HIERARCHICAL PLANNING IN SECURITY GAMES; A GAME THEORETIC APPROACH TO STRATEGIC, TACTICAL AND OPERATIONAL DECISION MAKING.

by

Sara Marie MC CARTHY

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

August 2018

Contents

List of Figures

List of Tables

Acknowledgements

There will never be any way that I can properly express how I feel about my time in graduate school, or my thanks to all those who have helped me through. This thesis captures such a small part of all that I've learned and done throughout what has been a long journey of success and failures, and of self growth and actualization. It seems unfair to be confined to these short pages, but I hope that all those who have shared this experience with me know what an important mark they will have forever left on my life.

I first want to thank my advisor and my mentor Milind Tambe, for not only his guidance, but his patience throughout these years. Your willingness to go the extra mile for your students is incredible and I will be forever grateful for all the time and effort you have dedicated to helping me not only complete a successful PhD but for helping me decide who I wanted to be after. I also want to thank Phebe Vayanos for her invaluable help in shaping my research as well Eric Rice, Jonathan Gratch and Jelena Mirkovic for all their time and feedback as members of my committee.

Thanks to all of my Teamcore family for being a part of this journey with me. Aaron Schlenker, thank you for being with me from the beginning, through all the highs and

lows. Ben Ford, for always making me laugh even when I wanted to bang my head against a wall. Sharzhad for being the best conference and travel buddy I could ask for. Bryan Wilder and Haifeng for being two of smartest, most humble and kind people I know. Fei, Thanh and Matthew for being so kind and helpful during my first years. And Amulya Yadav, for having more of an impact on me during my time here than he knows. Thank you for being so supportive, even when I didn't believe in myself and for encouraging me to take risks. I could always count on you for company during late nights in the lab and I can honestly say that my time at USC would not have been the same without you.

I also want to thank my second family who supported me while I was here. Kristen Too and Taylor Aubry, thank you both for being my home away from home. You've always been there for me, and I know I will always be able to count on you no matter what stupid situation I get myself into. James Joly, for putting up with all my insanity these last few months. Alana Shine, for your confidence and self assuredness which has always inspired me to try and find that same confidence in myself. James Preiss, for being one of the most down to earth and real people that I know. Andy Barron, thank you for enabling all my bad habits and for always being there to celebrate my victories and help forget my troubles. Shane Jackson, for being so caring and for keeping me from becoming a hermit during times of stress. Brendan Avent, for showing me how to not take myself too seriously and for listening to me vent more times than I can count. Becca Funke thank you for all our heart to hearts out on the SAL patch, for always cheering me on and being there to cheer me up. Elizabeth Orrico, for being one of my first real friends here, for all that you've shared with me both in and out of the lab, and for always being proud of me. You've made

this place feel like a home, and I mean it when I say I love you all.

Lastly, I want to thank my parents, Maria Luisa Iarocci and Tom McCarthy for always believing in me. For telling me I was brave, when I felt scared. For always being so proud of even my smallest accomplishments. And for seeing me as the person I've always wanted to be. Your help and support have made me that person today and there is no way that I could ever thank you enough for it.

Abstract

In the presence of an intelligent adversary, game theoretic models such as security games, have proven to be effective tools for mitigating risks from exploitable gaps in protection and security protocols, as they model the strategic interaction between an adversary and defender, and allow the defender to plan the use of scarce or limited resources in the face of such an adversary. However, standard security game models have limited expressivity in the types of planning they allow the defender to perform, as they look only at the deployment and allocation of a fixed set of security resources. This ignores two very important planning problems which concern the strategic design of the security system and resources to deploy as well as the usability and implementation of the security protocols. When these problems appear in real world systems, significant losses in utility and efficiency of security protocols can occur if they are not dealt with in a principled way.

To address these limitations, in this thesis I introduce a new hierarchical structure of planning problems for security games, dividing the problem into three levels of planning (i) Strategic Planning, which considers long term planning horizons, and decisions related to game design which constrain the possible defender strategies, (ii) Tactical Planning, which considers shorter term horizons, dealing with the deployment of resources, and

selection of defender strategies subject to strategic level constraints and (iii) Operational Planning, dealing with implementation of strategies in real world setting.

First, focusing on Strategic Planning, I address the design problem of selecting a set of resource and schedule types. I introduce a new yet fundamental problem, the Simultaneous Optimization of Resource Teams and Tactics (SORT) which models the coupled problem of both strategic and tactical planning, optimizing over both game design with respect to selection of resource types, as well as their deployment actual in the field. I provide algorithms for efficiently solving the SORT problem, which use hierarchical relaxations of the optimization problem to compute these strategic level investment decisions. I show that this more expressive model allows the defender to perform more fine grained decision making that results in significant gains in utility. Second, motivated by the relevance and hardness of security games with resource heterogeneity, I also address challenges in tactical planning by providing a framework for computing adaptive strategies with heterogeneous resources. Lastly, I look at the problem of operational planning, which has never been formally studied in the security game literature. I propose a new solution concept of *operationalizable strategies*, which randomize over an optimally chosen subset of pure strategies whose cardinality is selected by the defender. I show hardness of computing such operationalizable strategies and provide an algorithm for computing ϵ-optimal equilibria which are operationalizable.

In all of these problems, I am motivated by real world challenges, and developing solution methods that are usable in the real world. As such, much of this work has been in collaboration with organizations such as Panthera, WWF and other non-governmental organizations (NGOs), to help protect the national parks and wildlife against deforestation and poaching, and the TSA, to protect critical infrastructure such as our airports from

terrorist attacks. Because of this, in addressing these three levels of planning, I develop solutions which are not only novel and academically interesting, but also deployable with a real world impact.

Part I

Introduction and Background

Chapter 1

Introduction

It is a well known principle that there is no such thing as absolute security [Jim Breithaupt, 2014]. Limited availability of resources that may be deployed within a security protocol, due to either constraints on budget, time or capacity, mean that is is impossible to defend and protect against every risk. Indeed, when these resources are scarce, it becomes even more crucial to carefully plan their proper use in order to avoid resource waste. Thus the problem of resource allocation is a well studied planning problem in the security literature. However, static solutions to this problem which maintain the allocation of resources constant over time are highly predictable and thus severely suboptimal in the face of adversaries conducting surveillance of the defense protocol in place. Game-theoretic models, in particular *security games*, were proposed as a means to mitigate this predictability against strategic adversaries [Tambe, 2011, Korzhyk et al., 2010a, Yin et al., 2015, Balcan et al., 2015, Basilico et al., 2009] as they model the dynamic interactions of an adversary with the security system. Over the past decade, these models have continued to grow in sophistication and realism, incorporating uncertainty, complex adversary behavior models, spatio-temporal constraints and other extensions to better model real world problems. As this area of research continues to evolve, security agencies have begun adopting these more sophisticated game-theoretic defense strategies leading to many real world deployments and field tests of these models, among which are large organizations

such as the United States Coast Guard [Shieh et al., 2012a], the Transportation Security Administration [Brown et al., 2016], and the Los Angeles airport [Pita et al., 2008a].

However, all of the past work in this area has focused on a small subset of the planning problem; namely how to optimally deploy a given fixed team of resources. While this allows an organization to optimize over the tactical deployment of their resources i.e. their assignment to targets to be protected, it ignores other higher and lower level aspects of the problem which often influence the choice of tactical resource allocation. Concerns related to the higher level strategic planning problem of resource management, such as the purchasing of resources, as well as lower level operational planning, relating to the actual implementation of security protocols, are not captured by past work in security games. The resource allocation problems considered in previous work is limiting in that it only considers (i) an assumed prior set of fixed resources, with no optimization made over the resource types, or cost benefit analysis (ii) where those resources are mainly homogenous and (iii) coordination is scarce. Operational planning on the other hand has never been formally studied in the security game literature. While prior work has considered implementability of solutions with respect to constraint satisfaction, this does not model the problem of operationalizing these strategies, where proper execution may fail due to errors or unforeseen constraints.

The planning problems form a hierarchy where strategic level design decisions constrain the possible space of tactical level deployment decision, which must then be operationalized in the security system. This notion of using hierarchical levels of decision making and planning is common practice in both business and military organizations,

used in key management problems such as risk management and supply chain management [Bilge Bilgen, 2004, Al-Mashari et al., 2003, Farahanim Misni, 2017], where typically analysis is done on these three levels, *strategic*, *tactical* and *operational*. This type of planning decomposition allows organizations to more effectively analyze the effects of their decisions on multiple levels, allowing them to define long term goals, plan investments as well as day to day operations of their organization.

While the semantics of these types of planning can vary depending on the type of organization, in general they deal with similar time frames and scopes of decision making [Ballou, 1973]. *Strategic planning*, in general, looks at long term goals of an organization, and as such deals with decisions or commitments that will persist over long periods of time. These types of decisions are generally related to resource management and include investment planning or choice of particular infrastructure. *Tactical planning*, influenced by the decisions made in the strategic plan, define how an oraganization manages existing units, personnel and equipment. Lastly *operational planning* deals with ensuring that the routine activities of an organization are executed properly i.e. ensuring that the tactical plans are properly implemented. Examples of this include designing work flows and tasks to reduce error and improve speed or aligning staff incentive structures with risk objectives to avoid reckless behaviors. As an example of hierarchical planning in the real world, Table 1.1 shows an instantiation of these three levels of planning with respect to supply chain management.

Strategic Planning	**Longer Time Scale** • Defining Policy • Allocation of Funds • Portfolio Optimization • Aquisitions and Mergers	How many warehouses are needed? Where should they be located? What regions should each warehouse serve?
Tactical Planning	**Shorter Time Scale** • Resource Deployment • Supply and Demand Management • Transportation Planning	What items should be stocked in what locations? How much storage capacity should there be?
Operational Planning	**Day to Day Activities** • Production Management • Order Fulfillment	How should individual jobs be scheduled in the warehouse?

Table 1.1: Example of Strategic, Tactical and Operational Planning in supply chain management [Miller, 2016].

1.1 Problem Statement

Given the growing adoption of the security game model as a framework for real world security system, it is important to ensure that these models are as effective and efficient as possible; and so, given the ubiquitous use of hierarchical planning structure in other real world systems, this thesis not only formalizes the notion of hierarchical planning for security games, but also provides solution methods for solving these complex planning problems. The goal is to ensure that the security game model is both practical and profitable, meeting the security needs of real world organizations as well as being usable by real world systems. Thus the problem my thesis aims to address *is how to efficiently allocate resources in security games, while considering strategic, tactical, and operational considerations simultaneously*.

1.1.1 Hierarchical Planning in Security Games

Looking at the security game framework there exists a similar hierarchy of strategic, tactical and operational planning problems. To date, security game literature focuses solely on the tactical planning problem. This is due to the focus of these models being the allocation of a fixed set of homogeneous security resources. However, it is often the case that an organization interested in optimizing their defense system not only has flexibility in how they allocate their security resources, but in design decisions as well. In security games, these design decisions correspond to choosing what types of security resources should be used as well as how to form teams among these resources. Parallels may be draw between these design decision and the strategic planning problem of portfolio optimization in economics, where the choice of a set of resources to invest in forms a portfolio with a return

value determined by their tactical deployment. In contrast to previous work in security games, this optimization over a portfolio of security resources deals with high level strategic decisions on the management of non-security resources such as money and time rather than the deployment of physical resources. This *strategic planning* is particularly challenging given that the expected value of the investment of non-security resources requires computing the actual deployment of security resources in the field. In this thesis, I provide a formal model of this problem, which I refer to as the Simultaneous Optimization of Resource Teams and Tactics (SORT) as a new fundamental research problem in security games that combines strategic and tactical decision making [McCarthy et al., 2016b].

