuggler's optimal policy can be computed by linear program. Unfortunately, the number of constraints of the non-convex formulation grows exponentially with the size of the game and the nonconvexity of the problem makes the problem extremely difficult to solve. Therefore, we propose a linear relaxation approximation of transition probability to transform the multilinear terms in the program to bilinear terms with the solution quality guarantee and we further use the Multiparametric Disaggregation Technique to transform the bilinear terms into linear terms by introducing additional variables and constrains. Finally, we propose a novel iteration method to incrementally add the actions and states of the smuggler's MDP into consideration to further improve the scalability.

3.3.1 LP for Smuggler's MDP

For an inspector's inspection result X, the MDP of the smuggler can be solved by the linear program by adding the constraints of all states and actions [94]:

$$\min_{V} V(s_0) \tag{3.6a}$$

$$\text{s.t. } V(s) \geq \sum\nolimits_{s' \in \mathcal{S}} T(s, \mathbf{a}, s')[R(s, \mathbf{a}, s') + V(s')]$$

$$\forall \mathbf{a} \in \mathcal{A}_s, \forall s \in \mathcal{S} \setminus \mathcal{S}^T \tag{3.6b}$$

$$V(s) = 0, \quad \forall s \in \mathcal{S}^T \tag{3.6c}$$

We use V^* to denote the solution of Program (3.6) and then we can reconstruct the optimal policy of the smuggler π^* by:

$$\pi^*(s) = \arg\max\nolimits_{\mathbf{a} \in \mathcal{A}_s} Q(s, \mathbf{a}), \quad \forall s \in \mathcal{S} \setminus \mathcal{S}^T$$

where $Q(s, \mathbf{a}) = \sum_{s' \in \mathcal{S}} T(s, \mathbf{a}, s')[R(s, \mathbf{a}, s') + V^*(s')]$. We say $\mathbf{a} \in \pi^*(s)$ if $\pi^*(s)$ select the action $\mathbf{a}$ at the state s.

3.3.2 Nonconvex Program for Optimal Inspection

We can construct the inspector's results X by the inspector's strategy, i.e., $\langle C, \mathbf{c} \rangle$. Therefore, we can compute the inspector's optimal strategy through the following non-convex optimization formulation:

$$\min_{C,\mathbf{c},X,V} V(s_0) \tag{3.7a}$$

$$\text{s.t. Eqs.(3.1)–(3.3)} \tag{3.7b}$$

$$\text{Eqs.(3.6b)–(3.6c)} \tag{3.7c}$$

Eq. (3.7b) ensure the validity of the inspector's strategy and Eq. (3.7c) ensure the optimality of the value function V to minimize the objective $V(s_0)$.

Given the model, it seems that the MDP of the smuggler will last for infinite time step, which makes Program (3.7) impractical to solve. Fortunately, Lemma 3.1 states that the optimal policy π^* can terminate in finite time step.

Lemma 3.1. *The smuggler will transport all containers in $2m$ time steps.*

Proof. The proof is based on the fact that there is at least a container is smuggled at each time step under the normal mode. If there is no container transported under the normal mode, the operation mode of the inspector will stay in normal and the utility of the smuggler will be decreased due to the discount factor. Therefore, under the optimal policy, the smuggler transports at least a container in every 2 time steps, thus, the smuggling process at most lasts for $2m$ time steps. □

Although we can restrict the horizon of the smuggler's MDP into $2m$ timestep, the nonconvex Program (3.7) is still impossible to compute the exact optimal solution when the size becomes large, which are mainly due to the two kinds of terms in the problem: the multilinear terms $\Phi(\theta, X, \mathbf{a})$ and the term $T(s, \mathbf{a}, s') \cdot V(s')$ in Eq.(3.6b). To solve the program, we introduce a linear relaxation approximation of transition probability (3.4) and reward function (3.5) to transform the multilinear terms into bilinear terms.

3.3.3 Linear Relaxation of Φ

As we mentioned above, in fact, there are only a small percentage of containers being inspected at port due to the large volume of containers transported on the shipping lines. Therefore, we use the first order Taylor expansion of $\Phi(\theta, X, \mathbf{a})$ to approximate the real value of $\Phi(\theta, X, \mathbf{a})$ by noting that when the elements in X are far smaller than 1, this method can provide a good approximation:

$$\Phi(\theta, X, \mathbf{a}) = \prod\nolimits_{l\in\mathcal{L}}(1 - X_l^\theta)^{a_l} \approx 1 - \sum\nolimits_{l\in\mathcal{L}} a_l \cdot X_l^\theta$$

Applying the linear approximation to $\Phi(\theta, X, \mathbf{a})$ in the equation of the probability of the transitions (3.4) and the function of rewards (3.5), the smuggler's MDP is reformulated and we can transform Program (3.7) to compute the following formulation compute the equilibrium of the game:

$$\min_{C,\mathbf{c},X,V} V(s_0) \tag{3.8a}$$

$$\text{s.t. Eqs.(3.1)–(3.3)} \tag{3.8b}$$

$$V(s) \geq \gamma^t \sum\nolimits_{l\in\mathcal{L}}(1 - X_l^\theta)a_l u_l^a + (1 - \sum\nolimits_{l\in\mathcal{L}} a_l X_l^\theta)V(s^n) + \sum\nolimits_{l\in\mathcal{L}} a_l X_l^\theta V(s^e) \quad \forall \mathbf{a} \in \mathcal{A}_s, \forall s \in \mathcal{S} \setminus \mathcal{S}^T \tag{3.8c}$$

$$V(s) = 0, \quad \forall s \in \mathcal{S}^T \tag{3.8d}$$

where the state in Eq.(3.8c) is $s = \langle t, \theta, \tilde{m}\rangle$. We denote the two states reachable from taking action $\mathbf{a}$ in s as s^n and s^e where $s^n = \langle t+1, normal, \tilde{m}'\rangle$, $s^e = \langle t+1, emergency, \tilde{m}'\rangle$, and $\tilde{m}' = \tilde{m} - \sum_{l\in\mathcal{L}} a_l$. This reformulation technique brings the loss to the optimality. We state a bound between the utility computed by Program (3.8) and the exact optimal utility in Theorem 3.2.

Theorem 3.2. *For the given X, let π^* denote the optimal policy of the smuggler in Program (3.7) and π denote the optimal policy of the smuggler for Program (3.8). Let V^* and V be the value functions of π^* and π in the original MDP and the MDP applying*

the linear approximation, respectively. The following inequality holds:

$$V^*(s_0) - V(s_0) \leq m^2 \cdot \frac{\kappa^2 \cdot \gamma}{1-\gamma} \cdot \bar{V}$$

where $\bar{V} = m \cdot \max_{l \in \mathcal{L}} u_l^a$ *and* $\kappa = \max_{l \in \mathcal{L}, \theta \in \Theta} X_l^\theta$.

Proof. We note that $V^*(s_0) \geq V(s_0)$ always holds. Given the inspector's strategy X_l^θ, we have the following results ensured by the Taylor's theorem and the fact that $X_l^\theta << 1$:

$$\begin{aligned} &\textstyle\prod_{l \in \mathcal{L}} (1 - X_l^\theta)^{a_l} - (1 - \sum_{l \in \mathcal{L}} a_l \cdot X_l^\theta) \\ &\approx \sum_{l, l' \in \mathcal{L}, l \neq l'} a_l a_{l'} \cdot X_l^\theta X_{l'}^\theta \\ &\leq m^2 \cdot \kappa^2 \end{aligned}$$

We note that the third and higher order terms are ignored. To prove the theorem, we need to discuss the following two cases **C1** and **C2**.

C1: we consider the case that the smuggler has the same optimal policy, i.e., $\pi^* = \pi$. Suppose that the policy terminates at the time step T, that is to say, the smuggler transports all containers in T time steps. It is easy to verify that $V^*(s) \geq V(s)$ when the smuggler follows the same policy starting at the state s in both original MDP and the relaxed MDP because the probability in the original MDP of transiting to normal mode is higher than the corresponding probability in the relaxed MDP and the reward of both MDPs is equivalent. Therefore, we can bound the difference of the two values $V^*(s) - V(s)$ at each time step from $T-1$ by the induction method and finally give the bound of $V^*(s_0) - V'(s_0)$.

(1) We note that$V^*(s) = V(s)$ where $s = \langle T-1, \theta, \tilde{m} \rangle$ because the rewards of both MDPs are the same, as well as the optimal policies.

(2) Then, we assume that $0 \leq V^*(s) - V(s) \leq \varepsilon$ where $s = \langle t+1, \theta, \tilde{m} \rangle$, therefore, the following equations hold:

$$V^*(s) = \textstyle\prod_{l \in \mathcal{L}} (1 - X_l^\theta)^{a_l} \cdot V^*(s^n)$$

$$
\begin{aligned}
&+(1-\prod\nolimits_{l\in\mathcal{L}}(1-X_l^\theta)^{a_l})\cdot V^*(s^e)+\gamma^t\sum\nolimits_{l\in\mathcal{L}}(1-X_l^\theta)a_l u_l^a\\
V(s)=&(1-\sum\nolimits_{l\in\mathcal{L}}a_l\cdot X_l^\theta)\cdot V(s^n)\\
&+(\sum\nolimits_{l\in\mathcal{L}}a_l\cdot X_l^\theta)\cdot V(s^e)+\gamma^t\sum\nolimits_{l\in\mathcal{L}}(1-X_l^\theta)a_l u_l^a
\end{aligned}
$$

where $s=\langle t,\theta,\tilde{m}\rangle$, $s^n=\langle t+1,normal,\tilde{m}'\rangle$, $s^e=\langle t+1,emergency,\tilde{m}'\rangle$ and $\tilde{m}'=\tilde{m}-\sum_{l\in\mathcal{L}}a_l$. Then, we can calculate the difference as follows:

$$
\begin{aligned}
V^*(s)-V(s)=&(\prod\nolimits_{l\in\mathcal{L}}(1-X_l^\theta)^{a_l})\cdot(V^*(s^n)-V(s^n))\\
&+(1-\prod\nolimits_{l\in\mathcal{L}}(1-X_l^\theta)^{a_l}))\cdot(V^*(s^e)-V(s^e)))\\
&+\delta^\theta(V(s^n)-V(s^e))\\
\leq&\varepsilon+\delta^\theta(\gamma^{t+1}\bar{V})\\
\leq&\varepsilon+m^2\cdot\kappa^2\cdot\gamma^{t+1}\bar{V}
\end{aligned}
$$

where $\delta^\theta=\prod_{l\in\mathcal{L}}(1-X_l^\theta)^{a_l}-(1-\sum_{l\in\mathcal{L}}a_l\cdot X_l^\theta)$.

Therefore, the following result is obtained:

$$
\begin{aligned}
&V^*(s_0)-V'(s_0)\\
&\leq m^2\cdot\kappa^2\sum\nolimits_{t=1}^{T-1}\gamma^t\bar{V}\\
&\leq m^2\cdot\frac{\kappa^2\cdot\gamma}{1-\gamma}\bar{V}
\end{aligned}
\tag{3.9}
$$

C2: on the other hand, if the optimal policy of the smuggler in the relaxed MDP is different from the original MDP, we can always have: $V^*(\pi^*)-V(\pi)\leq V^*(\pi^*)-V(\pi^*)$ where $V^*(\pi)$ and $V(\pi)$ are the values of the smuggler if the smuggler follows the same policy π in both the original and the relaxed MDP, respectively. Thus, the result in Eq.(3.9) also holds, which closes this proof. □

Program (3.8) transforms the multilinear terms in Program (3.7) into bilinear terms and obtains a bilinear program, which is still non-convex. To further linearize the problem, we adopt the Multiparametric Disaggregation Technique (MDT).

3.3.4 Linearization Based on MDT

The basic idea of applying MDT to linearize the problem is to replace $X_l^\theta \cdot V(s)$ with a new variable $w_l^\theta(s)$ and then introduce a set of auxiliary variables and a set of linear constraints involving $w_l^\theta(s)$, X_l^θ and $V(s)$ to approximate the equation $w_l^\theta(s) = X_l^\theta \cdot V(s)$. Specifically, since $X_l^\theta \in [0,1]$ is bounded[1], we approximate X_l^θ with a number represented with Z digits located at the powers $\{-Z, ..., -1\}$. For each power $-z$, we define a binary variable λ_{lz}^θ and therefore, X_l^θ can be represented as follows:

$$X_l^Z = \sum\nolimits_{z=1}^{Z} 2^{-z}\lambda_{lz}^\theta + \tilde{X}_l^\theta, \tag{3.10}$$

where $\tilde{X}_l^\theta \in [0, 2^{-Z}]$ is the slack variable. Since $w_l^\theta(s) = X_l^\theta \cdot V(s)$, we have:

$$w_l^\theta(s) = \sum\nolimits_{z=1}^{Z} 2^{-z}\eta_{lz}^\theta(s) + \tilde{w}_l^\theta(s), \tag{3.11}$$

where $\eta_{lz}^\theta(s) = \lambda_{lz}^\theta \cdot V(s)$ with the following constraints and a large constant M:

$$\begin{aligned} 0 \leq &\eta_{lz}^\theta(s) \leq V(s) \\ V(s) - (1-\lambda_{lh}^\theta)M \leq &\eta_{lz}^\theta(s) \leq \lambda_{lz}^\theta M \end{aligned} \tag{3.12}$$

However, the equal relation of the slack variables $\tilde{w}_l^\theta(s) = \tilde{X}_l^\theta \cdot V(s)$ cannot be exactly represented with linear constraints. Therefore, We leverage the McCormick Envelope [95] to approximate the relationship with following linear constraints:

$$\begin{aligned} 0 \leq &\tilde{w}_l^\theta(s) \leq 2^{-Z}V(s) \\ 2^{-Z}V(s) + \bar{V}(\tilde{X}_l^\theta - 2^{-Z}) \leq &\tilde{w}_l^\theta(s) \leq \bar{V}\tilde{X}_l^\theta \end{aligned} \tag{3.13}$$

where $\bar{V}$ is the upper bound of $V(s)$ which can be roughly estimated as the maximal possible utility $m \cdot \max_{l\in\mathcal{L}} u_l^a$. To summarize, MDT applies the linear system (3.10)–(3.13) to approximate the bilinear terms $X_l^\theta \cdot V(s)$ with $w_l^\theta(s)$ in Program (3.8) and

[1] In practice, we can have a more tighter upper bound by taking the capability of each port into consideration.

obtain the following mixed integer linear program (MILP):

$$\min_{\substack{C,\mathbf{c},X,V\\ \lambda,\eta,\tilde{X},w}} \quad V(s_0) \tag{3.14a}$$

$$\text{s.t. Eqs.(3.1)–(3.3)} \tag{3.14b}$$

$$V(s) \geq \gamma^t \sum\nolimits_{l\in\mathcal{L}}(1-X_l^\theta)a_l u_l^a + V(s^n) - \sum\nolimits_{l\in\mathcal{L}} a_l w_l^\theta(s^n) + \sum_{l\in\mathcal{L}} a_l w_l^\theta(s^e) \quad \forall \mathbf{a}\in\mathcal{A}_s, \forall s\in\mathcal{S}\setminus\mathcal{S}^T \tag{3.14c}$$

$$V(s) = 0, \quad \forall s\in\mathcal{S}^T \tag{3.14d}$$

$$\text{Eqs.(3.10)–(3.13).} \tag{3.14e}$$

We note that $w_l^\theta(s)$ is not strictly equal to $X_l^\theta \cdot V(s)$ because the slack variable can take any value in $[0, 2^{-Z}V(s)]$. In fact, the relation between the values of $w_l^\theta(s)$ and $X_l^\theta \cdot V(s)$ is:

$$|w_l^\theta(s) - X_l^\theta \cdot \tilde{V}(s)| \leq \frac{\tilde{V}(s)}{2^Z}. \tag{3.15}$$

Hence, as we aim to minimize the objective of Program (3.8), the solution $\tilde{V}$ returned by MILP (3.14) is a lower bound of the optimal value function V of Program (3.8). Furthermore, when the value of Z increases, $w_l^\theta(s)$ takes a closer value to $X_l^\theta \cdot V(s)$ and $\tilde{V}$ will give a better approximation to the real optimal value function V. On the other hand, for a given inspector's strategy X, the optimal value function os Program (3.6) $\hat{V}$ is an upper bound of V. Therefore, we develop the algorithm depicted in Algorithm 2, **B**inary-**B**ased MDT for **C**IM (BBC), where the digits' number Z in the MDT is increased until the difference of the upper bound and the lower bound is smaller than some threshold. Theorem 3.3 analyzes the relationship between the gap $V(s_0) - \tilde{V}(s_0)$ and the number of digits Z.

Theorem 3.3. *The bounds obtained by Algorithm 2 satisfy the following inequality:*

$$V(s_0) - \tilde{V}(s_0) \leq \frac{|\mathcal{L}|}{2^{Z-1}} \cdot \frac{\gamma(1-(2\gamma)^{2m})}{1-2\gamma}\bar{V}$$

where $\bar{V} = m \cdot \max_{l\in\mathcal{L}} u_l^a$.

Proof. This proof use the bound between $w_l^\theta(s)$ and $X_l^\theta \cdot \tilde{V}(s)$ shown in Eq.(3.15).

We note that $V(s_0) \geq \tilde{V}(s_0)$ always holds and $V(s) = \tilde{V}(s)$ for any terminal state $s \in \mathcal{S}^T$. For the non-terminal states, we assume $0 < V(s) - \tilde{V}(s) \leq \varepsilon$ where $s = \langle t+1, \theta, \tilde{m} \rangle$. For the state $s = \langle t, \theta, \tilde{m} \rangle$, the constraints in Program (3.8) and Program (3.14) related to $V(s_0)$ and $\tilde{V}(s_0)$ are as follows:

$$\begin{aligned} V(s) \geq \gamma^t \sum\nolimits_{l \in \mathcal{L}} (1 - X_l^\theta) a_l u_l^a + (1 - \sum\nolimits_{l \in \mathcal{L}} a_l X_l^\theta) V(s^n) \\ + \sum\nolimits_{l \in \mathcal{L}} a_l X_l^\theta V(s^e) \end{aligned} \tag{3.16}$$

$$\begin{aligned} \tilde{V}(s) \geq \gamma^t \sum\nolimits_{l \in \mathcal{L}} (1 - X_l^\theta) a_l u_l^a + \tilde{V}(s^n) \\ - \sum\nolimits_{l \in \mathcal{L}} a_l w_l^\theta(s^n) + \sum\nolimits_{l \in \mathcal{L}} a_l w_l^\theta(s^e) \end{aligned} \tag{3.17}$$

where $s^n = \langle t+1, normal, \tilde{m}' \rangle$, $s^e = \langle t+1, emergency, \tilde{m}' \rangle$ and $\tilde{m}' = \tilde{m} - \sum_{l \in \mathcal{L}} a_l$. The difference between the right hand of Eq.(3.16) and Eq.(3.17) is upper bounded by:

$$\begin{aligned} & \varepsilon - \sum\nolimits_{l \in \mathcal{L}} a_l (X_l^\theta V(s^n) - w_l^\theta(s^n)) + \sum\nolimits_{l \in \mathcal{L}} a_l (X_l^\theta V(s^e) - w_l^\theta(s^e)) \\ & \leq \varepsilon + \sum\nolimits_{l \in \mathcal{L}} a_l (\frac{1}{2^Z} (\tilde{V}(s^n) + \tilde{V}(s^e))) \\ & \quad + \sum\nolimits_{l \in \mathcal{L}} a_l \cdot X_l^\theta ((V(s^e) - \tilde{V}(s^e)) - (V(s^n) - \tilde{V}(s^n))) \\ & \leq 2\varepsilon + \frac{|\mathcal{L}|}{2^{Z-1}} \gamma^{t+1} \bar{V}. \end{aligned} \tag{3.18}$$

We note that $\sum_{l \in \mathcal{L}} a_l \cdot X_l^\theta \leq 1$ holds naturally due to the linear approximation because the transition probability is always positive. As we are aiming to minimize the objective, the increment of the value from $\tilde{V}(s)$ and $V(s)$ is also upper bounded by:

$$V(s) - \tilde{V}(s) \leq 2\varepsilon + \frac{|\mathcal{L}|}{2^{Z-1}} \gamma^{t+1} \bar{V}.$$

Therefore, we can obtain the following inequalities about the two bounds computed by Algorithm 2:

$$\begin{aligned} V(s) - \tilde{V}(s) & \leq \frac{|\mathcal{L}|}{2^{Z-1}} \cdot \sum_{t=1}^{2m} 2^{t-1} \gamma^t \bar{V} \\ & \leq \frac{|\mathcal{L}|}{2^{Z-1}} \cdot \frac{\gamma(1 - (2\gamma)^{2m})}{1 - 2\gamma} \bar{V}. \end{aligned} \tag{3.19}$$

Specifically, when $\gamma < 0.5$, Eq.(3.19) is turned to be:

$$V(s) - \tilde{V}(s) \leq \frac{|\mathcal{L}|}{2^{Z-1}} \cdot \frac{\gamma}{1-2\gamma}\bar{V}.$$

□

Algorithm 2: Binary-**B**ased MDT for **C**IM (BBC)

1 Initialize the values of parameters H, ϵ;
2 **repeat**
3 $\quad \langle X, \tilde{V} \rangle \leftarrow$ solution of Program (3.14);
4 $\quad \hat{V} \leftarrow$ solution of Program (3.6), given the inspector's strategy X;
5 $\quad Z = Z + 1$;
6 **until** $(\hat{V}(s_0) - \tilde{V}(s_0))/\hat{V}(s_0) < \epsilon$;
7 **return** $\langle X, \hat{V} \rangle$;

3.3.5 Improving the Scalability

There are too many too many auxiliary (binary) variables and constraints introduced in Program (3.14) to transform the problem into a MILP, which causes the scalability issues to Algorithm 2. Therefore, we propose the **S**tate and **A**ction **G**eneration for **BBC** (SAG-BBC) algorithm, depicted in Algorithm 3 to iteratively enlarge the size of the programs solved in the algorithm, which is based on the observations that by transporting all containers as fast as possible, the smuggler can obtain the highest utility due to the discount factor and there are many redundant actions for each state which are never selected by the smuggler.