A major difficulty in addressing this new question comes from the fact that the tactical problem is already computationally challenging to solve, making it difficult to evaluate the effectiveness of any strategic level decision. While considering only the tactical decision problem is sufficient for many domains, there are many more in which both this and the strategic decision problem are critical components of the security problem. A current major motivation is the domain of environmental crime, and environmental sustainability, where the security game framework is being applied to secure large conservation areas for the protection of forests, fish and wildlife [Haskell et al., 2014a, Yang et al., 2014a, Fang et al., 2015]. Unfortunately these areas are most often found in developing countries, where budgets for security are often very limited, making it crucial to allocate not only the security resources efficiently but to also allocate the limited funds for these security resources efficiently.

The *tactical planning* problem on the other hand is the classic problem considered in the security game literature: how to best *deploy* a set of resources. While there has been much work in developing rich models for tactical planning, there has been little work in

the types of models which would benefit the most from additional strategic and operational planning; in particular models where there is resource heterogeneity. This is an important problem as many prominent classes of games, such as the Network Security Games (NSG) which are used to model the problems in green security and environmental crime currently do not model resource heterogeneity, meaning that contributions to the area of tactical planning are also needed in order to solve the SORT problem.

Lastly, *operational planning* in security games refers to the operationalization or implementation of security protocols in practice, and provides a way to explicitly reason about the usability of a system. The goal of operational planning should be to minimize user error and facilitate proper implementation and usability of security protocols. While usability concerns have always been present in deployed security games, these have often been addressed in an ad-hoc fashion, and not explicitly discussed in the literature. To date, there has been no formalization of the notion of usability in security games and no investigation on how to compute strategies which adhere to a notion of usability. This is problematic as improper implementation of strategies, either due to user error or unforeseen constraints can significantly impact the efficiency and effectiveness of a system.

1.1.2 Contributions

To address these limitations in this work, I formalize the hierarchical planning problems for security games and make contributions in the three areas of strategic, tactical and operational planning for security games. First, my work formalizes the strategic planning problem with the SORT model to combine *strategic* and *tactical* decision making. I provide an efficient algorithm for solving this problem which uses a hierarchical approach to abstract the security game at varying levels of detail, providing bounds on the value

of different teams of resources to speed up the search for the optimal team. This work is done in collaboration with conservation criminologists as well and NGOs engaged in forest protection in Madagascar where we worked to construct accurate game models to evaluate the solution approaches. I also look at strategic planning in the multistage setting, where the availability or budget for non-security resources may vary with time, and where strategic level decision may be conditioned on *past* and *learned* information. While these types of problems can be modeled as a type of stochastic control processes, these models are often not scalable and so in this work I provide a fast scalable Partially Observable Markov Decision Process (POMDP) formulation to address this challenge. This work also uses abstractions to build an ensemble of virtually distributed POMDP (VD-POMDP) agents, where individual policies for different sub-agents are planned separately and their sparse interactions are only resolved at execution time to determine the joint strategic level decisions and tactical deployments in each round.

To address limitation in tactical planning, my work builds on the past work in security games by addressing a space of tactical planning problems which has not been well studied: that of dealing with heterogeneous resource allocation, i.e. how to optimize the formation and assignment of teams of resources. These are the types of tactical planning problems that must be solved in the SORT problem, and thus form an important set of problems to consider in the context of hierarchical planning. There are several challenges associated with incorporating heterogeneous resources into these games, (i) scheduling constraints become more complicated to deal with and (ii) resources may be combined into a combinatorial number of teams. I address these challenges in two different classes of games, Network Security Games (NSG) and Threat Screening Games (TSG). In the first class, resources are constrained by the underlying graph structure, where the strategy

spaces then are combinatorial. This problem is made even more difficult with heterogeneous resources as the strategy space is then the product space of each of the allowed paths for each of the individual resources. To address this, I provide a novel and efficient double oracle formulation for computing exact and approximate equilibria to these games. The second class of games deal with the problem of dynamically allocating screening resources of different efficacies (e.g., magnetic or X-ray imaging) at checkpoints (e.g., at airports). The scheduling constraints in these games are due to there being limited capacity on the use of each of the resources, so that assigning too many items to one team may affect the use of another team if they share a resource. Complexities arise in these types of games as there is also uncertainty in the deployment of the team of resource. With threat screening, arrival times of the items to be screened may be uncertain. Past work in threat screening operated under the strong assumption of perfect knowledge in arrival times, which severely impedes the implementability and performance of this approach. Addressing this challenge is difficult, as it requires reasoning about all the possible realizations of the uncertainty and coming up with an optimal plan for each of those scenarios, one which appropriately balances the tradeoffs between throughput efficiency and screening effectiveness. To address this, I introduce a novel framework for dynamic allocation of heterogenous screening resources known as Robust Threat Screening Games (RTSG) that explicitly accounts for uncertainty in the screenee arrival times, as well as the coupling of capacity constraints in time[McCarthy et al., 2017]. I provide a tractable solution approach using compact linear decision rules combined with robust reformulation and constraint randomization. I performed extensive experiments which showcase that my approach outperforms *(i)* exact solution methods in terms of tractability, while incurring

only a very minor loss in optimality, and *(ii)* methods that ignore uncertainty in terms of both feasibility and optimality.

Lastly, with respect to *operational planning*, I look at how to improve the execution of the strategic and tactical plans by formalizing the notion of *usability* of the computed equilibrium strategy solutions of security games. This work is the first to look at *operational planning* in security games, and I propose a new game theoretic solution concept of *operationalizable* strategies which have desirable properties with respect to usability. I introduce the Price of Usability, which measures how the effectiveness of strategies degrade due to usability constraints, and provide analysis with on the Price of Usability for various types of games. To address this new problem of computing operationalizable strategies, I provide an approximate method for computing ϵ-equilibria with operationalizable strategies as well as a scalable heuristic method.

1.1.3 Thesis Overview

The structure of the remainder of the thesis will be as follows: Chapter 2 discusses background material for security games and for the specific classes of games which are relevant to this thesis. I will also discuss related work in the literature as it pertains to team formation and usability. In Chapter 3, I formalize the strategic planning problem for security games and provide example domains where the problem is particularly relevant. Chapter 4 discusses the strategic planning problem in the multistage setting and looks at how the sort problem may be addressed with decision theoretic models. Chapter 5 focuses on tactical planning in security games and what types of tactical problems induce a need for hierarchical planning. Chapter 6 formalizes operational planning in security games, first discussing why usability concerns are so important to address and then formalizes what it

means to have strategies which are usability compliant. Chapter 7 summarizes the thesis and discusses potential directions for future works.

Chapter 2

Background and Related Work

2.1 Game Theory & Security Games

Over the past decade, game-theoretic approaches, in particular *security games*, have become a major computational paradigm for security resource allocation [Tambe, 2011, Korzhyk et al., 2010a]. There are several sub-area of security games relevant to this thesis, which are described in the following sections.

2.1.1 Stackelberg Security Game (SSG)

The standard security game model is a Stackelberg Security Game; it is a two player game played between a leader (known as the defender) and a follower (known as the adversary). There are a set of target $\mathcal{V}$ under the defenders protection; the adversary chooses one target out of this set to attack. In order to protect the targets from the adversary, the defender has a limited set of resources $\mathcal{R}$ which may be deployed to thwart an attack. In order to protect a target, the defender must assign a resource to that target. Typically, $|\mathcal{R}| < |\mathcal{V}|$, i.e. the defender resources are limited so there are not enough resources to cover all the targets. The defender actions, being assignments of all resources $r \in \mathcal{R}$ to targets, are known as pure strategies, so that the defender strategy space is the set of all possible assignments of resources to targets. The adversary actions correspond to a choice of target $v \in \mathcal{V}$ to attack, so that the adversary pure strategy space is equal to $\mathcal{V}$.

Each target has a particular value to the each of the players, and so there is an associated reward and penalty received by each player in the event of a successful or unsuccessful attack. If the adversary chooses to attack a target v which is uncovered by the defender, the adversary receives a payoff of $\mathrm{R}_t a$ and the defender receives a penalty of $\mathrm{P}_{d,v}$. If on the other hand the target is protected by a resource the adversary receives a penalty of $\mathrm{P}_{a,v}$ and the defender receives a reward of $\mathrm{R}_{d,v}$, where $\mathrm{R}_{d,v} > \mathrm{P}_{d,v}$ and $\mathrm{R}_{a,v} > \mathrm{P}_{d,v}$.

While each player's strategy set is the set of all pure strategies for that player, both players may choose to play *mixed strategies* which are probability distributions over pure strategies. In Stackelberg Security Games the assumption is that the adversary may choose their strategy after observing the defender's strategy, the leader (defender) commits to playing a mixed strategy and the follower (adversary) observes this mixed strategy and responds. Optimal solutions to these games are typically mixed strategy solutions for the leader, as it is important to randomize over the assignments of resources to targets in order to remain unpredictable to the adversary.

The standard security game model assumes that each player is utility maximizing. The adversary will then choose a target among the set. The solution for such a game is known as a Stackelberg Equilibrium (SE) , of which there are two kinds: the Strong Stackelberg Equilibrium (SSE) and the Weak Stackelberg Equilibrium (WSE) [Leitmann, 1978, Breton et al., 1988]. The difference in these two types of equilibrian are in how the adversary breaks ties when there are multiple targets with the same utility; the adversary is then indifferent to which of the targets among these they choose to attack. The SSE assumes that the adversary breaks ties in favor of the defender and chooses the one that is optimal for the defender, while WSE assumes that the adversary chooses the worst one for the defender. The SSE is guaranteed to always exists, and can be induced by the

defender by deviating from the optimal strategy by infinitesimally small amounts in order to increase the utility of the desired target for the adversary [Stengel and Zamir, 2004].

2.1.2 Network Security Games (NSG)

Network Security Games are an important class of security games due to the complex scheduling constraints that are often present in these games. A Network security game (also known as a pursuit evasion game) is a game played on a graph $G(N, E)$. This may be used to represent any type of network, ranging from physical networks like transportation networks or airline networks, non-physical networks such as computer systems or social networks, and even abstract networks such as attack graphs. The key feature here is that the actions of one or more of the players in the game are constrained by the network. There are a set of targets corresponding to some subset of nodes in the graph $\mathcal{V} \subset N$. All players move along edges in the graph and thus their strategy spaces correspond to the set paths on the graph. A huge challenge in solving network security games is scalability; because players are choosing paths in the network, the space of actions or pure strategies of the defender or attacker can be combinatorially large, and grows exponentially with the size of the network, making these problems extremely computationally challenging. NSG appear in many domains, such as allocation of police checkpoints to roads in the city [Jain et al., 2011, Jain et al., 2013], scheduling air marshals on-board international flights [Tsai et al., 2009], and computing patrols routes for park rangers.

There are several important subclasses of NSG, all which differ in how each of the players in the game are restricted by the graph structure.

One subclass of network security games where the adversary is mobile and the defender resources are static, is the so-called Interdiction Games. This corresponds to

problems such as sensor placement when the goal of the defender is to collect information, or checkpoint placement when the goal of the defender is to interdict, while on the other hand the attacker is choosing a path. This problem has been studied in a variety of domains; notably it has been used to model the problem of placing checkpoints in urban road networks (specifically for Mumbai)[Jain et al., 2011, Jain et al., 2013] where the adversary is attempting to traverse a path through the city to reach nodes corresponding to valuable targets to attack and the defender can choose edges in the network on which to place checkpoints. In the realm of security, it has also been used to model drug traffic networks [Wood, 1993], planning air strikes [P.M. Ghare and Turner, 1971] and preventing nuclear smuggling [Morton et al., 2007].

Network security games where the attacker is stationary and the defender(s) are mobile are known as *search games* [Isaacs, 1965, Gal, 1980, Alpern and Gal, 2003]. These types of games often occur in the domain of green security and Green Security Games (GSG) which model key features in domains related to environmental sustainability, conservation and wildlife protection. In these domains several of the following challenges may be present: bounded rationality in the adversary decision making, frequent and repeated attacks, and complex spatio-temporal constraints. A common problem in this domain is planning patrol routes for rangers in conservation areas who are trying to detect animal snares placed by poachers looking to trap elephants, rhinos or other valuable species [Fang et al., 2017, Fang et al., 2016, Ford et al., 2014a, Nguyen et al., 2016b, Yang et al., 2014b].

Finally, when both defenders and attackers are mobile, we have network security games referred to as *Hider-Seeker Games or Patrolling Games*. These games are typically the most challenging to solve, as both players are restricted by the graph structure and thus the strategy spaces for both players are typically combinatorial in size.

While there has been much past work in solving these types of games, all of the existing work has looked only at the tactical allocation problem, assuming either a single defender or a fixed set of homogeneous defender resources. When considering higher level planning problems, like strategic level planning, this class of games is particularly interesting due to the complex scheduling constraints that the network imposes on the problem. Additionally many real world constraints can be modeled using networks and so NSG represent a broad class of games. Cyber security has received a large amount of attention in the NSG literature as the computer networks form the underying network structure of the game [Lelarge and Bolot, 2008, Preciado et al., 2014, Roy et al., 2012, Omic et al., 2009, Zhu and Basar, 2015, Vaněk et al., 2012]. Additionally, physical security problems form an important domain for NSG, as many problems with spatio-temporal constraints can be modeled as a network problem. Examples from interdiction, search and patrolling games previously discussed include domains such as protecting flight networks, to transportation networks in cities, wildlife protection, as well as the protection of critical infrastructure such as ports and ferries [Fang et al., 2013a].

2.1.3 Threat Screening Games (TSGs)

While the notion of teams and resource heterogeneity is missing from the large body of work in security games, Threat Screening Games are class of games related which explicitly model heterogenous teams of resources in a game theoretic setting. First introduced

by [Brown et al., 2016], Threat Screening Games (TSG) are Stackelberg games played between a leader (defender) and a follower (adversary). The defender operates a series of checkpoints wherein she must screen a set of incoming passengers with known arrival times using screening resources of different efficacies. The adversary disguises themselves as one of the passengers and attempts to bypass the defender's screening; the defender must then determine an optimal screening strategy (dynamic assignment of resources to screenees) to catch the adversary while simultaneously ensuring efficient throughput of passengers through the checkpoints. Optimizing the defender (mixed) strategy by means of the TSG captures the strategic behavior of attackers and thus yields more effective screening strategies. TSGs are inspired by previous research in security games where a defender protects a set of targets from a strategic adversary. However, TSGs differ significantly because they *(i)* do not have an explicitly modeled set of targets; *(ii)* include a large number of non-player screenees that must be screened while a single adversary attempts to pass through undetected; and *(iii)* encompass screening resources with differing efficacies and capacities that are combined to work in teams. These key differences make TSGs more appropriate for screening settings.

An important feature of TSG that is relevant to this thesis is the fact that there are heterogeneous screening resources which can be combined together to form teams. Heterogeneity in resources gives us more more flexibility and increases the size of the decision space when we consider higher level planning problems like strategic planning. Existing work in TSG have not explored this additional planning space, assuming that the teams of resources are a fixed given prior.

Additionally, even within the tactical planning problem that TSG's consider, there are significant limitations. Despite promising results, previous work in TSG fails in its mission to realistically model real-world settings as it relies on the strong assumption of perfect fore-knowledge of screenee arrival times (e.g., arrival times of passengers at airports). Addressing this challenge is difficult, as it requires reasoning about all the possible realizations of the uncertainty and coming up with an optimal plan for each of those scenarios. The importance of this is discussed further in later section, where in my work I address these significant limitations in the tactical deployment problem and provide efficient solution methods for dealing with this uncertainty.

2.2 Decision Theory and POMDP

Decision theory deals with planning in a non adversarial, non-reactive or adaptive world, where other agents do not respond to the actions taken by the planner. Markov Decision Processes (MDP) and Partially Observable Markov Decision Processes (POMDP) are decision theoretic models used for making decisions in stochastic environments under uncertainty. These models can be useful from a security perspective as they are very expressive and deal with problems of sequential decision making by considering the long term effects of each of the decisions and actions which are taken. As such they are well suited to model the SORT problem for non adversarial settings, and can integrate reasoning on long term strategic scales as well as shorter term tactical scales. Formally, an MDP model is a tuple (S, A, T, R) with state space S, action space A, state transition function T, and reward function R, where the goal of the MDP agent is to maximize the long term reward they receive from interacting with their environment. POMDP models

are similar except that the agent is not able to fully observe their environment. Instead they get observations about the environment and maintain a belief over the state space. The POMDP model is then also a tuple (S, A, T, O, Ω, R) with an additional observation space Ω, observation probabilities O. The solutions to these models are *policies* for choosing actions which depend on the current state or belief state of the agent. These models integrate strategic planning into the decision problem by reasoning about the long term effects of any actions. MDP's have been used to represent defender strategies in security games [Jain et al., 2013, Shieh et al., 2014, Feng et al., 2017]. However, when defender resources are heterogenous this can cause the state space of the MDP to grow very large. Additionally, when resources have the ability to perform information gathering actions, the strategies then need to be modeled as POMDP's. This occurs very frequently in domains such a cybersecurity where the defender has many sensors and detectors that can be deployed to determine when and where an attack is happening. However, it is known that offline POMDP solving is intractable for large problems [Gerkey and Mataric, 2003, Madani et al., 1999, Papadimitriou and Tsitsiklis, 1987] and thus, in this thesis, we focus on speeding up the offline POMDP solving by performing a series of abstractions of the original POMDP. Our technique of solving the POMDP is inspired by conflict resolution techniques in solving distributed POMDP [Nair and Tambe, 2005, Jung and Tambe, 2003] and distributed POMDPs with sparse interactions [Varakantham et al., 2009, Velagapudi et al., 2011].

2.3 Team Formation

An important part of strategic planning for security games is the problem of resource and team selection. While there has not been any work which addresses this problem for security games, there is significant research which has been done in team formation for multiagent systems, [Liemhetcharat and Veloso, 2012, Hunsberger and Grosz, 2000] with applications in network configuration [Gaston and desJardins, 2005], fantasy football [Matthews et al., 2012] and multi-objective coalition [Cho et al., 2013]). The problem of how to allocate agents to specific roles to maximize performance has been studied for mission rehearsal and RoboCupRescue [Nair and Tambe, 2005]. Other research has considered how to lead an ad-hoc team to the optimal joint action [Agmon and Stone, 2012], automatically configuring a network of agents [Gaston and desJardins, 2005], and agents cooperating in a board game [Obata et al., 2011]. There has also been some work on team formation in adversarial settings, though not specifically on security games. Team formation has been used to beat human players in fantasy football [Matthews et al., 2012], and build teams of diverse agents to improve performance of a Goplaying agent [Marcolino et al., 2014]. Multi-objective optimization for a coalition and how trust affects the success or failure of a mission in a tactical game is also studied in [Cho et al., 2013]. However, that work fails to security resource allocation at the tactical level.

There has also been some work in coordinating teams of heterogeneous defender resources. [Shieh et al., 2013] looks at the problem of centralized planning of multiple defender resources which conduct joint activities to protect a set of targets from an attack. This work is improved in [Shieh et al., 2014] in which the defender's pure strategies are represented as Dec-MDPs that allows for a defender to directly optimize for coordination

among resources given uncertainty. In regards to understanding and characterizing the cost of miscoordination in games with multiple defenders [Jiang et al., 2013, Hota et al., 2016] provides basic research that analyzes the trade-off between centralized and decentralized scheduling of defender resources. We plan to build on these works and conduct basic research that quantifies the cost of miscoordination in a decentralized game with multiple defenders that have uncertainty in the execution of their protection actions.

2.4 Usability

While usability concerns have always been present in the deployment of security game models, when these issues materialize in practice they are usually dealt with in an ad hoc manner due to there being a lack of a *principled way of reasoning about these issues.* While there exists a wealth of literature on the root causes of these usability issues, there is no existing security game literature which integrates these models explicitly in the optimization and planning problems. However, there are several areas of related research that looks at usability in other contexts. We focus here in particular on Cognitive Load Theory which studies how tasks can be designed to reduce working memory load in order to optimize performance.

Cognitive Load Theory

In practice, many security systems are operated by people, which becomes important to consider when we are looking at how to design an effective system. Cognitive load theory broadly studies how human brains store and retain information, and has found success in providing guidelines for the best way to present information and design tasks in order to

maximize performance and successful completion of these tasks. As such, we can rely on it to inform our design of security protocols, or in the case of systems built off of game theoretic models, mixed strategy solutions, as these are often executed by people. Studies have found that increased cognitive load negatively impacts the likelihood of successful completion of a task, as it increases the risk of user error. Identifying and decreasing the sources of cognitive load in any system then becomes an important component in ensuring the successful execution of a task. While there can be many sources of cognitive load, an important source is task complexity as it has been shown to not only increase the cognitive load on an individual due to limited memory [Hogg, 2007], but individuals under high cognitive load are more likely to perform poorly at complex tasks [Paas, 1992, Cooper and Sweller, 1987]. Additionally, it has also been reported that a heavy cognitive load can also increase stereotyping [Biernat et al., 2003], which can raise important ethical issues, e.g., in the airport passenger screening domain.

Simple Mixed Strategies

For many real world security domains, such as threat screening at airports [Brown et al., 2016, Schlenker et al., 2016, McCarthy et al., 2017] or planning federal air marshal schedules[Jain et al., 2010b], the corresponding security game models often have strategy spaces which grow exponentially with the number of resources to be allocated. Within these potential strategies the number that may be used in an optimal solution may also be extremely large, resulting in mixed strategies with extremely large supports. Additionally each of these strategies may be composed of a large number of complex tasks. Both of these place extra cognitive load on the individuals required to be familiar enough

with this large variety of security protocols (ie. pure strategies) to execute them all properly.

The notion of small support mixed strategies has appeared in the literature, where in [Lipton et al., 2003] the authors prove the existence of ϵ-optimal strategies, when only a subset of all pure strategies is used. Similarly in [Arieli and Babichenko, 2017], the authors prove the existence of an approximate equilibrium for multi-player games while using a limited number of pure strategies. However, none of this work provides methods for computing such equilibria in practice. To the best of our knowledge, [Paruchuri et al., 2007] is the only paper that explicitly discussed limiting the number of pure strategies in security games; although they only handled small games (100s pure strategies), and they did not consider the impact of such restriction on solution quality.

Part II

Strategic Planning

Chapter 3

Strategic Planning in Security Games

In the following section, I present the strategic planning problem in the context of security and game theory. I provide a formal definition of strategic planning for security games, discussing how it differs from the tactical planning problem and how these two problems are related. I will also discuss domains in which it becomes necessary to consider the strategic planning problem, in particular, those in which the strategic planning problem is difficult to solve in practice.

Security games are game theoretic models used to address the problem of *resource allocation and resource management* in the face of an intelligent and adaptive adversary. Strategic planning refers to the subset of planning problems that deals with long term and high level decisions in an organization. In order to understand what this means for security games, we look at the types of decisions which need to be made when discussing security. A key difference in strategic vs tactical planning is then in the types of resources that they manage. When discussing security, we can divide up the resources into *security resources* and *design resources* where the key difference in these types of resources is how they interact with the adversary. *Security resources*, being the focus of most of the existing work in security games, are resources which have direct interaction with the adversary, being assigned to protect targets in the game. These resources directly impact the probability of success of any attack and thus directly impact the defender's expected

utility, and thus directly affect the objective of the problem. However, the availability or use of these security resources are often constrained by other resources, such as a budget or the availability of staff. Since in most of the existing work in security games these resources are fixed and given as a part of the game design or game structure, I refer to these resources as *design resources*. Design resources do not directly interact with the adversary, but rather with the *security resources*, and thus generally form the basis for the constraints structure of the game.