The basic idea of Algorithm 3 can be described as follows: instead of considering the problem with $2m$ horizon, a restricted problem with a much smaller horizon h is solved (i.e., $\mathcal{S}_h = \{s | s = \langle t, \theta, m \rangle \in \mathcal{S}, t \leq h\}$). To obtain the optimal solution of the restricted problem $\langle X_h, V_h \rangle$, the algorithm first uses Algorithm 2 to compute the optimal solution $\langle X'_h, V'_h \rangle$ for a more restricted problem where only a subset of the actions $\mathcal{A}'_s \subset \mathcal{A}_s$ in the state $s \in \mathcal{S}_h$ are considered (i.e., Line 6). Then, the algorithm uses Program (3.6) to compute the smuggler's optimal policy $\hat{\pi}'_h$ against X'_h in the restricted problem with all actions $\mathcal{A}_s$ for $s \in \mathcal{S}_h$ for each state $s \in \mathcal{S}_h$ (i.e.,

Line 7). If there are actions which are selected by $\hat{\pi}'_h$ but not in $\mathcal{A}'_s$, we add them into $\mathcal{A}'_s$ and repeat to solve the restricted problem with action set $\mathcal{A}'_s$. Otherwise, the optimal solution $\langle X'_h, V'_h \rangle$ is also the optimal solution to the restricted problem with all actions $\mathcal{A}_s$ available, i.e., $\langle X'_h, V'_h \rangle = \langle X_h, V_h \rangle$. Given the optimal inspector's strategy of the restricted problem X_h , the algorithm calls Program (3.6) to compute the smuggler's optimal value $\hat{V}_h$ in the original MDP with horizon $2m$ and action set $\mathcal{A}$ (i.e., Line 14). If the value of initial state $V_h(s_0)$ of the restricted problem is equal to $V_h(s_0)$, the optimal solution is obtained and the algorithm is terminated; Otherwise, the algorithm will increase the horizon h of the restricted problem with a fixed number of time steps (e.g., 1) and repeat to solve the restricted problem. Theorem 3.4 ensures the algorithm reaches the global optimal solution at the termination.

Theorem 3.4. *Algorithm 3 can compute the optimal solution for Program (3.14).*

Proof. We first discuss the solution of the inner loop and then extend to the outer loop of Algorithm 3.

Inner loop: if all actions executed by the optimal policy of the smuggler $\hat{\pi}'_h$ belong to $\mathcal{A}'_s$, $\hat{\pi}'_h$ is considered in the constraints of Program 3.6. Therefore, the optimal policy constructed from V'_h gives the same utility to the smugger as $\hat{\pi}'_h$. Therefore, $\langle X'_h, V'_h \rangle$ is the optimal solution for the restricted problem.

Outer loop: We note that the inner loop can compute the optimal solution of the problem with the restricted horizon. Line 14 of Algorithm 3 computes the real smuggler's optimal value V against the inspector X_h in the original MDP. If the value of $V(s_0)$ at initial state equals to $V_h(s_0)$, as $V(s_0)$ is the optimal utility of the smuggler when the inspector plays X_h, which indicates that the optimal solution of the restricted problem $\langle X_h, V_h \rangle$ is also optimal to the original MDP. Thus, Algorithm 3 reaches the global optimal solution when terminated. □

Algorithm 3: **S**tate and **A**ction **G**eneration for **BBC** (SAG-BBC)

Initialize the value of the parameter h;
repeat
  Build the state set of the restricted problem $\mathcal{S}_h = \{s | s = \langle t, \theta, m \rangle \in \mathcal{S}, t \leq h\}$;
  Choose the actions $\mathbf{a}^s \in \mathcal{A}_s$ arbitrarily to build $\mathcal{A}'_s \subset \mathcal{A}_s, \forall s \in \mathcal{S}_h$;
  while *true* **do**
    $\langle X'_h, V'_h \rangle \leftarrow$ solution of Algorithm 2 by substituting $\mathcal{S}$ and $\mathcal{A}_s, \forall s \in \mathcal{S}$ with $\mathcal{S}_h$ and $\mathcal{A}'_s, \forall s \in \mathcal{S}_h$, respectively;
    $\hat{V}'_h \leftarrow$ solution of Program (3.6) by substituting $\mathcal{S}$ with $\mathcal{S}_h$, given X'_h;
    Find optimal policies of the smuggler $\hat{\pi}'_h$ for the restricted MDP, given $\hat{V}'_h$;
    if $\exists s \in \mathcal{S}_s$ *such that* $\pi'_h(s) \notin \mathcal{A}'_s$ **then**
      $\mathcal{A}'_s = \mathcal{A}'_s \cup \{\pi'_h(s)\}$;
    else
      $\langle X_h, V_h \rangle \leftarrow \langle X'_h, V'_h \rangle$
      break;
  $V \leftarrow$ solution of Program (3.6), given $X(h)$;
  $h = h + 1$;
until *the value of* $V_h(s_0)$ *at initial state equals* $V(s_0)$;
return $\langle X_h, V_h \rangle$;

3.4 Experimental Evaluation

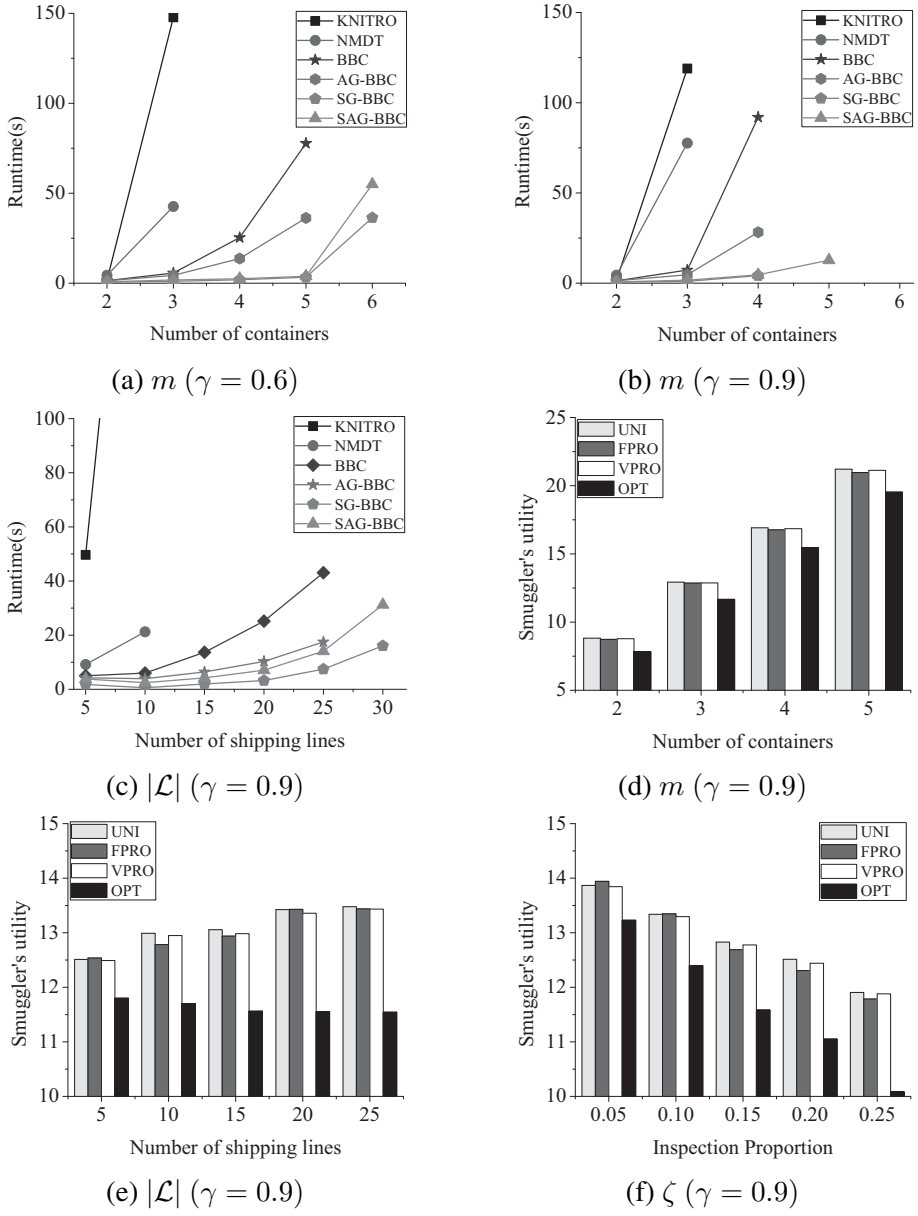
We evaluate our approaches through extensive experiments in this section. The mixed integer linear programs are solved by CPLEX (version 12.6) and nonlinear programs are solved by KNITRO (version 9.0.0). All experiments were conducted on a 64-bit PC with 16.0G RAM and a 3.50 GHz processor. All data points are averaged over 30 instances unless otherwise specified. The synthetic shipping networks are generated by the following method: each port is visited by each shipping line with a fixed probability. The flow f_l and the payoff u_l^a of the shipping line are sampled from the uniform distributions between $[0, 5]$ and $[4, 5]$, respectively. We denote the total number of inspection resources as $\zeta \cdot \sum_{l \in \mathcal{L}} f_l$, among which $0.01 \cdot \sum_{l \in \mathcal{L}} f_l$ are emergency inspection resources and other resources are randomly assigned to the ports. We note that ζ is the proportion of containers inspected by the inspector over all shipping lines at the emergency mode, which ranges in $[0.05, 0.25]$. The value of ϵ is 0.001 and the default values of $\langle |\mathcal{L}|, |\mathcal{P}|, m, \gamma, \zeta \rangle$ are $\langle 10, 10, 3, 0.9, 0.15 \rangle$.

We compare the scalability of four variants of our algorithms: i) BBC displayed in Algorithm 2; ii) AG-BBC: $T = 2m$ for Algorithm 3; iii) SG-BBC: $\mathcal{A}'_s = \mathcal{A}_s, \forall s \in \mathcal{S}_h$ for Algorithm 3; iv) SAG-BBC: Algorithm 3. All versions return the same solution, which is denoted by OPT. The benchmark methods are: i) KNITRO which is widely used for global optimization problems and ii) Normalized MDT (NMDT) [96] to solve bilinear programs.

We compare the solution quality of our solution with three baselines: i) UNI where the inspector uniformly allocates inspection resources to shipping lines, as well as for the emergency resources; ii) FPRO where the inspector allocates the resources to line l proportionally to the container flow f_l; iii) VPRO where the inspector allocates the inspection resources to line l proportionally to the payoff value of the smuggler u_l^a.

Scalability analysis. We compare the scalability of six variants of the algorithms on synthetic shipping networks. The results are displayed in Figures 3.3a-3.3c. We vary the value of m under different values of γ, i.e., $\{0.6, 0.9\}$. The results show that our approaches significantly outperform KNITRO. The results also indicate that SG-BBC and NMDT perform better under small γ, while SAG-BBC has a better performance when γ is larger. This is because when γ is larger, the smuggler prefers a long-term policy, i.e., the algorithm needs to take more time steps into consideration before the termination. SG-BBC adds all states and actions, while SAG-BBC incrementally adds actions of each state to reduce the number of constraints added into the program, thus making the program more scalable. We also vary the number of shipping lines and Figure 3.3c displays the results. It can be observed that the number of containers has much more influence on the scalability than the number of shipping lines because both the time horizon and the number of actions in each state depends on the number of containers.

Solution quality. Regarding the solution quality, we compare our algorithms with three baselines by varying the values of m, $|\mathcal{L}|$ and ζ. Figure 3.3d-3.3f displays the results, which demonstrates that our solution (i.e., OPT) outperforms the baselines. Besides, when the number of containers, the number of shipping lines and the proportion of containers being inspected increases, the advantage of our solution increases against



(a) m ($\gamma = 0.6$) (b) m ($\gamma = 0.9$)

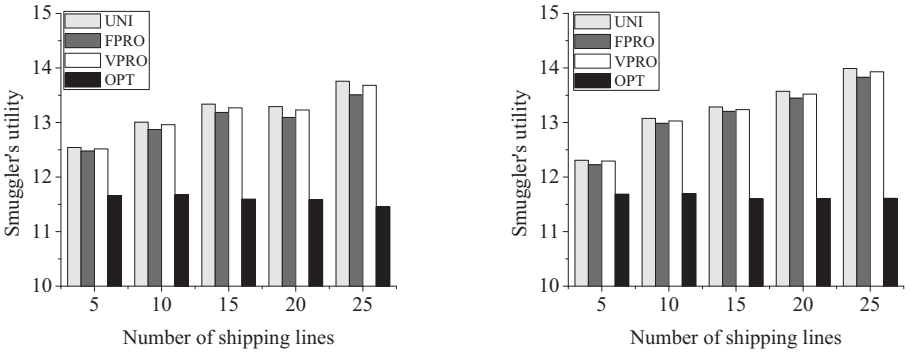
(c) $|\mathcal{L}|$ ($\gamma = 0.9$) (d) m ($\gamma = 0.9$)

(e) $|\mathcal{L}|$ ($\gamma = 0.9$) (f) ζ ($\gamma = 0.9$)

FIGURE 3.3: Scalability and Solution quality.

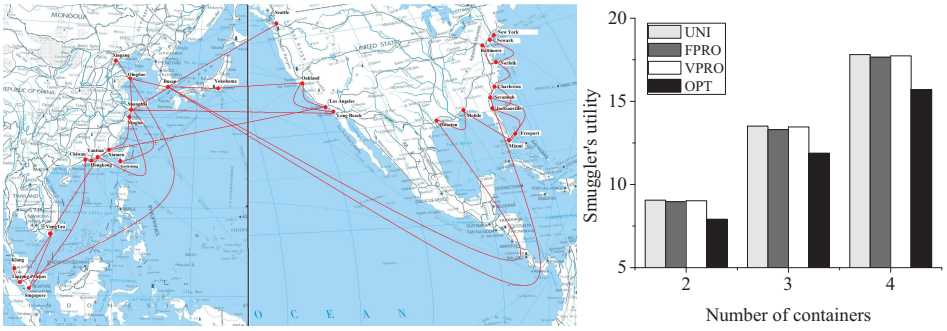
the baselines, which implies that the strategic allocation of inspection resources is effective to combat the smuggling activities.

Robustness. In the real world, the inspector cannot learn the value of the discount factor γ and the value of the payoff for each shipping line u_l^a exactly. Therefore, to test the robustness of our algorithm, we assume that the real value of $\hat{\gamma}$ would fall into the range $[\gamma-\delta, \gamma+\delta]$ where γ is the value taken by the inspector to determine the inspection strategy and for the reasonability, we assume that $\delta < \min\{\gamma, 1-\gamma\}$. Figure 3.4a shows the smuggler's utility where our solution still outperforms the baseline with $\delta = 0.1$ and



(a) γ ($\delta = 0.1$) (b) $u_l^a(\rho = 10\%)$

FIGURE 3.4: Robustness.



(a) Real shipping network. (b) Smuggler's utility.

FIGURE 3.5: Application on a real shipping network.

$\gamma = 0.9$ under the uncertainties. Analogously, we assume the real value of the payoff for the shipping line $\hat{u}_l^a$ can fall in the range $u_l^a \cdot [1-\rho, 1+\rho]$ where u_l^a is the value taken by the inspector to determine the inspection strategy and $0 < \rho < 1$. The results showed in Figure 3.4b demonstrate that our solution still outperforms the baselines with the value $\rho = 10\%$ and the default setting of three containers.

Application on a real shipping network. We also evaluate our algorithms on a real shipping network which includes 23 shipping lines and 32 ports of the three largest container shipping companies[1] between Asia and North America, as displayed in Figure 3.1a. The utility of the smuggler is displayed in Figure 3.5b, which indicates that our solution still outperforms the baselines for the real shipping network considered and when the number of containers is large, our solution is even better.

[1] https://www.msc.com/sgp, http://www.cma-cgm.com/, https://www.maerskline.com/

3.5 Chapter Summary

This chapter studies the problem of preventing nuclear smuggling on container shipping network through efficient container inspection. We introduce a novel container inspection model (CIM) based on Stackelberg game and propose efficient algorithms to compute the solutions, including a linear relaxation approximation which reformulates the problem into a bilinear optimization problem with the guarantee of the solution quality, a novel approach based on MDT to obtain the solution of the bilinear program and a state and action generation method, which incrementally enlarges the problem solved to further improve the scalability. Extensive experiments show that our algorithms outperform existing methods in the scalability significantly and can obtain a solution better than baselines, even under various uncertainties.

Chapter 4

Preventing Oil Siphoning in International Waters through Efficient Patrols

It was around midnight on the 4th of June 2015 when a group of pirates boarded the Malaysia-registered oil tanker Orkim Victory from a speed boat while under way to Kuantan port, Malaysia. After taking control of the ship, the pirates moved the Orkrim Victory to a different location where a second tanker pulled up alongside it. Within a couple of hours the pirates siphoned 770 metric tons of Automotive Diesel Oil and disappeared. This was the eighth of eleven piracy incidents occurring in the South China Sea (SCS) in 2015 [97, 98]. To combat the threat of piracy in the SCS, Singapore, Malaysia and Indonesia plan to establish patrols in the region [29]. Given the vast area of the SCS, the huge number of merchant ships passing through and the limited number of patrol boats, the question of how to efficiently deploy patrol resources is extremely challenging to the security agencies.

However, the methods in previous works from Stackelberg security game cannot be directly applied to our problem due to three main issues: i) our game is highly time and space dependent and both players take paths as their strategies while most previous works assume that at most one player takes paths [59, 99] or the values of targets are

static over time [100]; ii) continuous externalities, e.g., alarm influence, are incorporated to make our model more realistic, which are ignored [53] or only considered for protecting targets [9, 101]; iii) our game, with a reasonable discretion of time and space, is extremely large-scale and cannot be solved by the existing methods [41, 53]. We also note another line of research relate to our problem [102–104] where the attacker attacks the ships whose schedules are fixed and known to both players. However, according to the incident reports, the attacker may damage the communication system and move the ship to another place, which deviates from the schedule.

In this chapter, we address the above challenges and provide four key contributions. First, we construct a Stackelberg Model of the Oil-Siphoning problem (SMOS), where both players take time-dependent paths on the grid as their strategies, based on the incident reports from actual attacks as well as special reports conducted by maritime authorities to make the model realistic and computable; Second, in order to compute the efficient defender's patrol strategy, we propose a compact formulation and a constraint generation (CG) algorithm, which avoids the exponentially increasing number of pure strategies of the defender and the attacker, respectively; Third, to further improve the scalability, we propose an abstraction method to solve the extremely large-scale problem, which exploits intrinsic properties of the defender's strategy space to reduce the size of the game and makes a tradeoff between scalability and optimality; Finally, we evaluate our approaches through extensive simulations and a detailed case study with real ship traffic data. The results demonstrate that our approaches can scale up to realistic-sized problems with modest influence on the solution quality, significantly outperforming existing methods, and provide robust solutions against the uncertainties in the real world.