For security games, *strategic planning* then refers to planning the use of the *design resources* while *tactical planning* refers to planning, the use of *security resources*. While these two planning problems both deal with types of decisions, they are highly related. When it comes to the question of strategic planning if we want to determine what an optimal plan for the design resources are, we need to look at how the design resources affect the security resources, i.e how does the strategic plan affect the space of possible tactical plans, and which strategic plan allows us to execute the best tactical plan? This thesis provides a formal model of this problem, which I refer to as the Simultaneous Optimization of Resource Teams and Tactics (SORT) and describe in the next section, as a new fundamental research problem in security games that combines strategic and tactical decision making.

3.1 Simultaneous Optimization of Resource Teams and Tactics

Recall that in security games we look to compute a mixed strategy $p \in \mathcal{P}$ distribution over pure strategies $q \in \mathcal{Q}$ corresponding to a Stackelberg equilibrium given a payoff structure $\{\mathrm{R}_d, \mathrm{P}_d, \mathrm{R}_a, \mathrm{P}_a\}$ and an adversary.

Each pure strategy will correspond to an allocation of a set of (possibly heterogenous) security resources. Abstractly, let x_r^i represent the tactical allocation of security resource r in pure strategy i. Each resource deployed in pure strategy is subject to a set of design constraints G_r so that $x_r^i \in G_r \ \ \forall i$. A mixed strategy, being a distribution of a subset of all the pure strategies, is constrained by the design constraints of the problem: we can only randomize over pure strategies which all satisfy the set of design constraints G_r.

$$\mathcal{P} := \left\{ p \in \mathbb{R}^n \ : \begin{array}{lll} p_i \geq 0 & & \forall i = 1, ..., |\mathcal{Q}| \\ x_r^i \in G_r \ \text{ if } \ p_i > 0 & & \forall i = 1, ..., |\mathcal{Q}|, r \in \mathcal{R} \\ \sum_{i=1}^{|\mathcal{Q}|} p_i = 1 & & \end{array} \right\}.$$

The strategic planning problem allows us to make decisions that modify the design constraints of the problem by changing the use of the design resources. Let $\mathcal{D}$ be the set of all design resources and let y_d be the strategic planning variables, corresponding to the allocation of design resource $d \in \mathcal{D}$. Since randomization only occurs over the *security resources*, these variables are not indexed by i and thus must be constant across all pure strategies. The set of constraints on the resources now becomes a function of these new

variables $G_r(y)$.

Optimizing the strategic plan requires us to consider both the strategic and tactical plan simultaneously, as ultimately the objective we are attempting to optimize is the game value $F(x)$ which is explicitly determined by the tactical allocation variables.

Example 1. *Take as an example the problem of choosing how many resources r among a set $\mathcal{R}$ to purchase, given a budget B and cost per resource b_r. The set of $r \in \mathcal{R}$ chosen resource types are combined into a team t indexed in the set $t \in \mathcal{T}$ of possible teams. This set T has cardinality combinatorial in the number of resource types. The set of resources in team t is given by $\mathcal{R}(t)$. There is one design resource here, being the budget B, and strategic decision variables here model the allocation of this budget to each resource of type r, determining the number that will be purchased. We then want to assign these resources to a set targets $\mathcal{V}$. The tactical allocation is then a vector of binary variables indicating wether a resource of type r is allocated to target v, then denoted $x^i_{r,v}$ so that $\sum_{v\in\mathcal{V}} x_{r,v} \leq y_r$. Here $G_r(y_r) = \{0 \dots y_r\}$. The mixed strategy space can then be expressed as the following:*

$$\mathcal{P} := \left\{ p \in \mathbb{R}^n \ : \begin{array}{ll} p_i \geq 0 & \forall i = 1, ..., |\mathcal{Q}| \\ \sum_{i=1}^{|\mathcal{Q}|} p_i = 1 & \\ p_i \sum_{v\in\mathcal{V}} x_{r,v} \leq y_r & \forall i = 1, ..., |\mathcal{Q}|, r \in \mathcal{R} \\ \sum_{r\in\mathcal{R}} y_r b_r \leq B & \end{array} \right\}.$$

Where the first two constraints enforce that the mixed strategy is a probability distribution, and the last two constraints enforce that each of the pure strategies in the support of the mixed strategy are consistent with the design constraints $G(y)$. The expected defender utility $U_d(p)$ is a function of the mixed strategy p.

Let $F(t)$ be the optimal objective value of the underlying game played with team t.

$$F(t) =: \max_{p \in \mathcal{P}} \left\{ U_d(p) : \sum_{r \in \mathcal{R}(t)} 1 = y_r \right\}$$

The SORT problem can be formulated as:

$$\max_{t \subset \mathcal{T}} \left\{ F(t) : \sum_{r \in \mathcal{R}(t)} b_r \leq B \right\}$$

Theorem 1. *The SORT problem is NP-Hard even if we can evaluate $F(t)$ in constant time.*

Proof. The proof is constructed using a reduction to Knapsack

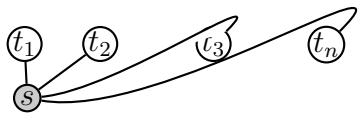
Construct an instance of the SORT problem from Example 1 as follows. Given a set of resources $r_i \in R$, each with a cost $b_i \in B$, we want to choose a set or resources $S \subset R$ which maximizes the game value function F(S), and which has a total cost less than B.

$$\max_{S \in R} \left\{ F(S) : \sum_{i \in S} b_i \leq B \right\} \tag{3.1}$$

Knapsack: Given a set of n items with non-negative weights w_i, and values v_i, is there a subset of items with total weight at most W, such that the corresponding value is at least V?

$$\max\left\{\sum_{i}^{n} v_i x_i : \sum_{i}^{n} w_i \leq W\right\}$$

We reduce the Knapsack problem to the team formation problem in the following way: we set the total number of resources equal to the number of items in the knapsack $|R| = n$, with a budget equal to the knapsack capacity $B = W$ and the cost of each resource equal to the cost of each item $b_i = w_i$. Additionally, let each resource cover a disjoint number of edges equal to the value of each item $L_i = v_i$.



We then construct the game graph using a single source and a number of targets equal to the sum of all the item values $|T| = \sum_i v_i$, with the value of each target being uniform and equal to the total number of targets $\tau_i = |T|$. Because the resource can cover disjoint edges, each resource can then cover $L_i = v_i$ targets. Because each of the targets are identical maxi-min strategy for the attacker and defender will always be a uniform distribution over the single edge paths to each of the targets. The game value of any game with $|T|$ targets will then be proportional to the number of edges E the defender can cover $F(S) = -\sum_p x_i a_i - \left(\frac{1}{p} \times \frac{E}{p}\right) p^2$. Because of the way the resource coverages were constructed, the total number of edges that can be covered by a team will be equal

to the total value of all the items in the knapsack and the game value can be written as $F(S) = \sum_i^n v_i r_i$.

The solution to the SORT problem will be equal to the maximum value of objects which can be placed in the knapsack since, both problems are then maximizing the same objective function, with the same constraints. The solution to the Knapsack problem is therefore the solution to the SORT problem, and vice-versa.

□

3.1.1 Examples of SORT in the Real World

The problem of simultaneously optimizing strategy and tactics is a problem that arises in many domains. While not limited to this example, it describes two very common decision problems faced in both economics and security. First, given a fixed budget to invest in a set of heterogeneous resources, each with different costs and capabilities, what is the optimal investment into these resources? And once the portfolio of resources has been decided, how should those resources be combined into teams and deployed for optimal efficiency in the field?

In domains related to health and social services, we have the problem of deciding what goods and services to provide to different populations of people. For example, there are many different types of counseling services, such as behavioral therapy, cognitive therapy, family therapy, relationship counseling, etc. with a limit to the number of types of services which may be offered due to budget or staff constraints. Here the strategic problem is deciding how much of each type of services to offer, while the tactical decision corresponds to determining who among the population should be given which combination of services.

Looking at security domains, we can consider the problem of deciding how many security personnel to hire and how much equipment should be purchased. This problem presents itself in many domains where theres are different types of resources available or different types of attacks which may occur; one such domain is the problem of screening for threats. Take the problems of passenger screening at airports where there are many different types of resources available to screen passengers such as x-ray machines, walk through metal detectors, k-9 dogs, patdowns, etc. and many different types of threats that might come through screening such as knives, explosives, guns, etc. and each resource may be more or less efficient at detecting different threats. The strategic problem here is deciding how many of each resource should be used and when to schedule operators for them throughout the day, while the tactical problem is deciding how many of each type of passenger to screen with each resource.

Another important domain is that of environmental crime, and environmental sustainability, where the security game framework is being applied to determine the best way to allocate the security resources available in order to secure large conservation areas for the protection of forests, fish and wildlife [Haskell et al., 2014b, Yang et al., 2014a, Fang et al., 2015]. One reason this domain is so relevant is that, unfortunately, these areas are most often found in developing countries, where budgets for protecting forests are often very limited, making it crucial to not only allocate resources efficiently but also the limited funds for these security resources. Indeed, many of the organizations tasked with protecting these large areas are faced with questions that are difficult to answer as research in these areas is extremely limited. Questions such as how many rangers to hire? What types of patrols (night patrols, day patrols) are most effective? Would this area benefit from

the use of UAV's? Knowing the answers to these questions is critical for the protection of these conservation areas, however organizations are mainly left to rely on intuition.

In the same way that tactical decision support is needed to optimize the deployment of limited resources to protect these large conservation areas, there is clearly a great need for strategic decision support at the management level to optimize the investment of a limited budge into many security resources. These domains are particularly relevant to the SORT problem as there often does not exist any security infrastructure in place, meaning that much or this strategic planning remains to be done, thus there is a real need for this investment planning in order to develop and implement a security system in these areas.

3.2 Motivating Domain and Game Model

Because of the particular relevance of these types of domains to the SORT problem as well as their important societal impact, in this thesis I will focus on environmental security as a motivating domain for the SORT problem, keeping in mind that the results and techniques are widely applicable to other security domains as well. In particular I will focus on the problem of preventing deforestation and securing protected forest areas from illegal logging. These are a major problem for many developing countries; The economic and environmental impact are severe, costing up to USD $30 billion annually and threatening ancient forests and critical habitats for wildlife. As a result, improving the protection of forests is of great concern for many countries [WWF, 2015, Dhital et al., 2015, Thomas F. Allnutt, 2013].

Unfortunately in developing countries, budgets for protecting forests are often very limited, making it crucial to allocate these resources efficiently. In order to protect these areas, security resources can be deployed in order to interdict the traversal of illegal loggers on the network of roads and rivers around the forest area. However, we must first choose the right security team for interdiction. This can be challenging as there are often many different organizations that may be involved — from local volunteers, to police, to NGO personnel — as well as different kinds of equipment which may be used — ground vehicles, UAVS — each differing in their interdiction effectiveness (individual or jointly with others), and with varying costs of deployment. This results in a very large number of resource teams and allocation strategies per team with varying effectiveness that could be deployed within a given budget. The challenge here is to simultaneously select the best team of security resources and the best allocation of these resources.