4.1 Motivation Scenario

In this section, we analyze the situation in the south of the SCS shown in Figure 4.1, which is a current hotspot of siphoning attacks. According to the ReCAAP information sharing center (ISC), there were 99 piracy attacks in the SCS from 2007 to 2017. As the considered region is too large to be considered as a continuous model, we discretize the

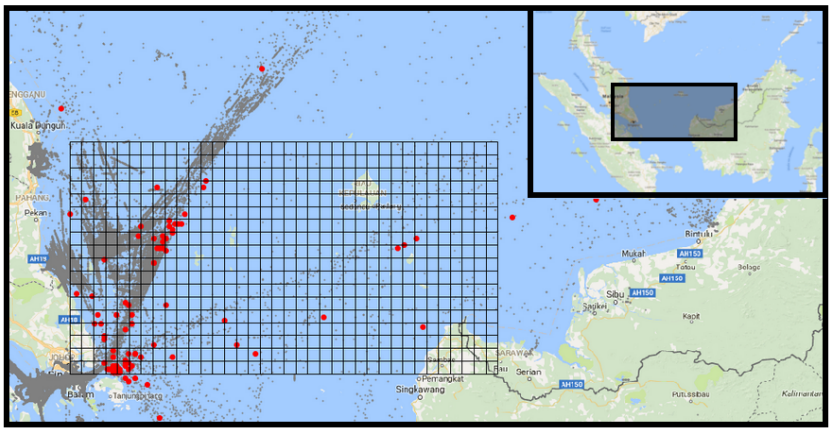


FIGURE 4.1: A map of the south region of SCS showing the ship locations during one week (grey) and pirate attacks between 2007 and 2016 (red dots). The rectangular area is considered in the case study in the evaluation section.

region into a grid of zones and assign the value of each zone according to the ship traffic data obtained from the automatic identification system (AIS). Besides, we discretize the time into equal steps, where the patrol boats use a number of steps from one node to its neighbors. Further, the vast majority of the attacks occurred during night time, which means that there is an ending time when the attackers have to finish their attack and that the patrol boats can relocate during the day, so there are no restrictions on the starting and ending zones of patrols and finite time steps are considered.

Note that siphoning can be done only in specific zones referred to as siphoning locations which can be determined via geographical features, usually close to the coast, or through ship density data. We assume that both the security agencies and the pirates know all possible siphoning locations. Based on the ReCAAP ISC reports [98, 105], the model of the attack process includes: i) capturing the ship, including approaching, boarding the ship, controlling the crew and damaging the communication system, etc, which takes a certain amount of time; ii) steering the ship to the siphoning location through a path on the grid; and iii) siphoning the oil to another ship, which also takes a certain amount of time and during which the attacker cannot move. We assume that both capturing time and siphoning time are the same for all zones, respectively.

By patrolling through the zones, the patrol boats offer two kinds of protection. The first is that they might discover pirates during any step of the attack, which is mainly in the zones they patrol in, but also a weaker extent in neighboring zones, due to the

noticed anomalies on AIS or radar systems installed on the boats. The second form of protection is due to the fact that patrolling ships can quickly react to alarms sent out by attacked ships before communication systems are damaged.

4.2 Stackelberg Model of Oil Siphoning

We propose a **S**tackelberg **M**odel of the **O**il-**S**iphoning problem (SMOS) where the defender commits to a mixed patrolling strategy, then, after careful observation, the attacker performs an optimal response [2]. We start by dividing the maritime territory into a set of zones $\mathcal{Z}$ which form a grid M. For a zone $i \in \mathcal{Z}$ we denote the neighbors of i on M by $\mathcal{N}(i)$[1]. Furthermore, we discretize the night into a sequence of time points $\mathbf{t} = \langle t_1, \ldots, t_\tau \rangle$, the consecutive time points are equidistant. Players can only act at these time points and every action, e.g., moving from one zone to a neighboring zone, is assumed to take one time step. The underlying grid together with the time line form a transition graph $G = (\mathcal{V}, \mathcal{E})$ on which the players can act. Every vertex $v = (i, t_k)$ consists of a zone $i \in \mathcal{Z}$ and a time step t_k. There is an edge e between two vertices $v = (i, t_k)$, $v' = (i', t_{k'})$ when $k' = k + 1$ and $i' \in \mathcal{N}(i)$.

Defender strategies. In our model the defender has m resources (i.e., patrol boats). Let $\mathcal{S} = \{v = (i, t_1)\}$ be the set of vertices corresponding to all possible zones at the first time step and $\mathcal{T} = \{v = (i, t_\tau)\}$ all possible zones at the last time step. Therefore, a patrol strategy P_r of a resource r is a path from a node in $\mathcal{S}$ to a node in $\mathcal{T}$. A pure strategy of the defender is then, consequently, given by a m-dimensional vector of patrol paths, i.e., $P = \langle P_1, \ldots, P_m \rangle$. A mixed strategy of the defender can be represented by a distribution of the possible pure strategies, i.e., $\mathbf{x} = \langle x_P \rangle$ where x_P is the probability of P being used. We have $\sum_P x_P = 1, \forall\, \mathbf{x}$.

Attacker strategies. Let $\mathcal{O} \subseteq \mathcal{Z}$ be the set of possible oil siphoning locations and let a and b denote the number of time steps it takes to capture a ship and to siphon the oil, respectively. We define an attacker's strategy as a path $F = ((i_1, t_h), \ldots, (i_l, t_{h+l}))$ on G where the first a time steps have to be spent on the same zone i, i.e., $i_1 = i_2 =$

[1] For convenience we define $i \in \mathcal{N}(i)$.

$\cdots = i_a$ and the last b time steps have to be spent on one of the oil siphoning zones, i.e., $i_{h+l} = i_{h+l-1} = \cdots = i_{h+l-b} \in \mathcal{O}$ and t_h is the time the attacker starts his attack. Since attacks are costly and occur with relative low frequencies, we assume there is an attacker and only pure strategies are considered, which is in line with previous works in security games [45].

Utilities and Stackelberg equilibrium. In our model each zone $i \in \mathcal{Z}$ has a certain value at each time step t_k denoted by $u(i, t_k)$ which is the value gained by the attacker when attacking a ship at zone i at time t_k. If the attacker successfully launches an attack F in zone i at t_k, he gets $U(F) = u(i, t_k)$ and the defender gains $-U(F)$. If the defender detects the attacker, both players get 0. Therefore, our game is zero-sum, as most related works in security domain.

A resource can detect the attacker via three ways:

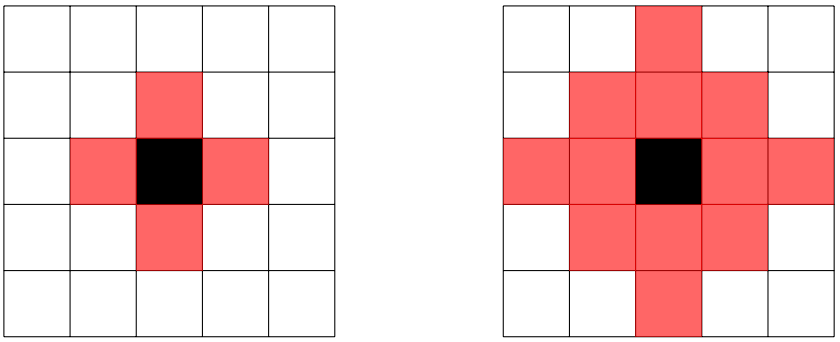
1. At a time step t_k, the attacker is detected with a probability $d \leq 1$ if they are inside the same zone;
2. At a time step t_k, the attacker is detected with a probability $e < d$ if they are in the neighboring zones;
3. At the starting time of the attack t_h, the attacker is detected if an alarm is raised by the ship being attacked and the resource is *close enough* to catch the attacker. The probability of raising an alarm is α.

A resource is considered close enough if it can interdict the attacker's path starting from its position at time t_h. Figure 4.2 presents a simple example of the three kinds of detection. Let P_r be the patrol strategy of resource r and F the attacker's strategy,

$$\mathcal{B}_r = \{v | v \in P_r \cap F\}$$

be the set of vertices they have in common,

$$\mathcal{N}_r = \{(i, t_k) | (i, t_k) \in P_r, \exists (j, t_k) \in F \text{ s.t. } j \in \mathcal{N}(i), j \neq i\}$$



(a) (b)

FIGURE 4.2: (a) The red zones are neighboring zones of the black zone. (b) Suppose the attacker arrives at the black zone at $t_h + 2$, if the resource at any one of the red zones at t_h, the resource is *close enough* to catch the attacker.

be the set of nodes where the patrol and the attacker are in neighboring zones and

$$\mathcal{R}_r = \{(i, t_h) | (i, t_h) \in P_r, \exists (j, t_k) \in F \colon d(i, j) \leq t_k - t_h\}$$

be the set of zones from which the patrol could reach the attacker (at unit speed). Following the formulation adopted in [41, 53, 103], the probability of detecting the attacker is calculated as $\min(1, d|\mathcal{B}_r| + e|\mathcal{N}_r| + \alpha \mathbb{1}_{\mathcal{R}_r})$ where $\mathbb{1}_{\mathcal{R}}$ is the indicator function of set $\mathcal{R}$, indicating if resource r could reach the attacker's path if an alarm is raised.

Further, we assume that the independent work of the patrols leads to an addition of the individual probabilities. Hence, given a strategy F of the attacker and a strategy $P = \langle P_r \rangle$ of the defender, the probability to detect the attacker is given by

$$dpp(P, F) = \min \left(1, \sum\nolimits_{r=1}^{m} (d|\mathcal{B}_r| + e|\mathcal{N}_r| + \alpha \mathbb{1}_{\mathcal{R}_r})\right).$$

From this we can deduce the players' expected utilities. Given a pair of strategies (P, F) the attackers utility is $U^a(P, F) = (1 - dpp(P, F))U(F)$. When the defender is allowed to play a mixed strategy $\mathbf{x} = \langle x_P \rangle$, the attacker's expected utility is $U^a(\mathbf{x}, F) = \sum_P x_P U^a(P, F)$ by slight abuse of notation, the expected utility of the defender is defined as $U^d = -U^a$.

We want to compute the optimal strategy for the defender to commit to. Given the zero-sum setting, this is equivalent to maximizing the defender's utility under the assumption of best response of the attacker. This leads to the following optimization

problem. Let $\mathcal{X}$ and $\mathcal{F}$ be the strategy space for the defender and attacker, respectively.

$$\max_{\mathbf{x}\in\mathbf{X}} U^d(\mathbf{x}, F)$$
$$\text{s.t. } U^a(\mathbf{x}, F) \geq U^a(\mathbf{x}, F'), \forall F' \in \mathcal{F}$$

4.3 Constraint Generation

In this section, we introduce a compact formulation to the original problem to avoid the exponentially large number of defender's pure strategies; and then adopt the constraint generation method to incrementally add the attacker's strategy into consideration to speed up the algorithm.

4.3.1 Compact LP Formulation

The exponentially increasing number of pure strategies of both the defender and the attacker in the size of the game makes the computation of optimal solutions in the original formulation intractable. Many previous researches use compact representation as a remedy [41, 53, 67]. In our model, a compact representation can be given via a coverage vector $\mathbf{c} = \langle c_v | v \in \mathcal{V} \rangle$ which assigns the marginal coverage on the node v in the graph. Let $\mathbf{x}$ be a given mixed strategy of the defender, we have $c_v = \sum_P x_P P(v)$ where $P(v)$ represents the number of resources going through v in the pure strategy P. For any pure strategy F of the attacker, we can write the attacker's utility $U^a(\mathbf{c}, F)$ as $(1 - dpp(\mathbf{c}, F))U(F)$ where

$$dpp(\mathbf{c}, F) = \min(1, d\sum\nolimits_{v\in F} c_v + e\sum\nolimits_{v\in\mathcal{N}(F)} c_v + \alpha\sum\nolimits_{v\in\mathcal{R}(F)} c_v).$$

Now we can construct the linear program coreLP to compute the optimal coverage:

$$\max_{\mathbf{c}} U \tag{4.1a}$$
$$U \leq -U^a(\mathbf{c}, F), \forall F \in \mathcal{F} \tag{4.1b}$$
$$c_{(i,t_k)} = \sum\nolimits_{j\in\mathcal{N}(i)} f_{((i,t_k),(j,t_{k+1}))}, \forall i \in \mathcal{Z}, k \in \{1, \ldots, \tau - 1\} \tag{4.1c}$$

$$c_{(i,t_k)} = \sum\nolimits_{j \in \mathcal{N}(i)} f_{((j,t_{k-1}),(i,t_k))}, \forall i \in \mathcal{Z}, k \in \{2, \ldots, \tau\} \tag{4.1d}$$

$$m = \sum\nolimits_{i \in \mathcal{Z}} c_{(i,t_k)}, k \in \{1, \tau\}, \tag{4.1e}$$

where $f_{((i,t_k),(j,t_{k+1}))} \in \mathbb{R}^+$ are flow variables, so that Eqs.(4.1c) and (4.1d) ensure that the coverage vector can be obtained by a mixed strategy [41]. Eq.(4.1e) forces the number of patrol resources at the beginning and end of the game to be m and Eq.(4.1b) represents that the attacker uses his best-response to the defender's strategy. Given a coverage vector $\mathbf{c}$, we can easily obtain a mixed strategy by splitting the flow into a set of weighted simple paths, which leads to only a minor loss for the defender [53]. Notice that this formulation is still exponentially large in the size of the game since the number of pure strategies of the attacker is exponentially large. Therefore, we use Constraint Generation (CG) to incrementally add the attacker's pure strategy into consideration to improve the scalability.

4.3.2 CG for Compact Representation

In order to improve the scalability of coreLP, we apply Constraint Generation (CG) to the compact representation. Instead of starting with all of the exponentially many constraints, we start with only a small subset $\mathcal{F}'$. After we obtain a coverage vector for the subset $\mathcal{F}'$, we call the attacker oracle (AO). The AO computes, based on a given coverage vector $\mathbf{c}$, for every possible starting point (i, t_h) of an attack F the path which offers the attacker the best chances to escape. After considering every (i, t_h), we obtain the best response F^* of the attacker. If F^* is already in $\mathcal{F}'$ the algorithm terminates in an equilibrium, otherwise we add F^* into $\mathcal{F}'$ and restart the coreLP.

At the heart of the AO is Algorithm 4. For a given coverage $\mathbf{c}$ as well as the starting node (i_s, t_s), every step on the attacker's path brings a certain additional probability to be caught. Viewing these probabilities as weights on a graph, the goal of the attacker is to find a shortest path from (i_s, t_s) to a position where the attack is finished, i.e., reach a siphoning location and finish siphoning. This shortest path can be found with a modified Dijkstra's algorithm. Algorithm 4 starts by computing the actual catching probability $\mathbf{c}'$ based on direct patrolling and the externality effect (lines 1 and 2). The

Algorithm 4: findBestPath $(\mathbf{c}, i_s, t_s)$

1 **for** $(i, t_k) \in \mathcal{V}$ **do**
2 $\quad c'_{(i,t_k)} \leftarrow d \cdot c_{(i,t_k)} + e \sum_{j \in \mathcal{N}(i)} c_{(j,t_k)}$;
3 $Q \leftarrow$ `initializeQueue()`;
4 **while** $Q \neq \emptyset$ **do**
5 $\quad (p, i, t, \hat{b}, A) \leftarrow Q$.pop();
6 $\quad t++$;
7 $\quad$ **if** $\hat{b} = 0$ **then break else if** $\hat{b} < b$ **then**
8 $\quad\quad p_a, A' \leftarrow$ `getAlarmPos`$(t_s, t, i, \mathbf{c}, A)$;
9 $\quad\quad Q$.push$((p + p_a + \mathbf{c}'_{(i,t)}, i, t, \hat{b} - 1, A'))$;
10 $\quad$ **else**
11 $\quad\quad$ **for** $j \in \mathcal{N}(i)$ **do**
12 $\quad\quad\quad p_a, A' \leftarrow$ `getAlarmPos`$(t_s, t, j, \mathbf{c}, A)$;
13 $\quad\quad\quad$ **if** $j \in \mathcal{O}$ **then** Q.push$((p + p_a + \mathbf{c}'_{(i,t)}, i, t, \hat{b} - 1, A'))$
$\quad\quad\quad Q$.push$((p + p_a + \mathbf{c}'_{(i,t)}, i, t, \hat{b}, A'))$;
14 **return** (`buildPath`(i, t), p);

elements $(p, i, t, \hat{b}, A)$ of priority queue Q consist of the probability to be caught p (this is the priority), the current position i, the current time t, the remaining time to finish the siphoning $\hat{b}$ and a list A of points which already have been considered as starting points of a patrol boat in case of an alarm. A is specific to the path and so cannot be precalculated. The initialization (line 3) also ensures that the first a time steps are spent in the same zone. Further, we keep track of nodes visited on the way to each node, so we can create the correct path at the end of the algorithm. In the main part of the algorithm (lines 4 to 13), we consider the node which currently has the lowest probability for detection, if the siphoning is finished we can stop (line 7), otherwise if the siphoning has started, we continue siphoning (lines 7 to 9) at the same position. The `getAlarmPos` finds new nodes from which a patrol starting at t_s could reach the current location in time to catch the attacker if an alarm would be raised. Moreover, `getAlarmPos` returns a list A' of all locations that have been considered so far. It also returns the additional catching probability p_a corresponding to the coverage of the locations added to A'. If the siphoning has not started (lines 10 to 13), the attacker can move to a neighboring zone j and if j is a siphoning location, the attacker also can start siphoning. Some special considerations are not displayed in Algorithm 4: If the attack

does not finish in time the attack automatically fails; As soon as the catching probability $p \geq 1$ the attack will not be successful and does not need to be considered; When $a = 0$ and $i_s \in \mathcal{O}$ the siphoning can start at t_s. Different from Dijkstra's algorithm, we do not need to consider the update of probabilities of elements inside Q since the inbound weights into a node are all equal and so its weight cannot decrease after it is added to Q. Finally, the `buildPath` computes the attacker's path based on references to prior visited nodes (line 14).

The AO can find the attacker's best-response in time $O(|\mathcal{V}|^2(\log(|\mathcal{V}|) + |\mathcal{Z}|))$. The first term is the running time of the underlying Dijkstra algorithm and the second comes from the `getAlarmPos` subroutine, which finds new zones in linear time in $|\mathcal{Z}|$. So CG can solve small games efficiently. However, when the game becomes larger, e.g., with 100 zones and 12 time steps, it takes a long time to solve the coreLP, even with a small set $\mathcal{F}'$ due to the large number of variables. To further improve the scalability, we need to reduce the number of variables. Hence, we introduce an abstraction method in the next section.

4.4 Abstraction to Further Improve Scalability

Abstraction has been successfully applied to solve large-scale games by exploiting the intrinsic similarity of strategy space to shrink the game to a simpler one before solving it. As both players take paths as their strategies rather than a specific node and the zones on the attacker's escape path play a vital role for the defender to detect the attacker, the methods proposed in [68, 69, 99], which typically remove the nodes unnecessary to reduce the size of the game, cannot be directly applied to our model. The basic idea of the abstraction method applied here is that the defender can only determine the coverage of a large area, which can be assigned to small areas following a pre-defined mapping method, which ensures that the generated coverage satisfies the flow conservation constraints. We note that the coverage of a specific node cannot influence the defender's utility dramatically, especially when the scale of the game becomes extreme large.

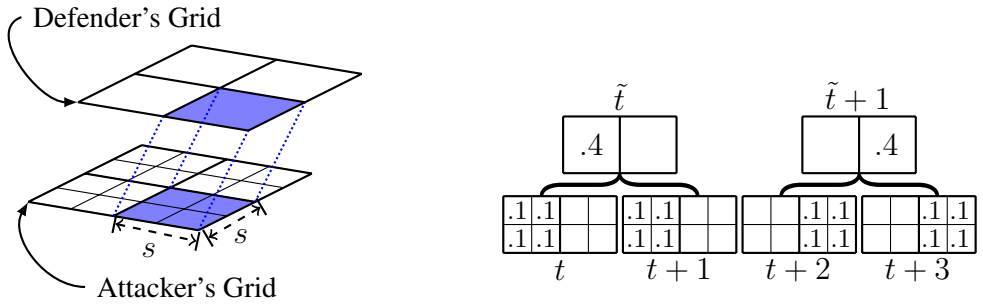
4.4.1 Abstraction of the Grid

The first step to apply the abstraction method is to obtain the abstracted defender's grid by combining $s \times s$ neighboring nodes into a node where s is named as *sparsity*. Let $\tilde{i}$ denote the node on the defender's grid and we say the $i \in \tilde{i}, i \in \mathcal{Z}$ if i is included in $\tilde{i}$. We assume that the width and the length of the grid M, as well as τ, are divisible by s. The original grid M is referred as the attacker's grid. Figure 4.3a shows both defender's and attacker's grids. Now the defender's strategy is the valid coverage on the defender's grid, i.e., the coverage can be satisfied by some flow $\tilde{\mathbf{f}}$, denoted by $\tilde{\mathbf{c}}$ and the attacker's pure strategy is the same as previous. Note that because the node of the defender's grid is $s \times s$ times as large as the node of the attacker's grid, the length of a defender's time step is s times of the attacker's. To compute the attacker's best-response to a given $\tilde{\mathbf{c}}$, as well as the corresponding utilities, we need to map the coverage from the defender's grid to the attacker's grid.