This work is done in collaboration with both partners in conservation criminology as well as the non-governmental organization Alliance Voahary Gasy [AVG,], working to protect the endagered forests in Madgascar, where valuable hardwood trees such as rosewood and ebony are being illegally harvested at an alarming rate. There is broad interest in improving forest protection via patrolling from different groups, not only AVG but the Madagascar National Parks, local police and community volunteers. However, there is very limited coordination among these groups and, as a result, there is limited attention paid at the *strategic level* to optimize the selection of the right team of resources from among these groups, and the *tactical* level to optimally allocate the resulting team's resources. Our model is designed to address these problems.

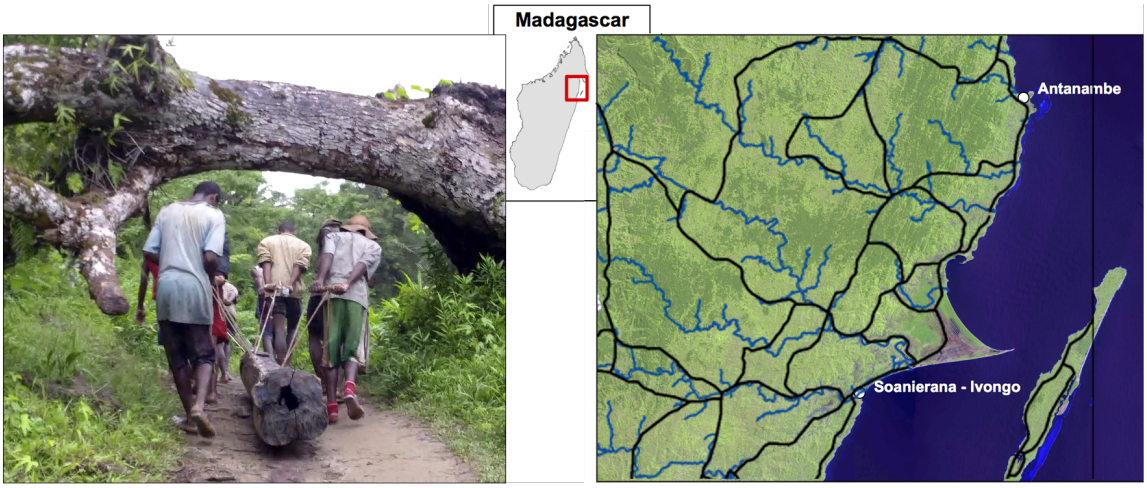


Figure 3.1: Illegal logging in progress in at risk area of Madagascar, provided by the Alliance Voahary Gasy.

3.2.1 SORT for Network Security Games

We now describe our model of the SORT for the illegal logging domain. At the tactical level, the decision of how optimally to allocate a team is a network security game. We model the physical space using a graph G = (N,E), consisting of source nodes $s \in S \subset N$, target nodes $v \in \mathcal{V} \subset N$, and intermediate nodes. The attacker (illegal loggers) acts by traversing a path from a source s_i to a target node v_i. For illegal logging, the s_i may be attacker's originating villages and v_i may be areas containing valuable trees. Each target node t_i has a payoff value that is domain dependent. Based on the research of collaborating domain experts in Madagascar, these depend on the density of valuable trees in a particular area, and the distance to the coast (for export) and local villages.

The choice to model the problem as a network arises from both discussion with local domain experts as well as past experiences in modeling these types of problems

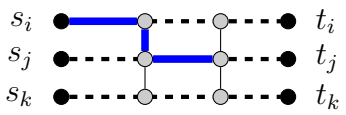


Figure 3.2: Pure strategies for the Defender (bold line) and Attacker (dashed line going from s to t).

[Fang et al., 2016], where due to the difficult terrain and dense forest area, movement in constrained to an underlying ridge-line, river and road network.

Previously models in NSGs assume a defender limited to homogenous resources with no probability of failure, no joint effectiveness and the ability to only cover a single edge [Jain et al., 2013, Okamoto et al., 2012]. This is insufficient to capture the complexities present in the illegal logging domain, and so we present a new model of the defender for green NSGs.

The status quo for monitoring these protected areas is to conduct patrols. Therefore we allow the defender to conduct patrols in the network to interdict the attacker by placing resources on edges of the graph, as shown in Figure 3.2. The defender has a set of $\mathcal{R}$ resource types r each of which can conduct a patrol along L_r connected edges. Multiple environmental factors can cause a resource to fail in detecting an attacker, such as limited visibility or collusion with the adversaries. We model this using an interdiction probability P_r for each resource type. The defender has a total budget B; each resource type has a cost b_r, and a team consists of m_r resources of type $r \in \mathcal{R}$. Multiple resources placed on a single edge results in a higher probability of detecting the attacker, which models coordination among resources.

A defender pure strategy X_i is an allocation of all resources in a given team to a set of edges of the graph, satisfying the connectedness and length constraints for each resource. An attacker pure strategy A_j is any path starting at a source node s_j and ending at a target

node v_j. Figure 3.2 shows three attacker pure strategies. Although these strategies take the shortest path from source to target, it is not a requirement in the game. The allocation of one resource of size $L_r = 2$ is show, which intersects paths i and j. The attacker and defender can play mixed strategies **a** and **x**, i.e., probability distributions over pure strategies. The probability of detecting an attacker on edge e if the defender follows a pure strategy X_i, allocating $m_{r,e}$ number of resources of type r to edge e is given in Equation 3.2.

$$P(e, X_i) = 1 - \prod_{r \in \mathcal{R}} (1 - P_r)^{m_{r,e}} \tag{3.2}$$

G(N,E)	Graph representing security domain
G^c	Compact Graph representing security domain
$\tau(t_i)$	Payoff of the i^{th} target v_i
$\mathcal{R}$	Set of defender resource types
L_r	Number of edges covered by resource type r
b_r	Cost of resource type r
P_r	Detection probability of resource type r
B	Total budget for the team
m_r	Number of defender resources of type r
X = $\{X_i\}$	Set of defender pure strategies
x = $\{x_i\}$	Defender's mixed strategy over X
A = $\{A_j\}$	Set of attacker pure strategies
a = $\{a_j\}$	Attacker's mixed strategy over A
$U_d(X_i, \mathbf{a})$	Defender utility playing X_i against **a**
$U_a(\mathbf{x}, A_j)$	Attacker utility playing A_j against **x**

Table 3.1: Notation and Game Description

The total probability that a defender pure strategy X_i protects against an attacker pure strategy A_j is given by the probability intersection function in Equation 3.3, where we take the product over all the edges in the attack path.

$$P(X_i, A_j) = 1 - \prod_{e \in A_j} (1 - P(e, X_i)) \tag{3.3}$$

The attacker obtains a payoff of $\tau(v_i)$ if he successful in reaching a target, and a payoff of zero he is caught. We assume a zero sum model, so the defender receives a penalty opposite of the attacker's payoff. For this zero-sum game, the optimal defender mixed strategy is the well-known minimax strategy. The game value is denoted $F(t)$, and is function of a team of resources t composed of some set of resources $\mathcal{R}(t)$.

The strategic aspect of the SORT problem can be formulated as the optimization problem in Equation 3.4, where the utility $F(t)$ is maximized subject to budgetary constraints.

$$\max_{t \in \mathcal{T}} \left\{ F(t) : \sum_{r \in \mathcal{R}(t)} b_r \leq B \right\} \tag{3.4}$$

3.3 Solution Approach

3.3.1 FORTIFY: The Hierarchical Search

In NSGs $F(t)$ is computationally difficult to calculate, because it requires finding the optimal tactical allocation to assess the utility of a given team t. Since there are an exponentially many possible teams, the sequential approach of evaluating $F(t)$ exactly for every

team and picking the best one is impractical. Instead, in our approach to SORT, we integrate the analysis of the strategic and tactical aspects of the problem to search the space of teams much more efficiently. We use fast methods to quickly evaluate upper bounds on the utilities for specific teams. Using these bounds we select the most promising team to evaluate in more detail, iteratively tightening the bounds as the search progresses until the optimal team is identified.

Algorithm 1: FORTIFY

```
1:  procedure FORTIFY(B, R)
2:      T^B ← getTeams(B, R), T_c^B = ∅, T_r^B = ∅, T_*^B = ∅
3:      for each t ∈ T_B:
4:          t.value ← Compact Layer(t)
5:          T_c^B ← T_c^B ∪ {t}
6:      repeat:
7:          t_max ← arg max_{t.value}(T_c^B, T_r^B, T_*^B)
8:          if (t_max ∈ T_*^B) return t_max
9:          else: t.value ← NextLayer(t_max)
10:             T^B_NextLayer ← T^B_NextLayer ∪ {t_max}
```

FORTIFY uses a three layer hierarchical representation NSG to evaluate the performance of teams at different levels of detail. Starting from the full representation of the game, each layer abstracts away additional details to approximate the game value $F(t)$. We call the bottom later without any abstraction the Optimal Layer, the Reduced Layer is in the middle, and the Compact Layer is the most abstract.

FORTIFY is described in Algorithm 1 and Figure 3.3. Initially (line 1), we enumerate all teams $\mathcal{T}_B$, that maximally saturate the cost budget B (so that no additional resources can be added to the team). Each team $t \in \mathcal{T}_B$ proceeds through the layers of the hierarchy by being promoted to the next layer; the first team to make it through all layers is the optimal

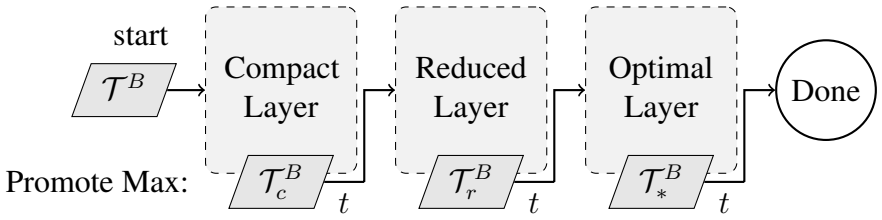


Figure 3.3: Flowchart for the Compact Search Algorithm. $\mathcal{T}^B$ is the initial set of teams. $\mathcal{T}_c^B$, $\mathcal{T}_r^B$ and $\mathcal{T}_*$ are the sets of teams which have passed through the compact, reduced and optimal layers. After all teams pass through the compact layer one team (with max value) is promoted at each step.

team. When a team is promoted to a new layer, it is evaluated to compute a tighter upper bound on the value based on the abstraction specified for that layer.

At the start of the algorithm we have no information on the team values, so each team is evaluated and ordered based on the Compact Layer (line 2-3). Next, the team with the highest bound is promoted to the Reduced Layer (line 9). This team can then be again promoted to the Optimal Layer or another team can be promoted to the Reduced Layer. The next team to be promoted is always the one with the highest current upper bound on the value, regardless of which layer it is in. When a team is evaluated in the Optimal Layer we know the true value of the team, so if this value is higher than the upper bounds on all remaining teams this team must be optimal (line 8), and the algorithm terminates.

3.3.2 Tactical Planning: Optimal Layer

We first introduce the optimal layer of FORTIFY in order to explain how the tactical deployment problem of the full NSG is solved. This step is computationally expensive as there are an exponential number of attacker and defender strategies and explicitly

enumerating them all in computer memory is infeasible. Incremental strategy generation addresses the first challenge, allowing us to obtain the optimal defender mixed strategy without enumerating all pure strategies [McMahan et al., 2003]. This approach decomposes the problem into (1) master MiniMax linear program (LP) and (2) oracles for the defender and attacker which incrementally generate pure strategies to add to the current strategy space via a separate optimization problem. The master LP computes a solution restricted to the set of pure strategies generated by the oracles. The steps are shown in Algorithm 2. The formulation for the MiniMax LP standard, but we provide new formulations for both oracles. The key novelty in our formulations is that the model complexities require us to not only handle heterogeneous resource types and coordination of multiple resources, but failure probabilities on edges imply that we have a non-linear optimization problem to solve in both oracles. We address this by constraining the variables to be binary valued, and we take advantage of efficient constraint programming methods in commercial tools like CPLEX.