4.4.2 Mapping Method Ψ

In applying the abstraction method, we need to map the coverage vector $\tilde{\mathbf{c}}$ to the attacker's grid. The mapping method Ψ must ensure that the mapped coverage $\mathbf{c}$ is executable by some mixed strategy, i.e., there exists a flow $\mathbf{f}$ which satisfies Eqs.(4.1c) and (4.1d) on the attacker's grid. A straightforward method is uniform mapping, which uniformly assigns the flow over the nodes over each attacker's time step. However, Figure 4.3b provides a counterexample to show that uniform mapping cannot work. Therefore, we propose Algorithm 5 to generate the valid coverage on the attacker's grid.

The basic idea of Algorithm 5 is keeping the trace of flow $\tilde{\mathbf{f}}$ on the defender's grid. Figure 4.4 is an example which shows the change of a flow at each attacker's time step from a node to one of its neighbors on the defender's grid. For each attacker's time step, if the node is covered, we will add the flow into its coverage. Note that the flow is uniformly assigned to each node it covers. After considering all flows on the defender's grid, we obtain the coverage $\mathbf{c}$.



(a) (b)

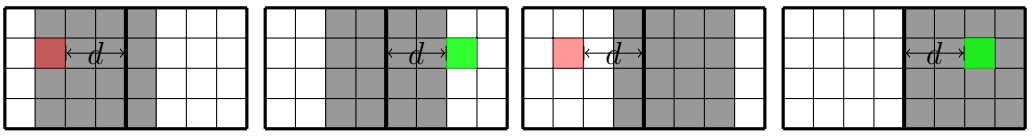
FIGURE 4.3: (a) Abstraction with $s = 2$. (b) A counterexample which shows that the uniform mapping method cannot generate the valid coverage $\mathbf{c}$. The coverage of the defender's grid from $\tilde{t}$ to $\tilde{t}+1$ is valid for the defender's grid. While we use uniform mapping to map the coverage to the attacker's grid, the coverage at $t+2$ cannot be implemented by any flow from the coverage at $t+1$.

Algorithm 5: Mapping method $\Psi(\tilde{\mathbf{f}})$

1 $\mathbf{c} = \mathbf{0}$;
2 **for** $\tilde{f}_{(\tilde{i},\tilde{t}_k),(\tilde{j},\tilde{t}_{k+1})} \in \tilde{\mathbf{f}}$ **do**
3 $t \leftarrow s \cdot \tilde{t}_k$;
4 **if** $\tilde{i} \neq \tilde{j}$ **then**
5 **for** $\Delta t = 1 : s$ **do**
6 **for** $i \in \tilde{i}$ && $d_i + \Delta t < s$ **do**
7 $c(i, t+\Delta t) += (1/s^2) \cdot \tilde{f}_{(\tilde{i},\tilde{t}_k),(\tilde{j},\tilde{t}_{k+1})}$;
8 **for** $j \in \tilde{j}$ && $d_j < \Delta t$ **do**
9 $c(j, t+\Delta t) += (1/s^2) \cdot \tilde{f}_{(\tilde{i},\tilde{t}_k),(\tilde{j},\tilde{t}_{k+1})}$;
10 **else**
11 **for** $\Delta t = 1 : s$ && $i \in \tilde{i}$ **do**
12 $c(i, t+\Delta t) += (1/s^2) \cdot \tilde{f}_{(\tilde{i},\tilde{t}_k),(\tilde{j},\tilde{t}_{k+1})}$;
13 **return** $\mathbf{c}$;

In detail, we first map the defender's time step $\tilde{t}_k$ to the attacker's (line 3). For each flow $\tilde{f}_{(\tilde{i},\tilde{t}_k),(\tilde{j},\tilde{t}_{k+1})}$ such that $\tilde{i} \neq \tilde{j}$, which implies a flow flowing from one node to another different node, we define the distance d_i for each node $i \in \tilde{i}$, as the distance from node i to the edge where the flow flows out. The distance $d_j, j \in \tilde{j}$ is defined analogously as the distance from node j to the edge where the flow flows in. For each attacker's time step Δt between $\tilde{t}_k$ and $\tilde{t}_{k+1}$ (line 5), for node $i \in \tilde{i}$, if $d_i + \Delta t < s$, which implies the flow does not leave this node, we will add the flow into i's coverage (lines 6 to 7). Analogously, if $d_j < \Delta t$, which implies the flow reaches this node, we

will add the flow into j's coverage (lines 8 to 9). When $\Delta t = s$, the flow $\tilde{f}_{(\tilde{i},\tilde{t}_k),(\tilde{j},\tilde{t}_{k+1})}$ reaches $\tilde{j}$. For the flow $\tilde{f}_{(\tilde{i},\tilde{t}_k),(\tilde{j},\tilde{t}_{k+1})}$ with $\tilde{i} = \tilde{j}$, which implies a flow from a node to itself, we simply assign the flow to the nodes $i \in \tilde{i}$ uniformly (lines 11 to 12). Then we obtain the coverage $\mathbf{c}$.



(a) $\Delta t = 1$ (b) $\Delta t = 2$ (c) $\Delta t = 3$ (d) $\Delta t = 4$

FIGURE 4.4: The change of a flow from a node to one of its neighbors (right neighbor) along with the attacker's time step with $s = 4$. The gray nodes are covered by the flow at each attacker's time step. Besides, the red (green) node is one of the node of the defender's grid where the flow flows out (in) and the corresponding distance $d = 2$, defined in Algorithm 5, to the edge where the flow passes is displayed.

Proposition 4.1. *The generated* $\mathbf{c}$ *can be satisfied by a flow.*

Proof. We can find a flow to satisfy $\mathbf{c}$. For each $\tilde{f}_{(\tilde{i},\tilde{t}_k),(\tilde{j},\tilde{t}_{k+1})} \in \tilde{\mathbf{f}}$ such that $\tilde{i} \neq \tilde{j}$, for $i \in \tilde{i}$, we denote the neighbor as i' of i where the flow reaches i' before i, as well as j' for $j \in \tilde{j}$. For $\Delta t = 1 : s$, if $d_i + \Delta t < s$, $f_{(i',t-1+\Delta t)(i,t+\Delta t)} = (1/s^2) \cdot \tilde{f}_{(\tilde{i},\tilde{t}_k),(\tilde{j},\tilde{t}_{k+1})}$; and if $d_j < \Delta t$, $f_{(j',t-1+\Delta t)(j,t+\Delta t)} = (1/s^2) \cdot \tilde{f}_{(\tilde{i},\tilde{t}_k),(\tilde{j},\tilde{t}_{k+1})}$. For each $\tilde{f}_{(\tilde{i},\tilde{t}_k),(\tilde{j},\tilde{t}_{k+1})} \in \tilde{\mathbf{f}}$ such that $\tilde{i} = \tilde{j}$, $f_{(i,t-1+\Delta t)(i,t+\Delta t)} = (1/s^2) \cdot \tilde{f}_{(\tilde{i},\tilde{t}_k),(\tilde{j},\tilde{t}_{k+1})}$. Now we get $\mathbf{f}$ which satisfies $\mathbf{c}$. □

4.4.2.1 Analysis

We can use the ratio $r = \frac{U - U_s^a}{U - U^a}$ to evaluate the performance of the abstraction method where $U = \max\{u(i, t_k)\}$, U_s^a and U^a are utilities of the attacker with and without the abstraction, respectively. For general games, it is difficult to obtain a good bound for the abstraction method theoretically due to the high interdependence between time and space. However, for *uniform* games where all zones are with the same value, we have the proposition.

Proposition 4.2. *For uniform games,* $r \geq 1/s^2$.

Proof. Because the zones are with the same value in the uniform game, the attacker will only attack the zones in $\mathcal{O}$, which can bring at least the same utility if he attacks other zones. Therefore, the defender would only assign the coverage to zones in $\mathcal{O}$ and keep the coverage steady during all time steps. Suppose the optimal coverage is $\mathbf{c}$ and the corresponding attacker's utility is U^a. Then, we can construct a valid coverage for the abstraction $\mathbf{c}_s$ with sparsity s: for each time step $t_k, k \in \{1, \cdots, \tau\}$, $c_s(i, t_k) = \frac{1}{s^2} \sum_{j \in \tilde{i}} c(j, t_k), \forall\ i \in \tilde{i}$, which implies that $c_s(i, t_k) \geq \frac{1}{s^2} c(i, t_k)$. As the values of all zones are the same, $r = \frac{dpp(\mathbf{c}_s, F_s)}{dpp(\mathbf{c}, F)}$ where F_s and F are the attacker's best-response to $\mathbf{c}_s$ and $\mathbf{c}$, respectively. If $F_s = F$, $r \geq \frac{1}{s^2}$ due to the fact that $c_s(i, t_k) \geq \frac{1}{s^2} c(i, t_k)$. If $F_s \neq F$, as F is the best response to $\mathbf{c}$, we can obtain that $dpp(\mathbf{c}_s, F_s) \geq \frac{1}{s^2} dpp(\mathbf{c}, F_s) \geq \frac{1}{s^2} dpp(\mathbf{c}, F)$, which leads to $r \geq \frac{1}{s^2}$. □

Proposition 4.2 provides the worst case guarantee and we found that the abstraction can achieve a close approximation to optimal solution quality with a dramatic improvement of scalability in the experimental evaluation.

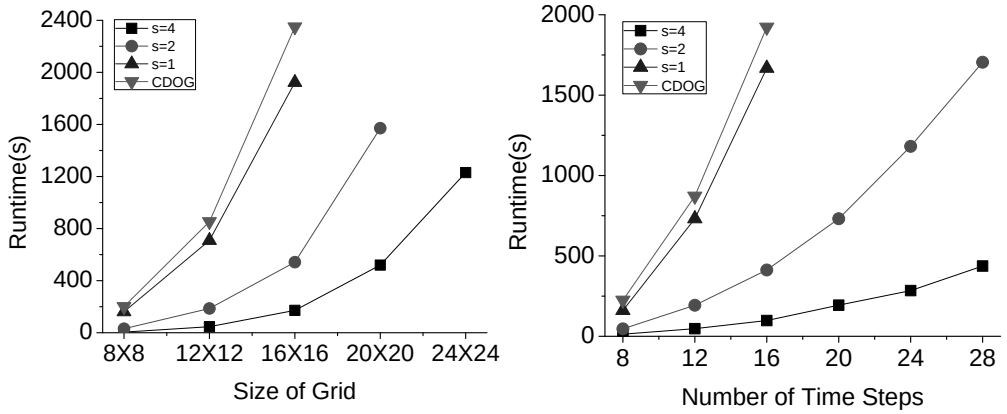
4.5 Experimental Evaluation

We evaluate the performance of our approach based on i) extensive experiments completed via simulations and ii) real-world ship density data of CSC. We use CPLEX (version 12.6) to solve the LPs and all computations are performed on a 64-bit PC with 16.0 GB RAM and a 3.50 GHz CPU. The detecting factor d, the external detecting factor e and the alarm probability α are fixed as 0.5, 0.1 and 0.25, respectively, unless otherwise specified.

4.5.1 Experimental Evaluation on Synthetic Data

In this section, we systematically generate square grids with different sizes. The values of nodes of grids are randomly generated from a uniform distribution in $[0, 100]$. For all simulations, the attack time a and the siphoning time b are fixed as 2 and 3, respectively. All values are averaged over 30 instances unless otherwise specified.

We compare the scalability of three versions of our algorithms: i) our CG algorithm, which is denoted as $s = 1$ in the figures; ii) our abstraction method with $s = 2$; and iii) our abstraction method with $s = 4$. The CDOG algorithm proposed in [41] is also implemented as a benchmark with a slight modification that we relax the coverage vector $\mathbf{c}$ to be continuous. To evaluate the solution quality, we compare our algorithms' solutions with two heuristic patrol strategies: i) **rand** where $c(i, t_k) = m/|\mathcal{Z}|$, which implies that the defender randomly patrols all zones; and ii) **prop** where $c(i, t_k) = \sum_{t_k=1}^{\tau} u(i, t_k) \cdot m / \sum_{i=1}^{|\mathcal{Z}|} \sum_{t_k=1}^{\tau} u(i, t_k)$, which implies that the defender patrols the zones proportionally to the values of zones. Note that when $s = 1$, the abstraction method is the same as CG.

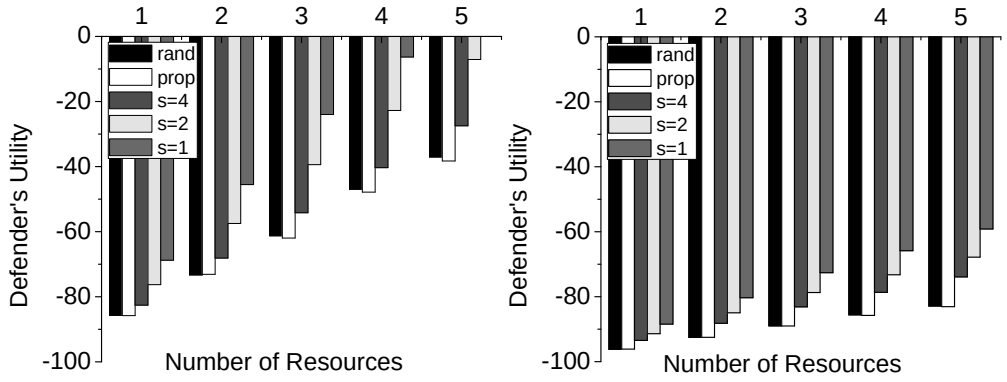


(a) $m = 3, \tau = 12$ (b) $|\mathcal{Z}| = 12 \times 12, m = 3$

FIGURE 4.5: Runtime analysis

Scalability analysis. We first compare the scalability of our algorithms. The result is depicted in Figure 4.5a where the x-axis indicates the size of the grid. The result shows that neither CG nor CDOG cannot scale up to the grid larger than 16×16 with runtime cap of 2400 seconds. Besides, CG is always faster than CDOG because CDOG needs to call the defender oracle many times to generate the subgrid. While, our abstraction methods with $s = 4$ and $s = 2$ can solve the grid with 20×20 in less than 1600 seconds, which significantly outperform the CG and CDOG. We also compare the scalability of our algorithm on games with different time length. The result is showed in Figure 4.5b. The result shows that our abstraction method can scale up to the realistic length of time steps and for small sparsity, the runtime increases faster because when τ increases, more variables and constraints are added into the formulation. We also vary

the number of resources, which has much less influence on the runtime and the result is not displayed.

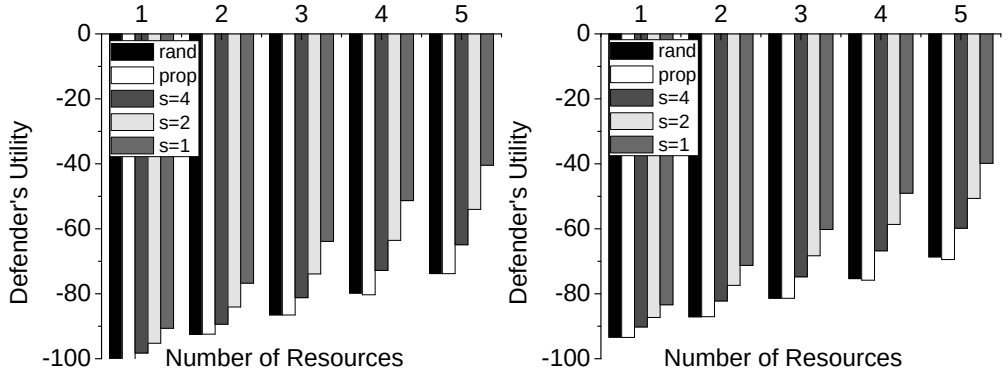


(a) $|\mathcal{Z}| = 8 \times 8, \tau = 12$ (b) $|\mathcal{Z}| = 16 \times 16, \tau = 12$

FIGURE 4.6: Solution quality

Solution quality. As the main advantage of the abstraction method is that it can significantly improve the scalability with some loss of the optimality, we then evaluate the quality of the solutions obtained by the abstraction method. The results, displayed in Figures 4.6a and 4.6b, show that given the grids, when the number of resources m increases, our abstraction method shows a greater advantage against the random patrol and when the sparsity s becomes smaller, the abstraction method provides a better approximation to the optimal solution. It is worth noting that in all experiments, the abstraction obtains a much higher value of the ratio r $(0.6 \sim 0.8)$ compared with the theoretical bounds. Given the number of resources m, when the gird becomes larger, the advantage of our solutions against the two baselines is reduced. Besides, the proportional patrol strategy gets even worse utility than the random patrol strategy in some cases, because patrolling the nodes with low values is also important to catch the attacker who takes paths as strategies.

Robustness analysis. In reality, the defender may not know the exact value of each zone due to the fluctuation of ship density over time. Besides, during the patrol process, the real coverage of each zone may deviate from the defender's strategy. Thus, we evaluate the robustness of our solutions in two aspects: i) the real value of each zone $\hat{u}(i, t_k)$ ranges in $[1 - \delta, 1 + \delta] \cdot u(i, t_k)$, and ii) the real coverage of each zone $\hat{c}(i, t_k)$ ranges in $[1-\rho, 1+\rho] \cdot c(i, t_k)$. The results are displayed in Figure 4.7 with $\delta = \rho = 0.1$. The results show that our solutions are robust enough and outperform the random patrol



(a) $\hat{u}$ ($\delta = 0.1$) (b) $\hat{c}$ ($\rho = 0.1$)

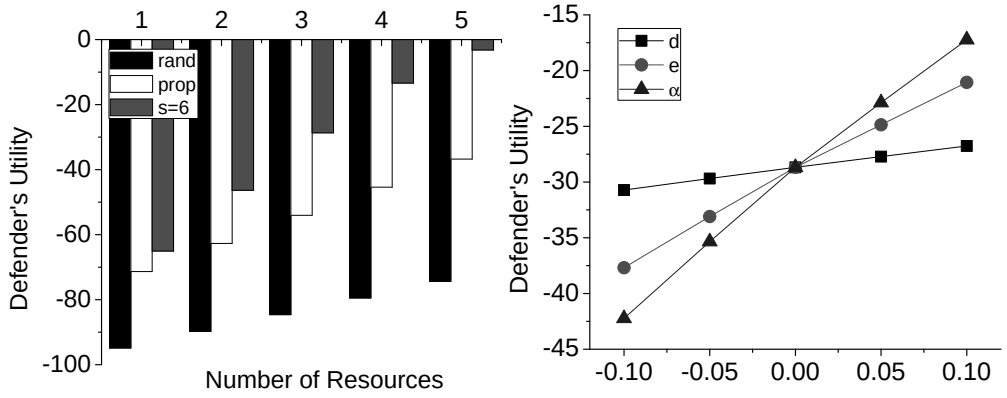
FIGURE 4.7: Robustness

strategy under a high level of uncertainty of targets' values and coverage. When m is extremely small, the advantage of our solution with large sparsity is reduced.

4.5.2 SCS Real-world Evaluation

We consider an area of 64800 NM^2 in the south of the SCS showed in Figure 4.1. We define the attacker's grid by dividing this area into small square zones with a length of 10 NM, which ensures that a patrol boat can provide efficient protection of a zone at each time step through its radar system. By a normal patrol speed of 15 knots, it takes 40 minutes from one zone to the next [106], so we consider 18 time steps for twelve hours of darkness, i.e., $\tau = 18$. We assume $a = 4$ and $b = 6$ which imply that the attackers need roughly three hours to capture a ship and four to siphon the goods according to the incident reports. In order to determine the value of each zone we obtained data from AIS used by vessel traffic services. The data cover the first week of February 2017 and after deletion of faulty entries, it contained about 20000 entries in the considered area. In 500 of the 648 zones no ship appeared during the week, we declare these zones as siphoning locations. The value of each zone at each time point was then mapped to lie in $[0, 100]$ where all locations with 30 or more ships came through in one time step were mapped to a value of 100[1].

[1]A flaw of AIS data is that there are some zones which have extremely larger traffic density than their neighbors at some time step. To make the data reasonable, zones with more than 30 are considered outliers, otherwise they will dominate the game.



(a) Solution quality (b) Range factors

FIGURE 4.8: Real-world application

The results are depicted in Figure 4.8. All instances are solved averagely in 2800 seconds, which is better than our simulated results due to the vast zones with value 0, which reduces the strategy space of both players. In Figure 4.8a, we range the number of resources and choose $s = 6$ to our abstraction method. The results show that our algorithm significantly outperforms the two baselines. In Figure 4.8b, we range different factors d, e and α by $[-0.1, 0.1]$ around the default values with $m = 3$ and $s = 6$. The results show that improving the alarm probability α has more influence on the utility, compared with d and e, because it increases the detecting probability of all neighboring zones near the attack, so we highly recommend that the ships install fast and responsive alarm system. Besides, improving e is more effective than improving d to improve the defender's utility due to the improvement of coverage of all neighbors of nodes.

4.6 Chapter Summary

This chapter tries to address the problem of preventing oil siphoning in international waters through efficient patrols. A novel Stackelberg model based on the traffic data is proposed. A compact formulation and a constraint generation algorithm with an efficient attacker oracle is proposed. To further improve the scalability, an abstraction method is proposed. Extensive experimental results and a detailed case study for SCS show that our approaches can result in dramatic improvement of scalability with modest influence on the solution quality and can scale up to realistic-sized problems.

Chapter 5

Preventing Cyber Attacks on Interdependent Users through Government's Subsidy

Cybersecurity has become one of the most important issues of the modern society, and cyber attacks can cause extremely high loss to many companies, organizations and governments. It is obviously clear that, from these recent cyber attacks, businesses are not doing enough to protect themselves and/or do not have sufficient awareness of cyber threats. Because companies and organizations are interdependent in cyber space, cyber attacks could propagate from one company to other organizations. As a result, the lack of protections against cyber threats not only causes damages to companies and organizations but also decreases the cybersecurity level of the country as a whole [1].

To combat these issues, governments around the world have launched various initiatives to improve the national cybersecurity level. For example, the UK government provided millions of subsidies, i.e., grants and vouchers, for businesses to boost their cybersecurity in 2015 and 2016 [107]. The UK government's goal is [34]

> *to intervene more actively and use increased investment, while continuing to support market forces to raise cyber security standards across the UK.*

[1] The national cybersecurity level in this chapter is defined as the social welfare for companies across a country against cyber crimes.

The government's goal is optimally assigning limited subsidies to companies to protect against the attacker. However, this assignment task is highly challenging because cyber companies are self-interested and interdependent (induced by the spreadability of cyber attacks) when making cyber protection decisions. In this work, we adopt game-theoretic methodologies to analyze cyber interactions and help the government to improve the national cybersecurity through subsidies.

There are several lines of the research related to this work. The first line is *interdependent security game* [70, 73] where the users are interdependent and can experience direct risk from internal contamination and indirect risk transferred from their neighbors. The indirect risk can be viewed as *stochastic one-hop propagation*[1] model [108]. An extension of these works is *interdependent defense game* [74, 76, 109] where an attacker is added into the game who may strategically attack the interdependent users. However, in these works, they focus on the computation of Nash equilibria between users and the attacker and there is no *defender* who tries to optimize the global security level (i.e., minimizing the sum of the loss of users caused by the attacks). The second line of the related research is *interdependent information security game* [108]. The famous Gordon-Loeb model [110] is a single-agent decision model, whose parameters are similar to our model, and its extension [111] studies the information sharing between two agents, which differs from our model. Various schemes are proposed to improve the security level, which are based on either game-theoretic equilibrium improvements [112] or mechanisms such as mandatory or optional insurances [113], subsidies and fines [114] and regulations [115]. However, the mechanisms in these works are modeled as a part of agent's utility explicitly, which is not determined by a strategic agent as in our setting. Besides, there is no strategic attacker in all these works. The third line of the related research is *Stackelberg security game* [2] for security problems where the defender allocates limited resources to protect valuable targets against the attacker. Some recent works extended the method to cybersecurity problems, such as allocating cyber alerts [14] and deceiving cyber adversaries [116]. Some works consider the games where multiple types of independent followers following a known distribution [6], the

[1] In one hop transfer, the attack can only be transferred to user(s) who are directly connected to the attacked user. The model can be easily generalized to the case where the attack can spread beyond one-hop transfer.

interdependence between targets [117], the externalities of the protection [101] and network security games [8]. However, the interdependence between self-interested followers is not tackled in all previous works. Some works consider the multiple defenders against an attacker [118], which also cannot be applied to our problem due to the lack of the global optimizer. Recent works [119, 120] proposed the game between a leader and multiple followers. Their algorithms for the optimistic case (corresponding to our model) needs to enumerate all pure strategies of followers which is impossible in our case because we adopt a more compact representation.

First, we formulate our cybersecurity setting as a Stackelberg game played between the government, interdependent companies/users and an attacker where the government moves first to allocate subsidies and the users and attacker move simultaneously to determine their protection and attack (pure or mixed) strategies, respectively. We comprehensively investigate three settings where the attacker has different capabilities (i.e., attack all users, a single user and multiple users). For the pure-strategy case, we show that computing an optimal allocation is NP-hard when the attacker is able to attack all users. We propose a linear reverse convex program to compute an optimal allocation in this setting. However, we show that there may not be a feasible allocation in the two other settings. For the mixed-strategy case, we show that there is a polynomial time algorithm to find an optimal allocation in the single-attack setting. We then provide a heuristic algorithm, based on best-response-gradient dynamics, to find an effective allocation in the general setting. We extensively evaluate our model and the heuristic algorithm on synthetic and real data. The results show that our model captures the interdependent behaviors of users and our heuristic algorithm outperforms other baselines.

5.1 Motivation Scenario

We use the UK as a motivating example to illustrate the applicability of our model. From 2015 to 2020, the UK government will invest £1.9 billion to improve its cybersecurity through various schemes, such as grants, vouchers and subsidies [107]. The UK

government also builds the national cyber security centre (NCSC)[1] to manage cyber incidents, analyze cyber threats and provide tailored expertise support to businesses [34]. With the help of NCSC, companies can obtain accurate information of cyber space and learn optimal cyber investment strategies. The information includes the vulnerabilities of businesses and the connections between businesses. The information is provided with the existence of the strategic cyber attacker (e.g., hacktivist) [34].

We assume that companies are self-interested and there is no cooperation or coordination among them. It is worth noting that it is difficult for the attacker to observe companies' strategies before the attacks due to the lack of transparency of the cyber space. For example, the attacker cannot observe anti-spam filtering systems of companies before sending spam emails in phishing attacks. Therefore, we assume that companies and the attacker move simultaneously. We note that Nash equilibrium is a canonical solution concept for non-cooperative simultaneous-move settings.

5.2 Stackelberg Cybersecurity Investment Game

A **S**tackelberg **C**ybersecurity **I**nvestment **G**ame (SCIG) is played between a *government* (e.g., cybersecurity agency), a set of interdependent *users* (e.g., companies and organizations) and an *attacker*. The whole procedure of the game can be divided into two stages: i) the government allocates budgeted subsidies to users, ii) each user, after obtaining the government's subsidy amount, and the attacker decide their own strategies simultaneously[2]. Both users and the attacker are termed as *followers*. The interactions between followers exactly follow the assumptions and models in *interdependent security (defense) game*, which is widely investigated in [70, 73, 74, 76].

[1] https://www.ncsc.gov.uk/

[2] The assumption that users and the attacker play simultaneously is motivated by the fact that players make strategic reasoning and observation of each other before committing to a particular strategy (i.e., Nash equilibrium) which is natural in cybersecurity scenarios because users (e.g., Symantec) spent millions of dollars to hire cyber experts and IT professionals to strategize the potential move of attacker, as well as for the attacker. Besides, users will protect their investment strategies from being observed by the attacker to avoid the exploitation.

Formally, we consider N interdependent users, of which each user $i \in [N] = \{1, 2, ..., N\}$ is characterized by a tuple with parameters $\langle p_i, c_i, l_i \rangle$ where p_i is the probability that user i will be compromised if being attacked by the attacker, i.e., *direct attack*, c_i is the cost of user i to get a cybersecurity system such as installing firewalls, building intrusion detection systems and investigation programs, to prevent himself from the contamination, and l_i is the loss user i may suffer if he is compromised. To avoid the trivial case, we assume that $c_i < p_i l_i, \forall i \in [N]$. We use $\boldsymbol{q} = \langle q_{ji} \rangle$ to model the spread of the attack between users. For each pair $j, i \in [N], j \neq i$, let q_{ji} denote the probability that user i is compromised as a result of a transfer of the attack from j, i.e., *indirect attack*.

Strategies. The government's strategy is denoted by the vector $\boldsymbol{x} = \langle x_i \rangle$ where x_i is the subsidy assigned to user i, constrained by the budget B (i.e., $\sum_{i=1}^{N} x_i \leq B$). The subsidies can only be used to invest in cybersecurity. The users' pure strategies are denoted by $\boldsymbol{a} = \langle a_i \rangle$ where $a_i = 1$ if user i invests, otherwise $a_i = 0$[1]. The users' mixed strategies are denoted by $\boldsymbol{y}$ where $y_i \in [0, 1]$ is the probability that user i will invest in cybersecurity. The attacker's pure strategy is denoted by the vector $\boldsymbol{b} = \langle b_i \rangle$ where $b_i = 1$ implies the attacker attacks user i, otherwise $b_i = 0$. Note that $\sum_{i=1}^{N} b_i \leq K$ where K is the number of users the attacker can attack. We use $\mathcal{B}^K$ to denote the set of pure strategies of the attacker. The mixed strategy of the attacker is denoted by $\boldsymbol{z}$ which is a distribution over all pure strategies in $\mathcal{B}^K$.

Utilities. Given a strategy profile $\langle \boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b} \rangle$ where the followers take pure strategies, the users' utilities are defined as

$$\begin{aligned} U_i^u(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}) = a_i(x_i - c_i) - \left[1 - (1 - b_i p_i)^{(1-a_i)}\right] l_i \\ -[(1-b_i p_i)^{(1-a_i)}] \left[1 - \prod\nolimits_{j \neq i} (1 - b_i q_{ji})^{(1-a_j)}\right] l_i. \end{aligned} \tag{5.1}$$

The first term of Eq.(5.1) is the investment cost of user i with the subsidy x_i assigned by the government where the user will obtain the subsidy if he invests, i.e.,

[1] In this work we consider the case where the users can only make binary decisions, which is motivated by the scenario of updating the patches of anti-virus software and buying the cyber insurance, where the users can only decide whether to invest or not. Our model can be generalized to the case where the users can make continuous investments in cybersecurity, such as the investments on firewalls or intrusion detection systems, which differs from the current model and will be discussed in future works.

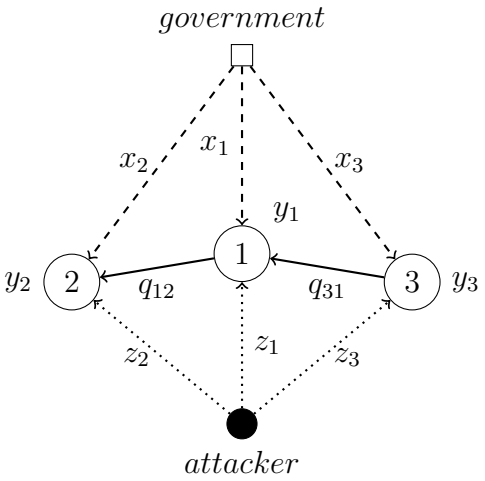


FIGURE 5.1: An illustrative example of SCIG where the square node is the government who assigns the subsidy to users (circle nodes) first, and users and the attacker (filled circle node) make their decisions simultaneously. Solid arrows between users indicate indirect risks.

$a_i = 1$. The second term is the expected loss to i due to the direct attack, i.e., $b_i p_i l_i$ if $a_i = 0$ and 0 if $a_i = 1$. The third term is the expected loss due to the indirect attack transferred from others, i.e., $(1 - b_i p_i)\left[1 - \prod_{j \neq i}(1 - b_i q_{ji})^{(1-a_j)}\right] l_i$ if $a_i = 0$ and $\left[1 - \prod_{j \neq i}(1 - b_i q_{ji})^{(1-a_j)}\right] l_i$ if $a_i = 1$ where $1 - \prod_{j \neq i}(1 - b_i q_{ji})^{(1-a_j)}$ is the probability that at least one of user i's neighbors will transfer the contamination to him. The utilities (omitting the x_i's) and terms are defined the same way as in [73, 76]. The government's utility is defined to be the negative summation of the expected loss of users

$$U^d(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}) = \sum_{i=1}^{N} [U_i^u(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}) - a_i(x_i - c_i)] \tag{5.2}$$

We use the user's utilities to define the government's utility for simplicity and note that the government's utility is independent of the terms $a_i(x_i - c_i)$, which are canceled out when substituting Eq.(5.1) into Eq.(5.2). The attacker's utility is defined as $U^a(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b}) = -U^d(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{b})$. Analogously, given a strategy profile $\langle \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \rangle$ where the followers take mixed strategies, the users' utility is

$$U_i^u(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = y_i[(x_i - c_i) - \Psi_i^1 l_i] - (1 - y_i)[\Psi_i^2 p_i + \Psi_i^1] l_i$$

The first term $y_i[(x_i - c_i) - \Psi_i^1 l_i]$ is the user's utility when he chooses to invest and $\Psi_i^1 = \sum_{\boldsymbol{b} \in \mathcal{B}^K} z(\boldsymbol{b}) \left[1 - \prod_{j \neq i}(1 - (1 - y_j) b_j q_{ji})\right]$ is the indirect risk of the user where

$1 - \prod_{j \neq i}(1 - (1 - y_j)b_j q_{ji})$ is the probability that the user will be compromised when other users take $\boldsymbol{y}$ and the attacker takes $\boldsymbol{b}$, i.e., the probability that there is at least one of the neighbors of user i will transfer the contamination to him. The second term $-(1 - y_i)[\Psi_i^2 p_i + \Psi_i^1]l_i$ is the user's utility when he chooses not to invest where Ψ_l^1 is the same as introduced before and $\Psi_i^2 = \sum_{\boldsymbol{b} \in \mathcal{B}^K} z(\boldsymbol{b}) b_i \left[\prod_{j \neq i}(1 - (1 - y_j)b_j q_{ji})\right]$ denotes the probability that user i is attacked and there is no neighbor of user i will transfer the contamination to him because for $\boldsymbol{b} \in \mathcal{B}^K$, if $b_i = 1$, which means that user i is attacked and we add $\prod_{j \neq i}(1 - (1 - y_j)b_j q_{ji})$ into Ψ_i^2, which is the probability that no user transfers the contamination to user i. We can check all the three cases of user i will suffer when he does not invest to verify the correctness that $\Psi_i^2 p_i + \Psi_i^1$, which are i) user i is successfully attacked, $\sum_{\boldsymbol{b} \in \mathcal{B}^K} z(\boldsymbol{b}) b_i p_i$, no matter whether the neighbors transfer the contamination to user i or not because the user can only be compromised once, ii) user i is attacked but not compromised and there is at least one of user i's neighbors transfers the contamination to him, $\sum_{\boldsymbol{b} \in \mathcal{B}^K} z(\boldsymbol{b}) b_i (1 - p_i) \left[1 - \prod_{j \neq i}(1 - (1 - y_j)b_j q_{ji})\right]$ and iii) user i is not attacked and there is at least one of user i's neighbors transfers the contamination to him, $\sum_{\boldsymbol{b} \in \mathcal{B}^K} z(\boldsymbol{b})(1 - b_i) \left[1 - \prod_{j \neq i}(1 - (1 - y_j)b_j q_{ji})\right]$. Readers can obtain the term $\Psi_i^2 p_i + \Psi_i^1$ by summing all three cases. The government's utility for mixed strategies $U^d(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ is defined analogously, also the attacker's utility $U^a(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.

Equilibrium. Our objective is to find the optimal strategy for the government to assign the subsidy to the users, as well as the users' and the attacker's strategies. In particular, we are interested in the Stackelberg equilibrium between the government and followers (the users and the attacker), and Nash equilibrium between followers. We denote this notion as **S**tackelberg-**P**ure-**N**ash **E**quilibrium (SPNE) when the followers take pure strategies and **S**tackelberg-**M**ixed-**N**ash **E**quilibrium (SMNE) when the followers take mixed strategies, respectively. We assume that users break ties in favor of the government, i.e., if there are multiple equilibria, followers would select the one maximizing the government's utility. This is the standard assumption in Stackelberg security game [2]. As the attacker's utility is the negation of the government's utility, the attacker does not need to break ties.

5.3 Solution Approach

In this section, we consider computing SPNE and SMNE in three settings where the attacker has different capabilities. We first show that SPNE always exists in all-user attacks and does not exist in the other two settings. We show that computing an SPNE is NP-hard and propose a linear reverse convex formulation to compute an SPNE. In the single-user attack setting, we provide an exact polynomial algorithm to compute an SMNE. Then, we provide an effective heuristic algorithm to compute an SMNE in all-user and multiple-user attacks.

5.3.1 All-user Attack: $K \geq N$

In this section, we consider the case where the attacker is able to attack all users, which is reasonable when the attack to users is easy and can be widely spread, such as worm attacks and phishing attacks. We note that the attacker will always attack all users due to the non-negative utilities an attack can obtain. We focus on the computation of SPNE in this section. SMNE will be discussed in the multiple-user attack case.

Given the strategy $\boldsymbol{x}$, there is at least one PNE, which can be computed in time $O(N^2)$, while computing all PNE of users is NP-complete [73]. Therefore, there always exists at least a PNE for all-user attacks. However, Theorem 5.1 proves that computing an SPNE is NP-hard.

Theorem 5.1. *Computing an SPNE is NP-hard.*

Proof. We reduce from the knapsack problem which is known to be NP-hard. In a knapsack problem, we are given a set of items $[N]$ and a budget $W > 0$. Each item $i \in [N]$ has a value $f_i > 0$ and weight $w_i > 0$. The goal is to find $S \subseteq [N]$ such that $\sum_{i \in S} w_i \leq W$ and $\sum_{i \in S} f_i$ is maximum.