Algorithm 2: Optimal Layer

```
1:  procedure OPT(t)
2:      Initialize X, A
3:      do:
4:          (U_d^*,x, a) ← MiniMax(X, A)
5:          X ← DefenderOracle(a)
6:          X ← X ∪ { X }
7:          A_j ← LinearAttackerOracle(x)
8:          if U_a(x, A_j) - U_a(x, a) ≤ ε then
9:              A_j ← AttackerOracle(x)
10:         A ← A ∪ { A_j }
11:     until convergence then return (U_d^*,x)
```

Minimax Linear Program: The game value is computed (line 3) by solving for the minimax strategy using a LP formulation (3.5). The inputs are the current set of attacker and defender pure strategies, **A** and **X** and the outputs are the game utility U_d^*, and mixed strategies, **x** and **a**. $U_d(\mathbf{x}, A_j)$ is the defender utility playing **x** against the pure strategy A_j.

$$\max_{U_d^*, \mathbf{x}} \quad U_d^* \quad \text{s.t.} \quad U_d^* \leq U_d(\mathbf{x}, A_j) \quad \forall j = 1 \ldots |A| \tag{3.5}$$

Defender Oracle: The defender oracle returns the best response strategy X_i to add to the MiniMax LP. The objective is to maximize the utility expressed in Equation (3.6), given an input distribution **a** over the current attacker strategies **A**, where a_j the probability of attacker taking path A_j.

$$U_d(X_i, \mathbf{a}) = -\sum\nolimits_j a_j(1 - P(X_i, A_j))\tau(v_j) \tag{3.6}$$

A pure strategy implies a single allocation of the given team's resources. Resources are allocated by setting the binary decision variables $\lambda_{m,e}^r \in \{0, 1\}$ which corresponds to the m^{th} resource of type r being allocated to edge e. Our contributions formalize the constraints needed to accommodate arbitrary path coverage as well as failure probability. Path constraints are enforced with $\sum_e \lambda_{m,e}^r = L_r$ and in Equations (3.7-3.8). Equation (3.7) ensures every allocated edge is connected to at least one other edge. Since the number of nodes in any path of length L_r should be $L_r + 1$, Equation (3.8) counts the number of nodes which are either a source or target of allocated edges, making sure not to double count nodes which belong to multiple edges. $\lambda_{n,m}^r \in \{0, 1\}$ and equals 1 if a node n is the source or target of any allocated edge $\lambda_{e,m}^r = 1$.

$$\lambda^r_{m,e} \leq \sum_{e1 \in in(n_s)} \lambda^r_{m,e1} + \sum_{e2 \in out(n_t)} \lambda^r_{m,e2} \quad \begin{smallmatrix} n_s \leftarrow \text{source(e)} \\ n_t \leftarrow \text{target(e)} \end{smallmatrix} \quad \text{if} \quad L_r \geq 1 \tag{3.7}$$

$$\lambda^r_{m,e} \leq \lambda^r_{m,n} \quad \text{s.t. } n \leftarrow \begin{smallmatrix} \text{source(e)} \\ \vee \text{ target(e)} \end{smallmatrix} \quad \sum_n \lambda^r_{m,n} = L_r + 1 \tag{3.8}$$

Attacker Oracle: The attacker oracle computes a best response strategy A_j which maximizes his utility (Equation **??**), playing against the defender mixed strategy **x**. An optimization problem in the form of (**??**-**??**) is solved for each target t_j; the best path for each target is computed and the target with the highest utility is chosen. The decision variables $\gamma_e \in \{0, 1\}$ are binary and correspond to edges $e \in A_j$.

$$\mathcal{A}_j := \left\{ \gamma_e : \begin{array}{ll} \sum_{e \in out(s)} \gamma_e = 1 \quad \sum_{e \in in(v*)} \gamma_e = 1 & \\ \sum_{e \in in(n)} \gamma_e = \sum_{e \in out(n)} \gamma_e & \forall n \in N, n \neq s, v \\ \gamma_e \in \{0, 1\} & \forall e \in E \end{array} \right\},$$

$$\begin{aligned} \max_{\gamma} \quad & \sum_i x_i (1 - P(X_i, \gamma)) \tau(v_j) \\ \text{s.t.} \quad & P(X_i, \gamma) = 1 - \textstyle\prod_{e \in E} \gamma_e (1 - P(e, X_i)) \\ & \gamma_e \in A_j \end{aligned}$$

Exactly solving the Attacker Oracle is computationally expensive. Therefore, in line 6, we introduce a new **Linear Attacker Oracle** approximation to quickly generate an approximate best response. Here the probability intersection function is approximated with an additive linear function, $P(X_i, A_j) = \sum_{e \in A_j} P(e, X_i)$ so we can write the oracle as an LP. (In the attacker oracle, the value of $P(e, X_i)$ does not need to be approximated, as it does not depend on attacker's decision variables, but rather on defender's variables

and thus is calculated outside the attacker oracle.) In the event that the approximation steps fail to generate a strategy that increases the oracle's expected utility (line 7), the oracle computes the optimal solution as a final step (line 8) to ensure that the algorithm converges to the true game value.

3.3.3 Compact Layer

The compact layer uses an *abstract representation* of the game model which reduces the problem in two ways: (1) the attacker is restricted to using *only a subset* of possible paths, (2) the defender chooses to allocate resources *directly to attacker paths* rather than edges in the graph.

Algorithm 3: Compact Graph

```
1:  procedure CompactGraph(G(N, E))
2:      for each s_i ∈ N, v_j ∈ N:
3:          {E_j} ← mincut(s_i,v_j)
4:          for each e ∈ E_j:
5:              A^c ←ShortestPath(s_i, e, t_j)
6:              A^c ← A^c ∪ {A^c}
7:              N^c ← N^c ∪ newNode(A^c)
8:      for each A^c_i ∈ A^c, A^c_j ∈ A^c:
9:          w_{i,j} ← D(i, j)
10:         G^c ← newEdge(i, j, w_{i,j})
11:     return G^c
```

Formally, the compact layer constructs a new graph $G^c(N^c, E^c)$ in Algorithm 3, where the attacker paths are represented by nodes N^c of the graph. We describe this using an example, transforming part of the graph from Figure 3.2 into it's compact representation in Figure 3.4. To choose the subset of attacker paths, for each source-target pair of nodes

we (1) calculate the min-cut for each target, and (2) find the shortest possible paths from source to target that go through each of the min-cut edges (lines 1-6). The three attacker paths A_i, A_j and A_k in Figure 3.2 are among several calculated from the min-cut of the graph. These three paths become nodes i, j, and k respectively, in part of the new compact graph. In order for a resource to cover paths i and j it's path coverage L_r must be at least as large as the minimum separation distance between the paths, plus any edges required to intersect the paths. We define this as $D(i, j)$, which for the example in Figure 3.2 is 3. Edges are added between any nodes i and j with weight $D(i, j)$, equal to the L_r required to cover both corresponding paths. These distances are calculated using Dijkstra's algorithm, and no edges are added between two nodes if the distance is greater than the largest path coverage of any defender resource.The defender can choose to cover any subset of nodes in G^c with a resource of type r as long as the induced subgraph has the property that (1) the subgraph is fully connected and (2) all edges have weight less than L_r. For example, the three paths in Figure 3.4 (i-j-k) can be all covered by a resource of size 4. If the defender has a resource of size 3, she can only cover paths (i-j) or (j-k).

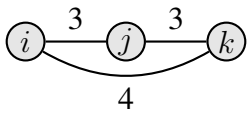


Figure 3.4: Compact Graph

The compact layer solves this abstract representation of the game for a single team. The problem is decomposed into master MiniMax and a single Defender Oracle. There is no oracle for the attacker, as the attacker's strategy space is enumerated using the compact

graph and fixed at the start of the algorithm. The game value is calculated using Algorithm 4. The compact graph and subset of attacker paths are first generated (line 1). The attacker's mixed strategy is initialized with a uniform distribution (line 2). The compact defender oracle continuously adds strategies to the master LP until the defender's value cannot be improved. Convergence occurs when the oracle can no longer find any strategy to add that will improve the defender's utility (line 7).

Compact Defender Oracle The same objective function (Equation 3.6) is maximized, however the constraints are modified to reflect the compact game representation. $P(X_i, A_j)$ is linearly approximated by Equation 3.9 and is capped at 1. Here, we want to conservatively over-estimate the defender's interdiction probability to ensure that the compact layer returns an upper bound. Therefore, when a defender resource covers a node in G^c, we assume that the corresponding attacker path is interdicted by the entire entire patrol of length L^k of that resource. The probability of catching the attacker on the compact graph is set to $(1-(1-P_k)^{L_k})$).The defender chooses to allocate the m^{th} resource of type k to a node n_j corresponding to attacker path A_j by setting the decision variables $\eta^k_{j,m} \in \{0,1\}$.

$$P(X_i, A_j) = \sum P^c_k \eta^k_{j,m} \qquad D(i,j)\eta^k_{i,m}\eta^k_{j,m} \leq L_k \tag{3.9}$$

Lemma 1 *Playing against the same attacker strategy, the Optimal Defender Oracle's strategy space is a subset of the Compact Defender Oracle strategy space.*

Proof. The strategy space of an oracle is defined by set of feasible assignments of values for their decision variables which determine the values which can be achieved for their utility function, where the best response strategy is the strategy which maximizes

Algorithm 4: Compact Layer

procedure COMPACTLAYER(t)
  $G^c \leftarrow$ CompactGraph(G(N,E))
  Initialize mixed strategy **a** $\leftarrow$ Uniform(A^c)
  do:
    $X_i^c \leftarrow$ CompactDefenderOracle(**a**)
    $\mathbf{X^c} \leftarrow \mathbf{X^c} \cup \{X_i^c\}$
    (**x, a**) $\leftarrow$ MiniMax($\mathbf{X^c}$, $\mathbf{A^c}$)
  until convergence $U_d(X_i^c$, **a**) - U_d(**x, a**) $\leq \epsilon$
  return U_d(**x, a**)

this function. Assume that the he set of feasible assignments in the Defender Oracle strategy space is smaller, and contained within the strategy space of the Compact Oracle. When the oracles are asked to compute a best response, one of two things can happen: Since the unrestricted strategy space always contains the restricted strategy space (1) either both oracles will return the best response if it exists in the restricted strategy space or (2) if the best response exists outside the restricted strategy space the restricted oracle will return a suboptimal strategy while the unrestricted oracle will return the best response. In the first case both oracles achieve the same utility, while in the second the restricted oracle will achieve a lower utility. The same logic can be applied to the Attacker Oracle playing with a full strategy space and when restricted to use only the min-cut paths.