We reduce the knapsack problem to our problem. For each item $i \in [N]$, we introduce a user i with $\langle c_i, p_i, l_i \rangle$ and two additional connected users, i.e., u_i^1 and u_i^2 and connects u_i^1 to user i, as displayed in Figure 5.2. For pairs of users u_i^1 and u_i^2, we set $c_i^1 = c_i^2 = c'$ and $p_i^1 = p_i^2 = p'$, $l_i^1 = l_i^2 = l'$ and $q_i^{12} = q_i^{21} = q'$. We set that

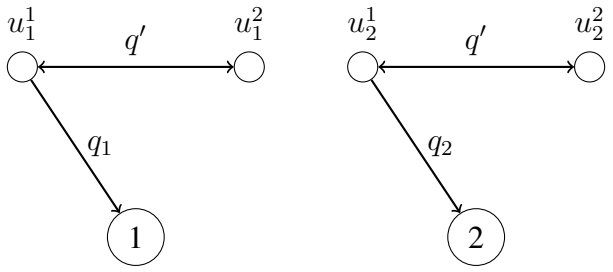


FIGURE 5.2: Reduction of the Knapsack problem to SCIG

$p'(1-q')l' < c' < q'l'$ and $c' - p'(1-q')l' > B$, i.e., any of the additional users will not invest even with the government's subsidy. For each user $i \in [N]$, the connection between u_i^1 and i is denoted as q_i and we set $p_i(1-q_i)l_i < c_i < p_i l_i$, $c_i - p_i(1-q_i)l_i = w_i$ and $p_i(1-q_i)l_i = f_i$, i.e., by assigning w_i to user i, the user would invest and the government gains a positive utility $p_i(1-q_i)l_i$. The government's budget B is set to be equal to W. Given a solution of the knapsack problem $S \subseteq [N]$, the solution is also ensured to be optimal to our problem, and vice versa. Note that the government will never assign the subsidy to the additional users. □

We can formulate the problem to compute an SPNE as Program 5.3, which is a bilevel optimization problem:

$$\max_{\boldsymbol{x},\boldsymbol{a}} U^d(\boldsymbol{x}, \boldsymbol{a}, \mathbf{1}) \tag{5.3a}$$

$$\text{s.t. } a_i \in \arg\max_{a_i \in \{0,1\}} \{U_i^u(\boldsymbol{x}, \boldsymbol{a}, \mathbf{1})\} \tag{5.3b}$$

$$\sum_{i=1}^{N} x_i \leq B \tag{5.3c}$$

To make the problem computable, we reduce the bilevel formulation to single level optimization by rewriting each user i's utility as

$$\begin{aligned} U_i^u(\boldsymbol{x}, \boldsymbol{a}, \mathbf{1}) &= a_i(x_i - c_i) - l_i \\ &\quad + (1-p_i)^{(1-a_i)} \prod\nolimits_{j \neq i} (1-q_{ji})^{(1-a_j)} l_i \qquad (5.4) \\ &= a_i \left[x_i - c_i + p_i l_i \prod\nolimits_{j \neq i} (1-q_{ji})^{(1-a_j)} \right] + \Omega_i \qquad (5.5) \end{aligned}$$

Eq.(5.4) can be easily obtained from Eq.(5.1). The reduction from Eq.(5.4) to Eq.(5.5) is due to the fact that $(1-p_i)^{(1-a_i)} = 1 - (1-a_i)p_i$ when $a_i \in \{0,1\}$ and $\Omega_i =$

$(1-p_i)\prod_{j\neq i}(1-q_{ji})^{(1-a_j)} - l_i$, which is independent of a_i and can be ignored when users maximize their own utilities. Therefore, we introduce Program 5.6 which is a single level optimization problem.

$$\max_{\boldsymbol{x},\boldsymbol{a}} U^d(\boldsymbol{x},\boldsymbol{a},\mathbf{1}) \tag{5.6a}$$

$$\text{s.t. } \left(x_i - c_i + p_i l_i \prod_{j\neq i}(1-q_{ji})^{(1-a_j)}\right) a_i \geq 0 \tag{5.6b}$$

$$\sum_{i=1}^{N} x_i \leq B \tag{5.6c}$$

$$a_i \in \{0,1\} \tag{5.6d}$$

Eq.(5.6b) ensures that if $x_i - c_i + p_i l_i \prod_{j\neq i}(1-q_{ji})^{(1-a_j)} < 0$, the user will not invest, i.e., $a_i = 0$, otherwise $a_i = 1$, which is straightforward from Eq.(5.5). Theorem 5.2 proves that Program 5.6 can be reformulated as a linear reverse convex program with a linear objective and constraints as $g(x) \geq 0$ where $g(x)$ is a convex function [88].

Theorem 5.2. *Program 5.6 can be reformulated as a linear reverse convex program.*

Proof. The utility function of user i can be reformulated as

$$U_i^u(\boldsymbol{x},\boldsymbol{a},\mathbf{1}) = a_i(x_i - c_i) + \prod_{j\in[N]}(1-q_{ji})^{(1-a_j)} l_i - l_i \tag{5.7}$$

For simplicity, we denote p_i as q_{ii}. Thus, the government's utility, i.e., the objective of Program 5.6, is reformulated as

$$U^d(\boldsymbol{x},\boldsymbol{a},\mathbf{1}) = \sum_{i\in[N]} \left[\prod_{j\in[N]}(1-q_{ji})^{1-a_j}\right] - \sum_{i\in[N]} l_i.$$

It can be easily verified by computing the Hessian matrix of $U^d(\boldsymbol{x},\boldsymbol{a},\mathbf{1})$ that $U^d(\boldsymbol{x},\boldsymbol{a},\mathbf{1})$ is convex and $\sum_{i\in[N]} l_i$ is constant. Then, Eq.(5.6b) can be equivalently rewritten as

$$v_i - a_i c_i + z_i \geq 0 \tag{5.8a}$$

$$0 \leq v_i \leq x_i \tag{5.8b}$$

$$x_i - (1-a_i)M \leq v_i \leq a_i M \tag{5.8c}$$

$$0 \leq z_i \leq p_i l_i \cdot w_i \tag{5.8d}$$

$$p_i l_i \cdot w_i - (1-a_i)M \leq z_i \leq a_i M \tag{5.8e}$$

$$w_i \leq \prod\nolimits_{j \neq i} (1 - q_{ji})^{1-a_j} \tag{5.8f}$$

Eqs.(5.8b)-(5.8c) ensure that if $a_i = 0$, $v_i = x_i$ and $v_i = 0$ otherwise. Eq.(5.8d)-(5.8e) ensure that if $a_i = 0$, $z_i = p_i l_i \cdot w_i$ and $v_i = 0$ otherwise, where M is a big constant. To maximize the government utility, w_i will always be equal to $\prod_{j \neq i} (1 - q_{ji})^{1-a_j}$ in Eq.(5.8f), which is a reverse convex constraint. Besides, as the variables a_i are binary, we can also add a reverse convex constraint into the program for each a_i

$$a_i^2 - a_i \geq 0, 0 \leq a_i \leq 1 \tag{5.9}$$

Furthermore, we introduce an auxiliary variable U as the linear objective with an additional reverse convex constraint

$$U \leq \sum\nolimits_{i \in [N]} \left[\prod\nolimits_{j \in [N]} (1 - q_{ji})^{1-a_j} \right] l_i - \sum\nolimits_{i \in [N]} l_i \tag{5.10}$$

Thus, we obtain the linear reverse convex program. □

The reformulation in Theorem 5.2 ensures that all terms in the objective and constraints are convex, which makes it easier to solve. In this work, we use global optimization solvers, e.g., BARON, to solve the problem.

5.3.2 Single-user Attack: $K = 1$

We then consider the game where the attacker can only attack one user. This case is reasonable when the attack is rare and complicated but with extremely huge impact, such as DDoS attacks, thus the attacker can only pick the optimal one to attack. The single-user attack without the government role is extensively investigated in [74–76, 109], which provides some useful results for us to develop our algorithm. The following proposition proves that, except the case where all users invest, there is no other pure NE between followers when the attacker can strategically attack one user. We consider the game where the attacker can only attack one user. The setting without the government is extensively investigated in [74, 76], which provides some useful results for us to develop our algorithm.

Proposition 5.3. *Given the strategy $\boldsymbol{x}$, if $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \langle \mathbf{1}, * \rangle$ is not an NE, then there is no pure NE between followers.*

Proof. As there is at most a user being attacked, we can write the attacker's utility as

$$U^a(\boldsymbol{x}, \boldsymbol{a}^*, \boldsymbol{b}^*) = \max_{i \in [N]} \left\{ (1 - a_i^*) \left(p_i l_i + \sum\nolimits_{j \neq i} q_{ij} l_j \right) \right\}.$$

If $U^a(\boldsymbol{x}, \boldsymbol{a}^*, \boldsymbol{b}^*) = 0$, it is the trivial case which means that all users invest under the given strategy $\boldsymbol{x}$, i.e., $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \langle \mathbf{1}, * \rangle$ is an NE where $*$ means the attacker's strategy can be any valid pure strategy because all pure strategies bring the same utility to the attacker, i.e., 0. We can easily check whether the strategy $\boldsymbol{x}$ can make $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \langle \mathbf{1}, * \rangle$ as an NE or not. For the case where $U^a(\boldsymbol{x}, \boldsymbol{a}^*, \boldsymbol{b}^*) > 0$, suppose that $b_k^* = 1$ for some $k \in [N]$, which implies that $a_k^* = 0$, i.e., user k does not invest. As $a_k^* = 0$ is user k's best-response to the attacker, we have $c_k \geq x_k + p_k l_k$. As we have $x_k \geq 0$, we obtain $c_k \geq p_k l_k$, which contradicts our assumption $c_i < p_i l_i$. □

The case that all users invest occurs when the budget is high, which is not the case in general. Therefore, we then focus on the mixed NE between followers. With a slight abuse of notation, we denote z_i as the probability of attacking i. Therefore, we can rewrite the user's utility as

$$\begin{aligned} U_i^u(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) &= y_i[(x_i - c_i) - \Psi_i^1 l_i] - (1 - y_i)[\Psi_i^2 p_i + \Psi_i^1] l_i \\ &= y_i(x_i - c_i) - \Psi_i^1 l_i - (1 - y_i)\Psi_i^2 p_i l_i \qquad (5.11) \\ &= y_i(x_i - c_i + z_i p_i l_i) - \Psi_i^1 l_i - \Psi_i^2 p_i l_i \qquad (5.12) \end{aligned}$$

where Ψ_i^1 and Ψ_i^2 are defined previously. The reduction from Eq.(5.11) to Eq.(5.12) is based on the key observation that $\Psi_i^2 = z_i$, so we can ignore the last two terms of Eq.(5.12) which are independent of the user's strategy. Then, we define that $\Delta_i = \frac{c_i - x_i}{p_i l_i}$ and $L_i = p_i l_i + \sum_{j \neq i} q_{ij} l_j$. We denote that $V(\boldsymbol{x}) = \{i | \Delta_i > 0, i \in [N]\}$ for a given $\boldsymbol{x}$ because if $\Delta_i \leq 0$, the user will definitely invest, so we can remove it from our game. Proposition 5.4 characterizes the set of mixed Nash equilibria between followers in the three cases where $\sum_{i \in V(\boldsymbol{x})} \Delta_i$ is less than , equal to and larger than 1, which plays a central role to compute the optimal strategy of the government.

Proposition 5.4. *Given $\boldsymbol{x}$, the strategy profile $\langle \boldsymbol{y}, \boldsymbol{z} \rangle$ is a mixed Nash equilibrium (MNE) in the game where*

1. $\sum_{i \in V(\boldsymbol{x})} \Delta_i < 1$, *if and only if* $\forall i \in V(\boldsymbol{x}), y_i = 1, z_i \geq \Delta_i$ *and* $\sum_{i \in [N]} z_i = 1$;
2. $\sum_{i \in V(\boldsymbol{x})} \Delta_i = 1$, *if and only if* $\forall i \in V(\boldsymbol{x}), y_i = 1 - L/L_i, 0 \leq L \leq \min_{i \in V(\boldsymbol{x})} L_i, z_i = \Delta_i$;
3. $\sum_{i \in V(\boldsymbol{x})} \Delta_i > 1$, *if and only if there is a non-singleton, nonempty subset* $I \subset V(\boldsymbol{x})$ *such that* $\min_{i \in I} L_i \geq \max_{j \in V(\boldsymbol{x}) \setminus I} L_j$ *and the followings hold: i)* $\forall i \in V(\boldsymbol{x}) \setminus I$, $y_i = z_i = 0$; *ii) let* $J = \arg\min_{i \in I} L_i$, $\forall i \in J$, $y_i = 0$, $0 \leq z_i \leq \Delta_i$; *iii)* $\forall i \in I \setminus J$, $y_i = 1 - \min_{j \in I} L_j / L_i, z_i = \Delta_i$.

Proof. We can rewrite the attacker's utility as

$$U^a(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \sum\nolimits_{i=1}^{N} z_i \left[(1 - y_i) \left(p_i l_i + \sum\nolimits_{j \neq i} q_{ij} l_j \right) \right] \tag{5.13}$$

The key observation is $\Psi_i^1 = \sum_{j \neq i} z_j (1 - y_j) q_{ji}$. Then the proof can be adapted from the proof of Proposition 5 in [76] by setting the attack cost to zero. Generally speaking, the proof of the proposition is based on three facts: i) when $z_i = \Delta_i$, the users will be indifferent to the investment which is observed from Eq.(5.12), ii) under any of the equilibria, the values $(1 - y_i) \left(p_i l_i + \sum_{j \neq i} q_{ij} l_j \right)$ are equal for all users with $z_i > 0$ and iii) for users with $z_i = 0, i \in V(\boldsymbol{x}), y_i = 0$. □

Proposition 5.5. *As the followers break ties in favor of the government, for the game where* $\sum_{i \in V(\boldsymbol{x})} \Delta_i \leq 1$, *the users will fully invest, i.e.,* $\boldsymbol{y} = \mathbf{1}$, *and the attacker will choose* $z_i \geq \Delta_i, \forall i \in V(\boldsymbol{x})$ *and* $z_i, i \in [N] \setminus V(\boldsymbol{x})$ *are irrelevant.*

Proof. According to Proposition 5.4, when $\sum_{i \in V(\boldsymbol{x})} \Delta_i < 1$, the only NE strategy for users is $\boldsymbol{y} = \mathbf{1}$ and the attacker's utility is 0 for all NE strategies of the attacker. For the case where $\sum_{i \in V(\boldsymbol{x})} \Delta_i = 1$, the only NE strategy for the attacker is $z_i, i \in [N] \setminus V(\boldsymbol{x})$. As we suppose the users will break ties in favor of the government, i.e., minimize the attacker's utility. Therefore, the users will choose $\boldsymbol{y} = \mathbf{1}$ as their NE strategy, which leads to the case where the attacker's utility is 0. Thus, for the game where

$\sum_{i\in V(\boldsymbol{x})} \Delta_i \leq 1$, the users will fully invest, i.e., $\boldsymbol{y} = \mathbf{1}$, and the attacker will choose $z_i \geq \Delta_i, \forall i \in V(\boldsymbol{x})$ and $z_i, i \in [N] \setminus V(\boldsymbol{x})$ are irrelevant. □

Given Proposition 5.5, we find that if $\sum_{i\in V(\boldsymbol{x})} \Delta_i \leq 1$, the users will always invest. And for the case where $\sum_{i\in V(\boldsymbol{x})} \Delta_i > 1$, Proposition 5.6 proves that all MNE can be computed in polynomial time in this case.

Proposition 5.6. *For the game where $\sum_{i\in V(\boldsymbol{x})} \Delta_i > 1$, given $\boldsymbol{x}$, all MNE can be computed in polynomial time.*

Proof. The algorithm can be adapted from Theorem 1 in [76] by setting the attack cost to zero. Roughly speaking, the algorithm sorts users in descending order according to L_i and then finds t such that $1 - \Delta_{idx_1(t)} \leq \sum_{j=1}^{t-1} \Delta_{idx_1(j)} < 1$, where $idx_1(\cdot)$ and $val_1(\cdot)$ are the index and the value in the sorted list. We denote $I = \{idx_1(k) | k \leq t\}$ and specify followers' strategies by Proposition 5.4. □

Algorithm 6: Compute government's optimal strategy

$\boldsymbol{x} = \mathbf{0}, t = 0, t' = N$;
Sort L_i in descending order and let $val_1(i)$ and $idx_1(i)$ be the ith value and index in the sorted list, respectively;
Sort $p_i l_i$ in descending order and let $val_2(i)$ and $idx_2(i)$ be the ith value and index in the sorted list, respectively;
Find t' such that $1 - \Delta_{idx_1(t')} \leq \sum_{j=1}^{t-1} \Delta_{idx_1(j)} < 1$;
while $t' > t$ **do**
    $b = B, \boldsymbol{x} = 0, t = t'$;
    while ***true*** **do**
        $j \in \arg\min_{idx_1(i)\leq t, i\in V(\boldsymbol{x})} val_2(i)$;
        if $j ==$ ***null*** **then**
            break
        else if $b \geq c_j$ **then**
            $x_j = c_j$;$b = b - c_j$
        else
            $c_j = b$; **break**
    Find t' where $1 - \Delta_{idx_1(t')} \leq \sum_{j=1}^{t'-1} \Delta_{idx_1(j)} < 1$;
return $\boldsymbol{x}$

Given Proposition 5.6, we now prove that the government's optimal strategy can be computed in polynomial time. Denote $\Delta = \sum_{i=1}^{N} \Delta_i$. We first check whether the

government can make $\Delta \leq 1$ by greedily assigning c_i to users in the ascending order of $p_i l_i$. For the case where the government cannot make $\Delta \leq 1$, we can see from Proposition 5.6 that the government's utility is equal to $val_1(t)$ in the sorted list such that $1 - \Delta_{idx_1(t)} \leq \sum_{j=1}^{t-1} \Delta_{idx_1(j)} < 1$. Therefore, maximizing the government's utility is equivalent to minimizing $val_1(t)$, which is depicted in Algorithm 6.

Proposition 5.7. *Algorithm 6 computes an SMNE in polynomial time when the attacker has a single-attack capability.*

Proof. We first prove that the loop in lines 7-14 can return the optimal solution if we only consider the users with $idx_1(i) \leq t$. As the attacker will attack the users with probability Δ_i, when the government assigns the subsidy with the minimal $p_i l_i$, which is denoted by j, i.e., $j \in \arg\min_{idx_1(i) \leq t} \{p_i l_i\}$. Then, user j will definitely invest and the attacker will assign the amount Δ_j to users with $idx_1(i) > t$. Thus, the greedy assignment of the subsidy is optimal due to the fact that assigning the subsidy to user j will reduce the largest amount of the probability that the attacker allocates to users with $idx_1(i) \leq t$. Then, we prove that when $t' = t$, the assignment is globally optimal. Observing that $t' \geq t$ at line 15, if $t' = t$, there is no user with $idx_1(i) > t$ being attacked by the attacker, given the optimal result of the loop in lines 7-14, the returned solution is globally optimal and we terminate the algorithm. The two sorting processes take $O(N \log N)$. The two while loops in line 5 and 7 run at most N times. The finding minimum operation in line 8 needs at most N comparisons. Thus, the runtime is $O(N^3 + N \log N) = O(N^3)$. □

5.3.3 Multiple-user Attack: $1 < K < N$

We now discuss the case where the attacker can attack multiple users simultaneously. An argument similar to Proposition 5.3 can prove that there is no PNE between followers, so we focus on the mixed NE case. Unfortunately, there is no simple expression of the users' utility as Eq.(5.12) due to the nonlinearity of the expected utility against the selection of users in each pure strategy. Furthermore, the number of the attacker's pure strategies is C_N^K, which grows exponentially against the number of users and makes the problem more difficult than the $K = 1$ case.

As claimed in [76], the computation of MNE in general case is intractable. Therefore, we adopt *best-response-gradient dynamics* (BRGD) [121] to compute an ϵ-MNE and greedily assign the subsidy to users. The algorithm is presented in Algorithm 7.

BRGD initializes the users' strategies with 0.5 and the attacker's strategy uniformly with $\sum_{\boldsymbol{b}\in\mathcal{B}^K} z_{\boldsymbol{b}} = 1$. At each round, BRGD updates $y_i = y_i + \alpha(U_i^u(\boldsymbol{x}, \boldsymbol{y}_{-i}, 1, \boldsymbol{z}) - U_i^u(\boldsymbol{x}, \boldsymbol{y}_{-i}, 0, \boldsymbol{z}))$ and $z_b = z_b + \alpha(U^a(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{b}) - U^a(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}))$ where $\boldsymbol{y}_{-i}$ is the strategies of other users except i, α is the step size of BRGD and $U_i^u(\boldsymbol{x}, \boldsymbol{y}_{-i}, \sigma, \boldsymbol{z}), \sigma \in \{0, 1\}$ is user i's utility when i takes the pure strategy σ, given that the other users take $\boldsymbol{y}_{-i}$ and the attacker takes $\boldsymbol{z}$. The intuition of BRGD is if $U_i^u(\boldsymbol{x}, \boldsymbol{y}_{-i}, 1, \boldsymbol{z}) - U_i^u(\boldsymbol{x}, \boldsymbol{y}_{-i}, 0, \boldsymbol{z}) > 0$, the user will increase the probability of investing and decrease the probability otherwise and the attacker updates his strategy similarly. We define that for each user, $r_i = |\frac{U_i^u - U_i^u(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})}{U_i^u(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})}|$ where $U_i^u = \max\{U_i^u(\boldsymbol{x}, \boldsymbol{y}_{-i}, 1, \boldsymbol{z}), U_i^u(\boldsymbol{x}, \boldsymbol{y}_{-i}, 0, \boldsymbol{z})\}$ and $r_a = |\frac{U^a - U^a(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})}{U^a}|$ where $U^a = \max_{\boldsymbol{b}\in\mathcal{B}^K}\{U^a(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{b})\}$ for the attacker. BRGD will terminate when $r < \epsilon$ where $r = \max\{r_1, ..., r_N, r_a\}$ and the solution is ensured to be an ϵ-MNE [121]. By using BRGD to obtain ϵ-MNE, we greedily assign the budget to users: for each iteration (Lines 3-13), we assign the $\delta = B/D$ to the user who can increase the government's utility the most where D is the maximum number of the iterations. We note that this algorithm can also be applied to solve the mixed strategy case of all-user attacks.

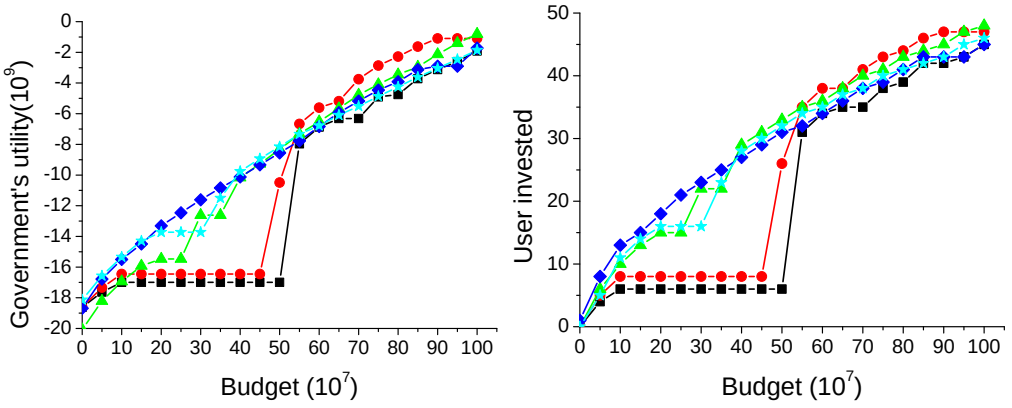
Algorithm 7: Allocation for multiple attacks

1 $\boldsymbol{x} = 0, \delta = B/D$;
2 $\langle \boldsymbol{y}, \boldsymbol{z} \rangle \leftarrow \texttt{BRGD}(\boldsymbol{x}, \epsilon)$;
3 **for** $d = 1 : D$ **do**
4 $\boldsymbol{icr} = \boldsymbol{0}$;
5 **for** $i = 1 : N$ **do**
6 $\boldsymbol{x}' = \boldsymbol{x}, x_i' = x_i + \delta$;
7 $\langle \boldsymbol{y}', \boldsymbol{z}' \rangle \leftarrow \texttt{BRGD}(\boldsymbol{x}')$;
8 $icr_i = U^d(\boldsymbol{x}', \boldsymbol{y}', \boldsymbol{z}') - U^d(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$;
9 $j = \arg\max_{i\in[N], icr_i>0} icr_i$;
10 **if** $j =$ ***null*** **then**
11 **break**
12 **else**
13 $x_j = x_j + \delta$
14 **return** $\boldsymbol{x}$

5.4 Experimental Evaluation

We evaluate the performance of our approaches through extensive experiments. All computations are performed on a 64-bit PC with 4.0 GB RAM and a 2.40 GHz CPU unless otherwise specified. The number of users varies in the range of $[10, 50]$, because only a few companies are critical for the national cybersecurity level. All games are randomly generated with $c_i = 10^7 + [0, 10^8]$, $l_i = 10^8 + [0, 10^9]$ and $p_i, q_{ji} \in [0, 1]$ (Similar parameter values are also used in [74]). As there is an exact polynomial algorithm for the single-user attack, we focus on the other two kinds of attacks in this section.

5.4.1 Results of All-user Attack for SPNE



(a) Government's utility (b) Number of users investing

FIGURE 5.3: Results of all-user attack. Best viewed in color.

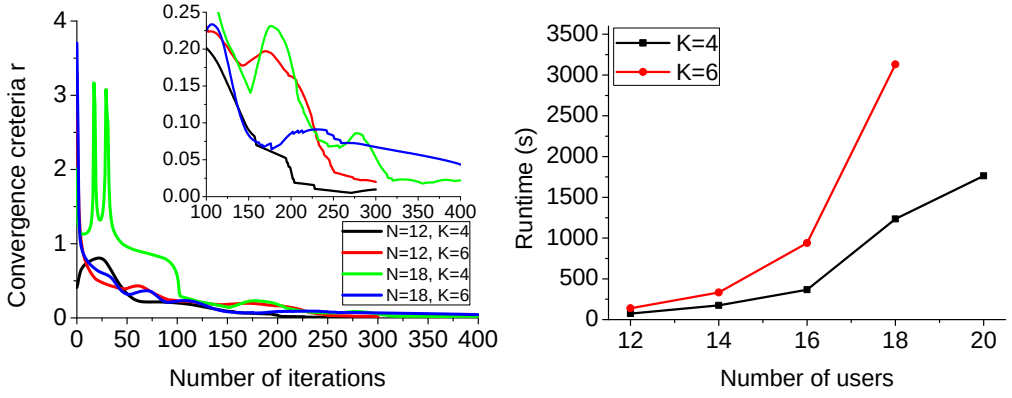
We use global nonlinear solvers to find global or local optimal solution of the problem. NEOS server [122] provides on-line solvers for non-linear programs and the global optimization solver BARON is used, which is based on branch and reduce algorithms.

Figure 5.3 displays five generated instances of 50 users with different government budget. BARON can compute the government's optimal strategy in 5 minutes on average. We observe that without subsidies only a few users invest. When the budget increases, more users invest. Figure 5.3a and 5.3b show the government's utility and the number of users who invest with various budget amounts, respectively. There are two observations from the results: i) the smooth transition where by adding more government budget, as showed in blue, green and cyan lines, the number of users who invest

increases smoothly, as well as the government's utility and ii) the phase transition where the added government budget makes many more users to invest (16 in red line and 25 in blue line) and the government's utility is also significantly increased. Note that there are smooth transitions before and after the phase transition (0-40 and 50-100 of both red and blue lines). For smooth transitions, when the budget is high, it needs more subsidy to incentivize the remaining users to invest, i.e., the return of adding the budget is diminishing. The conclusion is supported by more instances conducted. For the relation of the phase transitions and the parameters of the model, which is beyond the scope of this chapter and will be investigated in future works.

5.4.2 Results of Multiple-user Attack for SMNE

We now present the experimental results of the multiple-user attack. We assume that the attacker can attack at most half of the users. The convergence criteria is $\epsilon = 0.05$, the step size is $\alpha = 0.05/(K \cdot \max_{i\in[N]} l_i)$, the government's budget is $B = 5 \cdot 10^8$ and the number of iterations D is 10 (i.e., the government assigns $5 \cdot 10^7$ to a user at each iteration of Algorithm 7). To generate the synthetic data, we vary $K \in \{4, 6\}$ and $N \in \{12, 14, 16, 18, 20\}$. We also test our heuristics on the real network data and vary $K \in \{4, 6\}$.

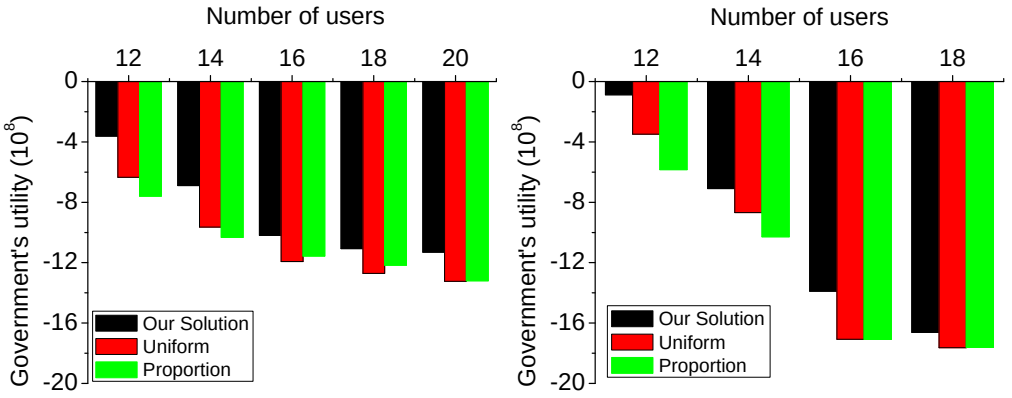


(a) Convergence of BRGD (b) Runtime

FIGURE 5.4: Convergence and runtime results of Algorithm 7.

Convergence and runtime. We first investigate the convergence of BRGD. Figure 5.4a shows the convergence results and, to make the figure readable, only 4 among all cases are depicted. Each case is averaged over 30 instances. All cases reach the

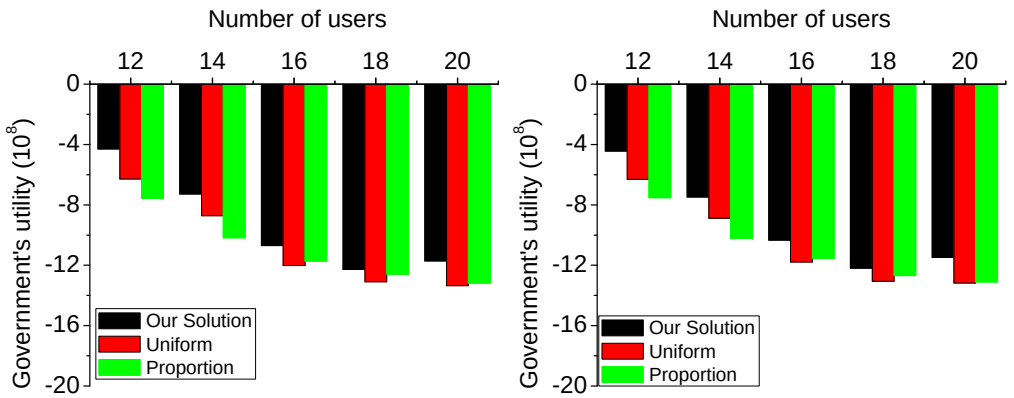
termination in less than 250 iterations, which is larger than the number of iterations reported in [74] because the users' utilities in our model are more dependent on their neighbors' strategies. An interesting phenomenon is that on average the convergence criteria will increase in the first 50 iterations and then decrease, which depends on the start points of BRGD. We also note that the number of iterations it needs to converge does not show a clear relation of the values of N and K, which influence the number of attacker's pure strategies. Furthermore, we displayed the runtime results of the algorithm with a cap of 3600 seconds in Figure 5.4b and note that when $N = 20$ and $K = 6$, the runtime of Algorithm 7 is beyond the cap and is not considered in the rest of this section. The results show that when K is larger, the algorithm takes more time before the termination due to the fact that the number of attacker's pure strategies grows exponentially with the number of K and it takes a longer time to make an iteration of BRGD to update the strategies of users and the attacker, though the number of iterations to converge does not increase much, as showed in Figure 5.4a.



(a) $K = 4$ (b) $K = 6$

FIGURE 5.5: Solution quality of Algorithm 7.

Solution quality. We compare our solution with two other baselines: **uniform**, where the government divides the subsidy to users uniformly, and **proportion**, where the government assigns the subsidy proportionally to the user's loss l_i. Figure 5.5 shows the solution quality with different values of K. The results show that our solution is better than the two baselines. Fixing $N = 16$, the advantage of our solution against the two baselines is reduced when K is smaller, i.e., users are more reluctant to invest in safer environments.



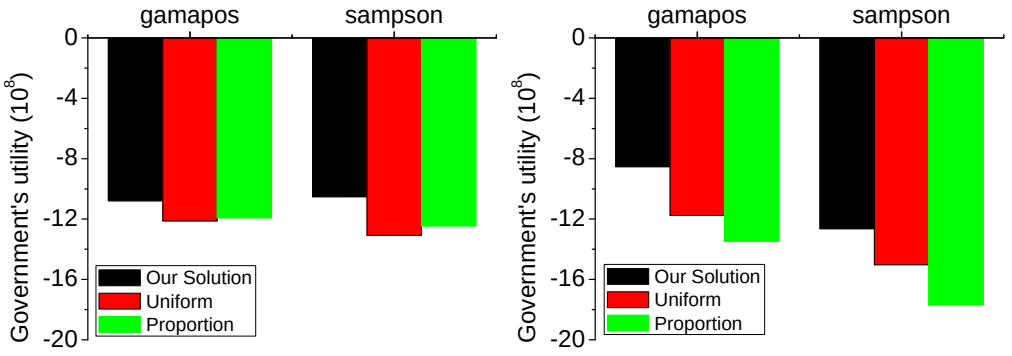
(a) Transfer probability (b) Number of K

FIGURE 5.6: Robustness of Algorithm 7 with $K = 4$.

Robustness. As there are various uncertainties in the real world, we then evaluate the robustness of our solution. We consider two kinds of uncertainties: i) the uncertainty of transfer probability due to the fact that it is difficult for the government to know the interdependence between users exactly, we suppose that $\tilde{q}_{ji} = (1 + \delta)q_{ji}, \delta \in [-10\%, 10\%]$ where $\tilde{q}_{ji}$ is the real transfer probability and q_{ji} is the transfer probability that the government uses to compute his strategy; and ii) the uncertainty of the number of K due to the fact that it is difficult for the users and the government to know the attacker's ability, we suppose that the real number of users $\tilde{K}$ the attacker can attack is either $K + 1$ or $K - 1$ with a probability of 10%, respectively, where K is the number used by the government to decide his strategy. Note that the attacker is assumed to know these parameters exactly, which is the worst case to the government's strategy. Figure 5.6 shows that our solution outperforms the two baselines against the two kinds of uncertainties considered. Note that we only display results for $K = 4$ and results for $K = 6$ are similar and omitted.

Experiments on real networks. To further evaluate Algorithm 7, we test it with $K = 4$ and $K = 6$ on two real networks, given that business networks are often generated socially,: i) **gamapos** with 16 nodes and 58 edges and ii) **sampson** with 18 nodes and 55 edges[1]. The results are showed in Figure 5.7 where our solution still outperforms the two baselines considered. The results support the conclusion that when K is small,

[1]Both networks are real human social networks and can be found at http://konect.uni-koblenz.de/networks/.



(a) $K = 4$ (b) $K = 6$

FIGURE 5.7: Experiment results on real network data.

the advantage of our solution against the two baselines are reduced. We will apply our algorithm to more and larger networks in the future work.

5.5 Chapter Summary

This chapter focuses on the problem of preventing cyber attacks on interdependent users through government's subsidy. We propose a novel Stackelberg cybersecurity investment game to model interactions between the government, interdependent users and an attacker, where the government moves first to assign the subsidy to users and then both users and the attacker play simultaneously after observing the government's strategy. We investigate three cases where the attacker can at-tack all, single and multiple users and propose a reverse con-vex formulation, an exact polynomial algorithm and a heuristic algorithm for the three cases, respectively. Experimentally, we show that our heuristic algorithm is effective and outperforms baselines on synthetic and real data.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we follow the Stackelberg security game methodologies and extend them to some important security domains with structures such as networks or grids. We provide models and computational methods of these scenarios and experimentally demonstrate the effectiveness of our methods.

The first part of this thesis studies the problem of preventing nuclear smuggling on container shipping network through efficient container inspection. We introduce a novel container inspection model (CIM) between the inspector and the smuggler, where the smuggler's decision making process is modeled as an MDP. We develop several key approaches to compute the efficient strategy of the inspector, which includes a linear relaxation approximation to reformulate the multi-linear optimization into a bilinear formulation, a novel approach based on MDT to solve the bilinear formulation exactly (within a given accuracy) and a state and action generation method, which incrementally adds the states and action of the smuggler's MDP into consideration to improve the scalability. Extensive experiments show that our algorithms outperform existing methods in the scalability significantly and can obtain a robust solution better than baselines.

The second part of this thesis tries to address the oil siphoning problem by efficiently assigning patrol resources. We propose a novel Stackelberg model where both

the defender and the attacker take paths as their strategies and the externalities are also included. We propose a compact representation for the defender's strategy and a constraint generation algorithm to incrementally add the attack's strategy into consideration. To further improve the scalability, we propose an abstraction method by exploring the structural properties of the patrol area. Extensive experimental results demonstrate that our approaches can result in dramatic improvement of scalability with modest influence on the solution quality and can scale up to realistic-sized problems.

The third part of this thesis focuses on the problem of preventing cyber attacks on interdependent users through government's subsidy. We propose a novel Stackelberg cybersecurity investment game between the government, interdependent users and a cyber attacker. We comprehensively investigate the three cases where the attacker can attack all, single and multiple users and propose a reverse convex formulation, an exact polynomial algorithm and a heuristic algorithm for the three cases, respectively. We extensively evaluate our model and algorithms on synthetic and real data. The results show that our model captures interdependent behaviors of users in all-user attack and our heuristic algorithm converges efficiently under mild convergence criteria and outperforms baselines with good robustness against uncertainties in multiple-user attack.

6.2 Future Work

In this section, we discuss the two future directions related to the works presented in this thesis. The first direction is to apply stochastic game to Stackelberg security game. Most of previous works in Stackelberg security games assume that the targets are static, i.e., both values and positions of the targets are determined and known by the defender and the attacker [6, 9]. In this case, the defender will commit to an optimal mixed strategy to protect the targets and the pure strategies to be implemented are sampled from the mixed strategy when deployed to the real world. However, in many wildlife and forest conservation domains, the adversary can build support facilities (e.g., set up camps or build roads to facilitate smuggling) for executing illegal activities to obtain a higher utility, which would change the values of targets in these domains. The values of targets are defined as the payoffs of the adversary when the illegal activities are

successfully executed at the targets, e.g., setting up a camp would allow poachers to kill more endangered wildlife in the target zone. However, the defender may not know these changes unless the target is visited by the defender. Some works consider the case where the targets' values are not known by the defender and learning algorithms are proposed [123, 124]. However, in these works, the targets' values are static where the algorithm can learn the optimal defender's strategy offline. There are several works which consider the case that the targets dynamically change their values [125] or positions [126] over a finite time period. Both changes are deterministic and known by the players, however, none of them considers the case where the target can be affected by the players' actions, therefore the previous methods cannot be directly applied to our problem. To apply the game theoretic methodology to security domains where the targets' values are affected by players' actions, a novel partially observable stochastic Stackelberg security game where the defender can only observe the targets' values being protected and the attacker can fully observe the game can be proposed.

The second direction is that the correlated equilibrium can be a more suitable solution concept between the cyber users in the cyber security scenarios [127], instead of considering the Nash equilibrium. There are several advantages of adopting the correlate equilibrium in these scenarios: first, the correlated equilibrium assumes that there is a message sender who can coordinate the users, which is more realistic because the government can design his policy to regulate the cyber space in the real world; second, it is difficult for users in cyber space to know the connections of others users, which brings difficulties for users to find their Nash equilibrium strategies by themselves, however, the government can learn this through extensive investigations, and third, from the computational perspective, the correlated equilibrium can be computed by linear programs, however, computing a Nash equilibrium in general-sum games is extremely difficult. Therefore, we can consider the model between the government and interdependent users where the government moves first and recommends the correlated equilibrium with the highest social welfare to the users. Further more, we can add cyber attackers into the model and extend the model to consider the multi-hop prorogation of the cyber attacks such as independent cascade model and linear threshold model [128, 129].

Bibliography

[1] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *EC*, pages 82–90, 2006.

[2] Milind Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge university press, 2011.

[3] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Deployed ARMOR protection: The application of a game theoretic model for security at the Los Angeles international airport. In *AAMAS*, pages 125–132, 2008.

[4] Bo An, Eric Shieh, Milind Tambe, Rong Yang, Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrett Meyer. PROTECT – A deployed game theoretic system for strategic security allocation for the United States coast guard. *AI Magazine*, 33(4):96–110, 2012.

[5] Fei Fang, Thanh H Nguyen, Rob Pickles, Wai Y Lam, Gopalasamy R Clements, Bo An, Amandeep Singh, Milind Tambe, and Andrew Lemieux. Deploying PAWS: Field optimization of the protection assistant for wildlife security. In *IAAI*, pages 3966–3973, 2016.

[6] Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. Playing games for security: An efficient exact algorithm for solving bayesian Stackelberg games. In *AAMAS*, pages 895–902, 2008.

[7] Jason Tsai, Zhengyu Yin, Jun-young Kwak, David Kempe, Christopher Kiekintveld, and Milind Tambe. Urban security: Game-theoretic resource allocation in networked domains. In *AAAI*, pages 881–886, 2010.

[8] Qingyu Guo, Bo An, Yair Zick, and Chunyan Miao. Optimal interdiction of illegal network flow. In *IJCAI*, pages 2507–2513, 2016.

[9] Jiarui Gan, Bo An, Yevgeniy Vorobeychik, and Brian Gauch. Security games on a plane. In *AAAI*, pages 530–536, 2017.

[10] Rong Yang, Benjamin Ford, Milind Tambe, and Andrew Lemieux. Adaptive resource allocation for wildlife protection against illegal poachers. In *AAMAS*, pages 453–460, 2014.

[11] Arunesh Sinha, Thanh H Nguyen, Debarun Kar, Matthew Brown, Milind Tambe, and Albert Xin Jiang. From physical security to cybersecurity. *Journal of Cybersecurity*, 1(1):19–35, 2015.