We now compare the game values played in the Optimal Layer (Defender Oracle vs Attacker Oracle), the Reduced Layer (Defender Oracle vs Restricted Attacker Oracle) and the Compact Layer (Compact Oracle vs Restricted Attacker). By the previous statement, it is obvious that the utility of Reduced Layer, the Defender Oracle playing against the restricted Attacker Oracle will be greater than the Optimal Layer, if it were playing against

the unrestricted attacker oracle. Similarly, the Compact Layer, will always achieve a better utility than the Reduced Layer. Since all oracles are still best response oracles (always selecting the best available strategy in their strategy space), even though they may play different strategies, the game played with the compact layer will always converge to a greater value than the optimal layer. □

Theorem 2. *If the Compact Oracle strategy space contains the full strategy space of the DefenderOracle, then the game value of the Compact Layer will always upper bound the true game value.*

Proof. We show any optimal strategy can be represented in the compact strategy space. The true number of resources of type k which can cover path A_j is the sum of all resources N_S^k assigned to cover any subset of paths S which include path A_j, making sure not to double count. The compact representation only considers pairwise distances between paths, and approximates any terms where $|S_i| > 2$ as the number of pairs of paths $\{a, b\}$ which can be reached from path j.

$$\sum_{D(S_i)\leq L_k} N_{S_i}^k \quad \text{s.t.} \quad j \in S \quad \rightarrow \quad N_j^k + \sum_{\substack{a\neq b\neq j \\ D(j,a),D(j,b),D(a,b)\leq L_k}} N_{a,b}^k \tag{3.10}$$

LHS: Each $N_{S_i}^k$ belongs to S_i,the set of i attacker paths which can be covered by a resource of size L_k. The sum then counts the size of the union $\bigcup_{i=3}^{\infty} S_i$. **RHS:** Each pairwise assignment $N_{\{a,b\}}^k$ belongs to $S_{\{a,b\}}$, the set of pairwise attacker paths A_a and A_b which satisfy the condition $D(j,a) \wedge D(j,b) \wedge D(a,b) \leq L_k$. The distance functions D are constructed to use the minimum distance needed to cover a set of paths so that $D(S) \leq D(S')$ if $S \subseteq S'$. Therefore if S' can be covered by a path of size L_k so can S. This

means that any pair of attacker paths in $\bigcup_{i=3}^{\infty} S_i$ must also be in the set $S_{a,b}$ Any assignment of resources on the LHS can be represented on the RHS, therefore any strategy (coverage of paths) which is feasible in the defender oracle strategy space must be feasible in the compact oracle's strategy space. □

3.3.4 Reduced Layer

Algorithm 5: Reduced Layer

```
1: procedure REDUCEDTLAYER(t)
2:     Initialize mixed strategy a ← CompactLayer(A^c)
3:     do:
4:         X_i^c ← DefenderOracle(a)
5:         X^c ← X^c ∪ {X_i^c}
6:         (x, a) ← MiniMax(X^c, A^c)
7:     until convergence U_d(X_i^c, a) - U_d(x, a) ≤ ε
8:     return U_d(x, a)
```

The Reduced Layer uses the same restricted strategy space for the attacker as the Compact Layer. However, the defender uses the original, unrestricted strategy space to allocate resources. While the reduced layer is more difficult to solve than the compact layer, it allows us to iteratively tighten the upper bounds and avoid more computation in the Optimal Layer. The evaluation of teams in this layer follows Algorithm 3. We additionally reduce the computation effort spent in this layer by warm starting the attacker's mixed strategy with the solution from the compact layer.

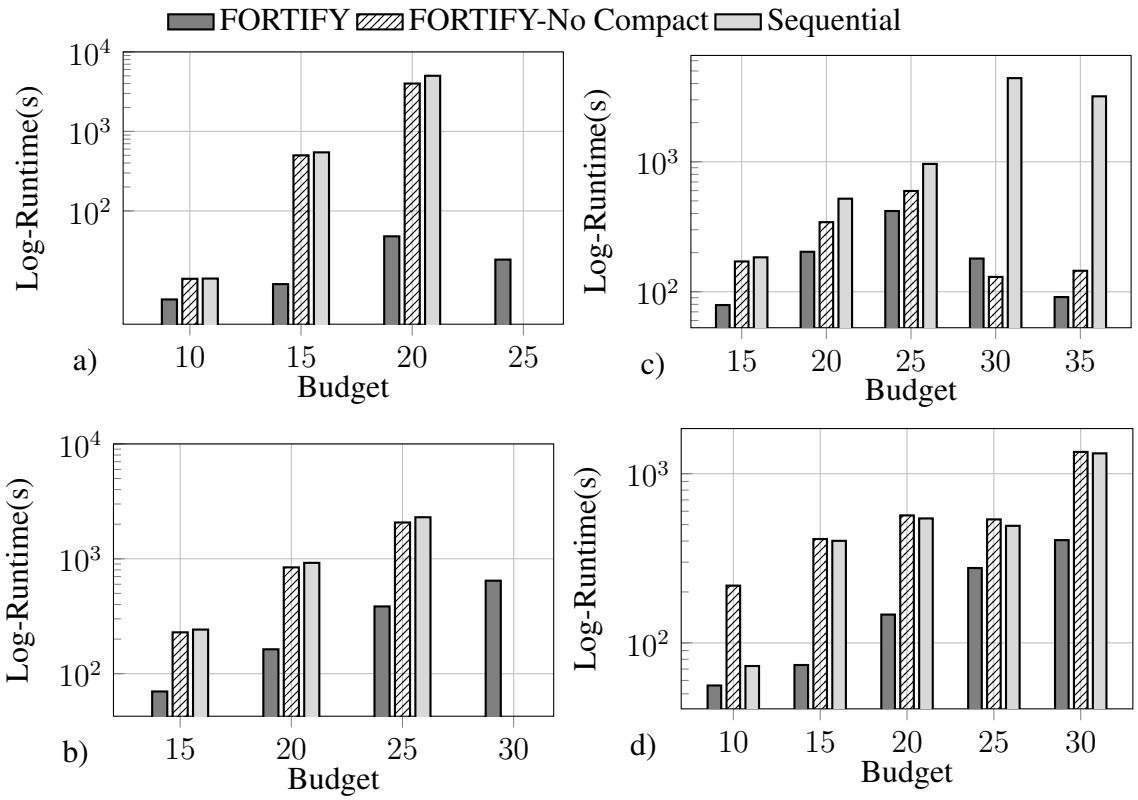


Figure 3.5: Runtime scalability comparing FORTIFY against Sequential and No-Compact method. (a) Λ^1 teams on a $G_{5,20,5,5}$ graph. (b) Λ^2 teams on a $G_{4,4,4,4}$ graph. (c) Λ^1 teams on a $R_{70,5,5,0.1}$ graph. (d) Λ^2 teams on a $R_{25,4,4,0.1}$ graph.

3.4 Evaluation

We present four sets of experimental results: (1) We evaluate the scalability and runtime performance of FORTIFY on several classes of random graphs. We benchmark with a sequential search which sequentially evaluates enumerated teams with cost saturating the budget. (2) We also evaluate the impact of the initial compact layer on the runtime by comparing the runtimes of FORTIFY with and without the compact layer. (3) We investigate the benefit of optimizing team composition as well as the diversity of optimal teams and

(4) we demonstrate that FORTIFY can scale up to the real world by testing performance on a case study of Madagascar using real graph data. All values are averaged over 20 trials. The experiments were run on a Linux cluster with HP-SL250, 2.4 GHz, dual-processor machines. We use the following graphs:

(1) Grid graphs Labeled $G_{w,h,s,t}$ consist of a grid with width w, height h, sources s, targets t and nearest neighbor connections between nodes. We define start and end points for the attacker, with sources located at one end and targets at another.

(2) Geometric graphs provide a good approximation of real road networks [Eppstein and Goodrich, 2008] allowing us to model the networks of villages and rivers in forest regions. n nodes are distributed randomly in a plane and are connected based on their distance which determines the density d of the graph. We label them $R_{n,s,t,d}$.

3.4.1 Scalability

We first evaluate the performance of FORTIFY using two sets of resource types, and target values of 50. Each set contains 4 resource types, with varying costs of $b = \{5, 6, 7, 8\}$. The first set of resource types $\mathcal{R}^1$ have varied path coverages $L^1 = \{1, 2, 3, 4\}$ and constant detection probability $P^1 = \{1, 1, 1, 1\}$ while the second set $\mathcal{R}^2$ has constant path coverage $L^2 = \{2, 2, 2, 2\}$ and varied detection probabilities $P^2 = \{0.5, 0.6, 0.7, 0.8\}$. Experiments that did not terminate in 3 hrs were cutoff and are shown as missing bars. Figure 5.4 show the runtimes for our algorithms run on both graph types for both $\mathcal{R}^1$ and $\mathcal{R}^2$ teams. The budget varies on the x-axis and the runtime is shown on the y-axis in log scale. FORTIFY consistently outperform the sequential method; on both the grid and geometric graphs. FORTIFY performs particularly well on the grid graphs, and scaling past budgets of 25

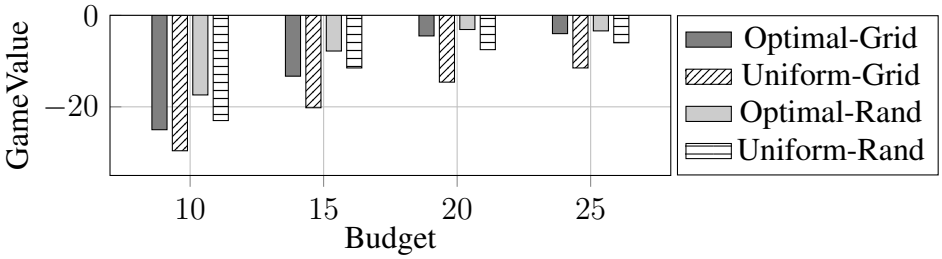


Figure 3.6: Team optimization comparison. Teams have 6 resource types, and vary both edge coverage L=$\{2, 2, 5, 3, 3, 6\}$, and detection probability P=$\{0.7, 0.9, 0.7, 0.6, 0.6, 0.6\}$ with costs b=$\{5, 8, 10, 5, 8, 10\}$.

while all instances of the sequential search were cut off. We observe a peak in the runtime for teams with perfect detection probability in 5.4 (a) and (C) around a budget of 20-25, which is due to the deployment vs saturation phenomenon which occurs in these types of network models [Jain et al., 2012a].

Removing the Compact Layer: We also compare the performance of FORTIFY with and without the compact layer in Figure 5.4. It is apparent that this layer is crucial to the performance of the algorithm, particularly for the grid graphs in parts (a-b) as FORTIFY without the compact layer performs almost as poorly as the sequential method. In fact, removing the compact layer can cause FORTIFY to perform worse than the sequential method for small budgets due to the overhead required for the approximation.

3.4.2 Team Composition

We demonstrate the value of optimizing over team composition by looking at the loss in game value incurred by allocating all of the budget to a random set of resources. Here we do this by randomly selecting resources until there is no more remaining budget. This is done 30 times, so that we are averaging over a set of 30 randomly chosen teams. The

B	Runtime(s)	GV
10	203	-266
15	388	-256
20	653	-239
25	1308	-230
30	1742	-220
35	2504	-216
40	3675	-204

Table 3.2: Runtime on Madagascar Graph

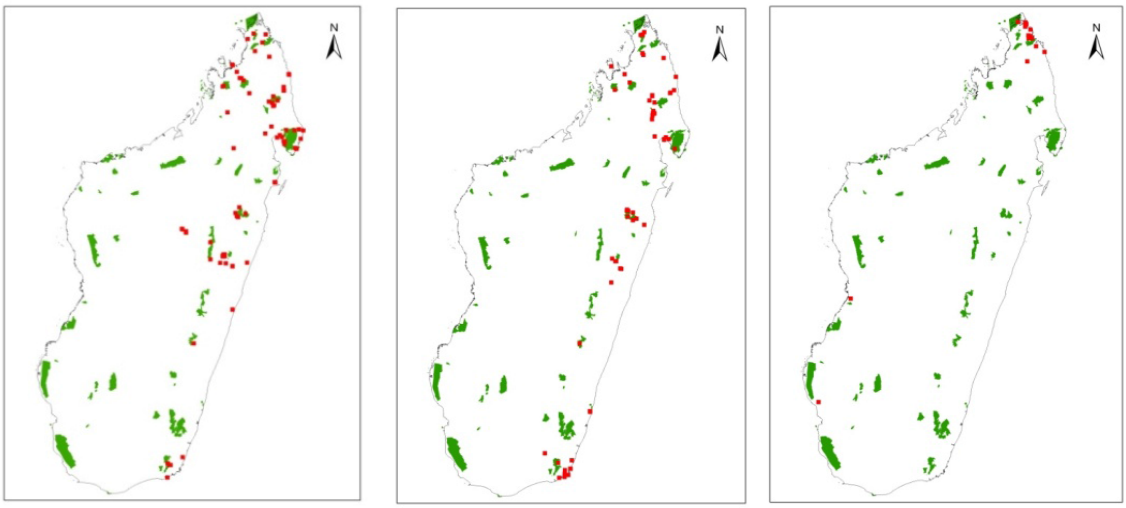
teams and resources are pre-processes so that there are no obviously dominant resources (i.e. resource which has a high coverage and lower cost) and no obviously dominant teams (no teams which are subsets of other teams, and not teams with a total higher cost and lower coverage). Games are played on $G_{4,4,4,4}$ and $R_{25,4,4,0.1}$ graphs with target values of 50. The results are shown in Figure 3.6 with budget on the x-axis and game value on the y-axis. As expected, the game value decreases with budget as we form larger teams, however the relative benefit increases as well, with almost a 300% loss in solution quality at budgets of 25 without our team optimization algorithm. This is due to the increase in the space of possible teams which can be formed, making it more likely to form a suboptimal team.