[12] Ondřej Vaněk, Zhengyu Yin, Manish Jain, Branislav Bošanskỳ, Milind Tambe, and Michal Pěchouček. Game-theoretic resource allocation for malicious packet detection in computer networks. In *AAMAS*, pages 905–912, 2012.

[13] Karel Durkota, Viliam Lisy, Christofer Kiekintveld, and Branislav Bosansky. Game-theoretic algorithms for optimal network security hardening using attack graphs. In *AAMAS*, pages 1773–1774, 2015.

[14] Aaron Schlenker, Haifeng Xu, Mina Guirguis, Chris Kiekintveld, Arunesh Sinha, Milind Tambe, Solomon Sonya, Darryl Balderas, and Noah Dunstatter. Don't bury your head in warnings: A game-theoretic approach for intelligent allocation of cyber-security alerts. In *IJCAI*, pages 381–387, 2017.

[15] Zhen Wang, Yue Yin, and Bo An. Computing optimal monitoring strategy for detecting terrorist plots. In *AAAI*, pages 637–643, 2016.

[16] CBP. *Container Security Initiative: In Summary*. U. S. Customs and Border Protection, 2011.

[17] NNSA. Megaports Initiative. https://nnsa.energy.gov/aboutus/ourprograms/nonproliferation/programoffices/internationalmaterialprotectionandcooperation/-5, 2015. Last accessed on 30 June 2019.

[18] DHS. Secure Freight Initiative. http://www.dhs.gov/secure-freight-initiative, 2015. Last accessed on 30 June 2019.

[19] Timothy J. Leonard, Philip Gallo, and Simon Véronneau. Security challenges in United States sea ports: An overview. *Journal of Transportation Security*, 8 (1-2):41–49, 2015.

[20] Endre Boros, L Fedzhora, PB Kantor, K Saeger, and P Stroud. A large-scale linear programming model for finding optimal container inspection strategies. *Naval Research Logistics (NRL)*, 56(5):404–420, 2009.

[21] Jose Emmanuel Ramirez-Marquez. Port-of-entry safety via the reliability optimization of container inspection strategy through an evolutionary approach. *Reliability Engineering & System Safety*, 93(11):1698–1709, 2008.

[22] Nitin Bakshi, Stephen E Flynn, and Noah Gans. Estimating the operational impact of container inspections at international ports. *Management Science*, 57(1): 1–20, 2011.

[23] Niyazi Onur Bakır. A stackelberg game model for resource allocation in cargo container security. *Annals of Operations Research*, 187(1):5–22, 2011.

[24] Vicki M. Bier and Naraphorn Haphuriwat. Analytical method to identify the number of containers to inspect at US ports to deter terrorist attacks. *Annals of Operations Research*, 187(1):137–158, 2011.

[25] Naraphorn Haphuriwat, Vicki M Bier, and Henry H Willis. Deterring the smuggling of nuclear weapons in container freight through detection and retaliation. *Decision Analysis*, 8(2):88–102, 2011.

[26] João P Teles, Pedro M Castro, and Henrique A Matos. Global optimization of water networks design using multiparametric disaggregation. *Computers & Chemical Engineering*, 40:132–147, 2012.

[27] Ted Kemp. Crime on the high seas: The world's most pirated waters. *CNBC*, 2015. URL http://www.cnbc.com/2014/09/15/worlds-most-pirated-waters.html.

[28] Anna Bowden, Kaija Hurlburt, Eamon Aloyo, Charles Marts, and Andrew Lee. *The Economic Costs of Maritime Piracy*. One Earth Future Foundation, 2010.

[29] Jianyue Xue. Singapore, Malaysia and Indonesia could extend joint patrols in South China Sea. http://www.channelnewsasia.com/news/singapore/singapore-malaysia-and/1838338.html, 2015. Last accessed on 30 June 2019.

[30] Kaspersky. Kaspersky lab survey: Cyberattacks cost large businesses in north america an average of $1.3m. https://usa.kaspersky.com/about/press-releases/2017_kaspersky-lab-survey-cost-of-cyberattacks-for-large-businesses-in-north-america, 2017. Last accessed on 30 June 2019.

[31] CBS. Wannacry ransomware attack losses could reach $4 billion. https://www.cbsnews.com/news/wannacry-ransomware-attacks-wannacry-virus-losses/, 2017. Last accessed on 30 June 2019.

[32] Thycotic. 2017 state of cyber security metrics annual report. https://thycotic.com/wp-content/uploads/2013/03/2017-Cyber-Security-Strategy-Executive-Summary.pdf, 2017. Last accessed on 30 June 2019.

[33] UK. Cyber essentials. https://www.cyberessentials.ncsc.gov.uk/, 2014. Last accessed on 30 June 2019.

[34] UK. *National Cyber Security Strategy 2016-2021*. Government of the United Kingdom, 2016.

[35] Martin J Osborne et al. *An Introduction to Game Theory*, volume 3. Oxford university press New York, 2004.

[36] Roger B Myerson. *Game theory*. Harvard university press, 2013.

[37] John Nash. Non-cooperative games. *Annals of Mathematics*, pages 286–295, 1951.

[38] Heinrich Von Stackelberg. *The Theory of the Market Economy*. Oxford University Press, 1952.

[39] Bernhard von Stengel and Shmuel Zamir. Leadership with commitment to mixed strategies. *Games and Economic Behavior*, 69(2):446–457, 2010.

[40] H Brendan McMahan, Geoffrey J Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *ICML*, pages 536–543, 2003.

[41] Yue Yin and Bo An. Efficient resource allocation for protecting coral reef ecosystems. In *IJCAI*, pages 531–537, 2016.

[42] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. The complexity of computing a nash equilibrium. *SIAM Journal on Computing*, 39 (1):195–259, 2009.

[43] Carlton E Lemke and Joseph T Howson, Jr. Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics*, 12(2): 413–423, 1964.

[44] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic game theory*. Cambridge university press, 2007.

[45] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. Computing optimal randomized resource allocations for massive security games. In *AAMAS*, pages 689–696, 2009.

[46] Manish Jain, Jason Tsai, James Pita, Christopher Kiekintveld, Shyamsunder Rathi, Milind Tambe, and Fernando Ordónez. Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces*, 40(4):267–290, 2010.

[47] Jason Tsai, Christopher Kiekintveld, Fernando Ordonez, Milind Tambe, and Shyamsunder Rathi. IRIS-A tool for strategic security allocation in transportation networks. In *AAMAS*, pages 37–44, 2009.

[48] James Pita, Milind Tambe, Chris Kiekintveld, Shane Cullen, and Erin Steigerwald. Guards: game theoretic security allocation on a national scale. In *AAMAS*, pages 37–44, 2011.

[49] James Pita, Manish Jain, Fernando Ordóñez, Milind Tambe, Sarit Kraus, and Reuma Magori-Cohen. Effective solutions for real-world stackelberg games: When agents must deal with human uncertainties. In *AAMAS*, pages 369–376, 2009.

[50] 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.

[51] Bo An, Matthew Brown, Yevgeniy Vorobeychik, and Milind Tambe. Security games with surveillance cost and optimal timing of attack execution. In *AAMAS*, pages 223–230, 2013.

[52] Albert Xin Jiang, Zhengyu Yin, Chao Zhang, Milind Tambe, and Sarit Kraus. Game-theoretic randomization for security patrolling with dynamic execution uncertainty. In *AAMAS*, pages 207–214, 2013.

[53] Zhengyu Yin, Albert Xin Jiang, Matthew P Johnson, Christopher Kiekintveld, Kevin Leyton-Brown, Tuomas Sandholm, Milind Tambe, and John P Sullivan. TRUSTS: Scheduling randomized patrols for fare inspection in transit systems. In *IAAI*, pages 2348–2355, 2012.

[54] Daniel Kahneman. Prospect theory: An analysis of decisions under risk. *Econometrica*, 47(2):263–292, 1979.

[55] Richard D McKelvey and Thomas R Palfrey. Quantal response equilibria for normal form games. *Games and Economic Behavior*, 10(1):6–38, 1995.

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

[57] Rong Yang, Fernando Ordonez, and Milind Tambe. Computing optimal strategy against quantal response in security games. In *AAMAS*, pages 847–854, 2012.

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

[59] Fei Fang, Peter Stone, and Milind Tambe. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *IJCAI*, pages 2589–2595, 2015.

[60] Richard L Church, Maria P Scaparra, and Richard S Middleton. Identifying critical infrastructure: the median and covering facility interdiction problems. *Annals of the Association of American Geographers*, 94(3):491–502, 2004.

[61] Douglas S Altner, Özlem Ergun, and Nelson A Uhan. The maximum flow network interdiction problem: Valid inequalities, integrality gaps, and approximability. *Operations Research Letters*, 38(1):33–38, 2010.

[62] Michael O Ball, Bruce L Golden, and Rakesh V Vohra. Finding the most vital arcs in a network. *Operations Research Letters*, 8(2):73–76, 1989.

[63] Eitan Israeli and R Kevin Wood. Shortest-path network interdiction. *Networks*, 40(2):97–111, 2002.

[64] R Kevin Wood. Deterministic network interdiction. *Mathematical and Computer Modelling*, 17(2):1–18, 1993.

[65] F Pan, W Charlton, and DP Morton. Stochastic network interdiction of nuclear material smuggling. *Network Interdiction and Stochastic Integer Programming*, pages 1–19, 2001.

[66] Manish Jain, Dmytro Korzhyk, Ondřej Vaněk, Vincent Conitzer, Michal Pěchouček, and Milind Tambe. A double oracle algorithm for zero-sum security games on graphs. In *AAMAS*, pages 327–334, 2011.

[67] Nicola Basilico and Nicola Gatti. Automated abstractions for patrolling security games. In *AAAI*, pages 1096–1101, 2011.

[68] Anjon Basak, Fei Fang, Thanh Hong Nguyen, and Christopher Kiekintveld. Abstraction methods for solving graph-based security games. In *AAMAS*, pages 13–33, 2016.

[69] Chao Zhang, Victor Bucarey, Ayan Mukhopadhyay, Arunesh Sinha, Yundi Qian, Yevgeniy Vorobeychik, and Milind Tambe. Using abstractions to solve opportunistic crime security games at scale. In *AAMAS*, pages 196–204, 2016.

[70] Howard Kunreuther and Geoffrey Heal. Interdependent security. *Journal of Risk and Uncertainty*, 26(2-3):231–249, 2003.

[71] Geoffrey Heal and Howard Kunreuther. Ids models of airline security. *Journal of Conflict Resolution*, 49(2):201–217, 2005.

[72] Konstantinos G Gkonis and Harilaos N Psaraftis. Container transportation as an interdependent security problem. *Journal of Transportation Security*, 3(4): 197–211, 2010.

[73] Michael Kearns and Luis E Ortiz. Algorithms for interdependent security games. In *NIPS*, pages 561–568, 2004.

[74] Hau Chan, Michael Ceyko, and Luis E. Ortiz. Interdependent defense games: Modeling interdependent security under deliberate attacks. In *UAI*, pages 152–162, 2012.

[75] Hau Chan and Luis E Ortiz. Computing nash equilibria in generalized interdependent security games. In *NIPS*, pages 2735–2743, 2014.

[76] Hau Chan, Michael Ceyko, and Luis Ortiz. Interdependent defense games with applications to internet security at the level of autonomous systems. *Games*, 8 (1):13, 2017.

[77] Clifford A Meyer and Christodoulos A Floudas. Global optimization of a combinatorially complex generalized pooling problem. *AIChE journal*, 52(3):1027–1037, 2006.

[78] Ruth Misener and Christodoulos A Floudas. Advances for the pooling problem: Modeling, global optimization, and computational studies. *Applied and Computational Mathematics*, 8(1):3–22, 2009.

[79] Artyom Nahapetyan and Panos M Pardalos. A bilinear relaxation based algorithm for concave piecewise linear network flow problems. *Journal of Industrial and Management Optimization*, 3(1):71, 2007.

[80] Artyom G Nahapetyan and Panos M Pardalos. A bilinear reduction based algorithm for solving capacitated multi-item dynamic pricing problems. *Computers & Operations Research*, 35(5):1601–1612, 2008.

[81] Faiz A Al-Khayyal and James E Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273–286, 1983.

[82] Leslie Richard Foulds, Dag Haugland, and Kurt Jørnsten. A bilinear approach to the pooling problem. *Optimization*, 24(1-2):165–180, 1992.

[83] Kurt M Anstreicher. On convex relaxations for quadratically constrained quadratic programming. *Mathematical programming*, 136(2):233–251, 2012.

[84] Nilanjan Adhya, Mohit Tawarmalani, and Nikolaos V Sahinidis. A lagrangian approach to the pooling problem. *Industrial & Engineering Chemistry Research*, 38(5):1956–1972, 1999.

[85] Pietro Belotti, Jon Lee, Leo Liberti, Francois Margot, and Andreas Wächter. Branching and bounds tighteningtechniques for non-convex minlp. *Optimization Methods & Software*, 24(4-5):597–634, 2009.

[86] Mu Avriel and AC Williams. Complementary geometric programming. *SIAM Journal on Applied Mathematics*, 19(1):125–141, 1970.

[87] Hoang Tuy. Convex programs with an additional reverse convex constraint. *Journal of Optimization Theory and Applications*, 52(3):463–486, 1987.

[88] Reiner Horst and Hoang Tuy. *Global Optimization: Deterministic Approaches*. Springer Science & Business Media, 2013.

[89] Mengchen Zhao, Bo An, and Christopher Kiekintveld. Optimizing personalized email filtering thresholds to mitigate sequential spear phishing attacks. In *AAAI*, pages 658–664, 2016.

[90] GAO. *Combating Nuclear Smuggling: DHS Research and Development on Radiation Detection Technology Could Be Strengthened*. United States Government Accountability Office, 2015.

[91] GAO. *Combating Nuclear Smuggling: Megaports Initiative Faces Funding and Sustainability Challenges*. United States Government Accountability Office, 2012.

[92] NTI. Illicit trafficking in weapons-useable nuclear material: Still more questions than answers. http://www.nti.org/analysis/articles/illicit-trafficking-weapons-useable-nuclear-material-still-more-questions-answers/, 2011. Last accessed on 30 June 2019.

[93] George Leitmann. On generalized Stackelberg strategies. *Journal of Optimization Theory and Applications*, 26(4):637–643, 1978.

[94] Paul J. Schweitzer and Abraham Seidmann. Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110(2):568–582, 1985.

[95] Garth P McCormick. Computability of global solutions to factorable nonconvex programs: Part I - convex underestimating problems. *Mathematical programming*, 10(1):147–175, 1976.

[96] Pedro M. Castro. Normalized multiparametric disaggregation: An efficient relaxation for mixed-integer bilinear problems. *Journal of Global Optimization*, 64 (4):765–784, 2016.

[97] ReCAAP. Annual report - piracy and armed robbery against ships in Asia. http://www.recaap.org/DesktopModules/Bring2mind/DMX/Download.aspx?Command=Core_Download&EntryId=421&PortalId=0&TabId=78, 2015. Last accessed on 30 June 2019.

[98] ReCAAP. Incident update siphoning of fuel/oil from Orkim Victory. http://www.recaap.org/Portals/0/docs/Latest%20IA/2015/Incident%20Update%20Orkim%20Victory%20(4%20Jun%2015).pdf, 2015. Last accessed on 30 June 2019.

[99] Nicola Basilico, Nicola Gatti, and Francesco Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS*, pages 57–64, 2009.

[100] Youzhi Zhang, Bo An, Long Tran-Thanh, Zhen Wang, Jiarui Gan, and Nicholas R Jennings. Optimal escape interdiction on transportation networks. In *IJCAI*, pages 3936–3944, 2017.

[101] Jiarui Gan, Bo An, and Yevgeniy Vorobeychik. Security games with protection externalities. In *AAAI*, pages 914–920, 2015.

[102] Branislav Bošanskỳ, Viliam Lisỳ, Michal Jakob, and Michal Pěchouček. Computing time-dependent policies for patrolling games with mobile targets. In *AAMAS*, pages 989–996, 2011.

[103] Michal Jakob, Ondřej Vaněk, and Michal Pechoucek. Using agents to improve international maritime transport security. *IEEE Intelligent Systems*, 26(1):90–96, 2011.

[104] Ondřej Vaněk, Michal Jakob, Ondřej Hrstka, and Michal Pěchouček. Agent-based model of maritime traffic in piracy-affected waters. *Transportation Research Part C: Emerging Technologies*, 36:157–176, 2013.

[105] ReCAAP. Half yearly report - piracy and armed robbery against ships in Asia. http://www.neptunemaritimesecurity.com/imb-recaap-isc-release-2015-half-yearly-reports-on-piracy-armed-robbery-at-sea/, 2015. Last accessed on 30 June 2019.

[106] DAMEN. Stan Patrol 3007 - Executive Summary. http://products.damen.com/en/ranges/stan-patrol/stan-patrol-3007, 2016. Last accessed on 30 June 2019.

[107] UK. *National Security Strategy and Strategic Defence and Security Review 2015*. Government of the United Kingdom, 2015.

[108] Aron Laszka, Mark Felegyhazi, and Levente Buttyan. A survey of interdependent information security games. *ACM Computing Surveys (CSUR)*, 47(2):23, 2015.

[109] Hau Chan and Luis E Ortiz. Computing nash equilibrium in interdependent defense games. In *AAAI*, pages 842–850, 2015.

[110] Lawrence A Gordon and Martin P Loeb. The economics of information security investment. *ACM Transactions on Information and System Security (TISSEC)*, 5 (4):438–457, 2002.

[111] Lawrence A Gordon, Martin P Loeb, and William Lucyshyn. Sharing information on computer systems security: An economic analysis. *Journal of Accounting and Public Policy*, 22(6):461–485, 2003.

[112] Libin Jiang, Venkat Anantharam, and Jean Walrand. How bad are selfish investments in network security? *IEEE/ACM Transactions on Networking (TON)*, 19 (2):549–560, 2011.

[113] Rainer Böhme, Galina Schwartz, et al. Modeling cyber-insurance: towards a unifying framework. In *WEIS*, 2010.

[114] Jens Grossklags, Svetlana Radosavac, Alvaro A Cárdenas, and John Chuang. Nudge: Intermediaries' role in interdependent network security. In *TRUST*, pages 323–336, 2010.

[115] Jasmina Omic, Ariel Orda, and Piet Van Mieghem. Protecting against network infections: A game theoretic perspective. In *INFOCOM*, pages 1485–1493, 2009.

[116] Aaron Schlenker, Omkar Thakoor, Haifeng Xu, Fei Fang, Milind Tambe, Long Tran-Thanh, Phebe Vayanos, and Yevgeniy Vorobeychik. Deceiving cyber adversaries: A game theoretic approach. In *AAMAS*, pages 892–900, 2018.

[117] Yevgeniy Vorobeychik and Joshua Letchford. Securing interdependent assets. *Autonomous Agents and Multi-Agent Systems*, 29(2):305–333, 2015.

[118] Jiarui Gan, Edith Elkind, and Michael Wooldridge. Stackelberg security games with multiple uncoordinated defenders. In *AAMAS*, pages 703–711, 2018.

[119] Nicola Basilico, Stefano Coniglio, and Nicola Gatti. Methods for finding leader-follower equilibria with multiple followers. In *AAMAS*, pages 1363–1364, 2016.

[120] Stefano Coniglio, Nicola Gatti, and Alberto Marchesi. Pessimistic leader-follower equilibria with multiple followers. In *IJCAI*, pages 171–177, 2017.

[121] Drew Fudenberg and David K Levine. *The Theory of Learning in Games*, volume 2. MIT press, 1998.

[122] Joseph Czyzyk, Michael P Mesnier, and Jorge J Moré. The NEOS server. *IEEE Computational Science and Engineering*, 5(3):68–75, 1998.

[123] Joshua Letchford, Vincent Conitzer, and Kamesh Munagala. Learning and approximating the optimal strategy to commit to. In *International Symposium on Algorithmic Game Theory*, pages 250–262, 2009.

[124] Avrim Blum, Nika Haghtalab, and Ariel D Procaccia. Learning optimal commitment to overcome insecurity. In *NIPS*, pages 1826–1834, 2014.

[125] Yue Yin, Bo An, and Manish Jain. Game-theoretic resource allocation for protecting large public events. In *AAAI*, pages 826–834, 2014.

[126] Fei Fang, Albert Xin Jiang, and Milind Tambe. Protecting moving targets with multiple mobile resources. *Journal of Artificial Intelligence Research*, 48:583–634, 2013.

[127] Sham Kakade, Michael Kearns, John Langford, and Luis Ortiz. Correlated equilibria in graphical games. In *EC*, pages 42–47, 2003.

[128] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, 2003.

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