3.4.3 Real World : Madagascar National Parks

We demonstrate the ability of FORTIFY to scale up to real world domains, evaluating the performance on a network constructed from GIS data of at-risk forest areas in Madagascar. We present the following model which was built working closely with domain experts from AVG.

The area is 78 139 ha large and situated between the Marimbona river in the south and the Simianona river in the north. The communities in this area are Ambahoabe d?Antenina and d?Andapafito. The Madagascar National Police (MNP) forest guards earn 8.000Ar per day when doing the patrols and they patrols 20 days per month. The equipment of these forest guards is a kind of camping equipment. In each sector there is a little cabin etc., a sector has an average size of 26.000ha, so each forest guard has to survey an area of about 3.700ha. Usually during a patrol mission on average they travel 20 km a day. They usually follow their habitual routes, but of course if they get specific information, they change their route for investigations. Additionally the MNP is establishing a partnership system with the neighboring communities to improve the surveillance in the most retracted areas. These local community park agents shall help to improve surveillance of the park in the areas surrounding there villages and they are supposed to work 5 days per month. They get paid 5.000 Ariary per day and MNP supports them in developing micro projects for improving their revenues. While these volunteers can detect any illegal activity, they cannot arrest loggers like the forest guard [Dr. Georg Jaster Private Communications].

Graph: Figure 3.1 shows the road and river networks used by the patrolling officers, as well as the known routes taken by groups of illegal loggers. We used this to build the nodes and edges of our network. Edges correspond to distances of 7-10 km. 10 target locations were chosen by clustering prominent forest areas. 11 villages in the surrounding area were chosen as sources. We have data on locations of different types of trees, which can be seen in Figure 3.7. Several domain experts identified their corresponding risk level and level of attractiveness for logging, based on the size of the forest, the ease of access and the value of the trees. Using this information we assigned values ranging from 100 to 300 to each of the targets.



Diospyros haplostylis *Diospyros gracilipes* *Diospyros aculeata*

Figure 3.7: Geographical distribution of some species of Diospyros trees in Madagascar with a wide distribution (Diospyros haplostylis, Diospyros gracilipes) and restricted distribution (Diospyros aculeata). Tree locations are shown in red, while green areas correspond to conservation areas [UNFCCC, 2016].

Resources: Communal policemen and local volunteers conduct patrols in the forest. A typical patrol covers 20 km in a day and patroller can conduct two types of patrols, a short patrol covering 2 edges and a long patrol covering 3 edges. Based on expert input, we assign the detection probability for communal police as 0.9 for short patrols and 0.8 for long patrols; and for volunteers, 0.7 for short patrols and 0.6 for long patrols. The lower probabilities for volunteers are because they must call backup for interdiction, which may allow the adversary to escape. Thus, in total we have 4 resource types available $L = \{2, 3, 2, 3\}$, $P = \{0.7, 0.6, 0.9, 0.8\}$. The costs are proportional to the salaries patrollers receive for a day of patrolling $b = \{5, 5, 8, 8\}$.

Experiment: The runtime experiments are shown in Table 3.2 for increasing budgets. Data is averaged over 20 runs. FORTIFY can scale up to real world networks, able to

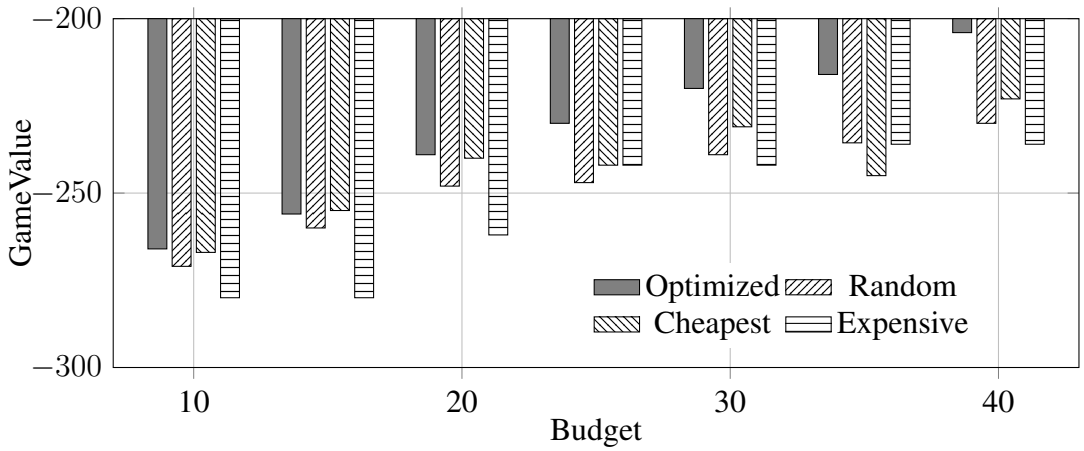


Figure 3.8: Team optimization on Madagascar Graph

handle both the large graph size and number of source and target nodes, even for large budgets. The value of performing this optimization is shown in Figure 3.8 with the solution quality (game value) on the y-axis and budget on the x-axis, where we compare the optimal game value to the average value achieved by randomly allocating our total budget. Here again we preprocessed the resources and teams so that there were no obviously dominating teams or resources, and we average this over a set of 30 teams. We also compared to two other heuristics that may be used to select teams, where we compared to a team made of all the cheapest resources with the highest detection probability, and the most expensive resource with the highest detection probability. The comparison agains the most expensive resource represents the current status quo for patrolling, as it is currently only the forest guard that patrol. Here again we see significant utility gains when strategically optimizing the resources in the team.

3.5 Chapter Summary

In this chapter I present a fundamentally new problem in Security Games– SORT– which addresses the strategic planning challenge of optimal investment and deployment of security resource teams. I provide a formal definition of this problem as well as analysis and hardness results of solving this problem for general security games. Motivated by the rising challenges in environmental sustainability I address this challenge for a broad class of games known as Network Security Games using the threat of illegal logging in Madagascar as a motivating domain where we must work with a limited budget to coordinate and deploy teams of resources. I extend the model of NSG to include teams of heterogenous resources, and provide a novel double oracle formulating for solving the underlying tactical deployment problem. I also develop FORTIFY, a scalable branch and bound style solution approach for solving the SORT problem, which uses hierarchical abstractions of the game in order to compute the optimal strategic design parameters, here being teams of resources, for the game. This method is fast and scalable, with the ability to both model and solve real world problem instances. Thus FORTIFY provides a valuable tool for AVG, and other environmental protection agencies.

Chapter 4

Multistage Strategic Planning

In this section I describe the extension of the problem to the multistage setting. We are motivated here by domains where we are tasked with making repeated strategic level decision and where due to changing domain features, it becomes necessary to re-evaluate and adapt to new information. This may to due to fluctuations in time dependent processes or the realization of uncertain parameters in the problem. As such the optimal strategy is then a policy, an adaptive strategy determined not only by their interactions with the adversary, but also conditioned on the realization of the uncertain parameters and changing features of the problem.

We can extend the problem of strategic planning and the SORT problem to the multistage setting, where here the design variables may vary with time so that $y_{d,w}$ corresponds to the allocation of design resource $d \in \mathcal{D}$ at time period w and the set of constraints on the resources may not now change with time $G_r(y_w)$.

Example 2. *Consider again the problem of choosing a subset of resources to purchase given budgetary constraints, but where the budget for investment in these resource comes in installments, so that in each time period w we are asked to make investment decisions given a budget B_w. Resource costs may also change over time so that the use of each resource costs $b_{k,t}$, and we want to determine what the optimal investment strategy should*

be in each time period, given that our decisions now may affect the utility of future investments. Additionally we may condition our decision on the realization of unknown variables ξ from previous time periods. The multistage SORT problem can be formulated as:

$$\max_{t \in \mathcal{T}} \left\{ E[F(t, \xi)] : \sum_{r \in \mathcal{R}(t)} b_{r,w} \leq B_w \forall\ w \right\}$$

Where the expectation operator $E[\dot{]}$ is taken with respect to the underlying probability distribution for ξ.

4.1 Examples of Multistage SORT in the Real World

While the green security game model described in Chapter 1 may also be formulated as a multistage game, the time scale over which each deployment of the resources occurs . When games move at a much quicker pace it becomes much more important to consider multistage models. Since the environment in such games changes much more rapidly we may want to design strategies which adapt to information about the environment, and so multistage models become more appropriate for such settings.

Such problems arise frequently in cyber-security where the time scale of events is much shorter than in physical security. In network security a big challenge is the problem of active sensing in a computer network, where the defender observes a stream of suspicious alerts (or absence of alerts), and is tasked with inferring if an attack is taking place, and determining the best response policy. In this domain, the defender may have many kinds of security resources available to monitor network traffic, where such resources have varied efficiencies at detecting attacks. Resources may not only miss attacks, but may

falsely identify legitimate network use as an attack, thus information from these resources is inherently noisy and the deployment of each resource does not provide a high confidence estimate about the security state [Sommer and Paxson, 2010][FarnHam, 2013]. Thus, the defender needs to come up with a sequential plan of actions while dealing with uncertainty in the network and in the alerts, and must weigh the cost of deploying detectors to increase their knowledge with the potential loss due to successful attacks as well as the cost of misclassifying legitimate network use. This problem of active sensing is common to a number of cyber security problems; here I focus on the challenge of detecting and addressing advanced persistent threats (APTs), with particular focus on data exfiltration over DNS queries. I developed a decision-theoretic planning model to reason about noisy observations in order to dynamically allocate security resources such as sensors and determine whether or not the suspicious activity is malicious, and compute the best response policy.

More specifically, I provide a novel Virtually Distributed Partially Observable Markov Decision Process (VD-POMDP) formulation to address the challenge [McCarthy et al., 2016a], where the efficiency of the formulation is based on two key contributions: (i) the problem is decomposed in a way that allows for individual sub-POMDPs with sparse interactions to be created. Individual policies for different sub-POMDPs are planned separately and their sparse interactions are only resolved at execution time to determine the joint actions to perform; (ii) The abstraction in planning step allows for large speedups and scalability, after which a fast MILP is used to implement the abstraction while resolving any interactions. This allows us to determine optimal sensing strategies, leveraging information from many noisy detectors, and subject

to constraints imposed by network topology, forwarding rules and performance costs on the frequency, scope and efficiency of sensing we can perform. I provide conditions under which our methods are guaranteed to result in the optimal joint policy, and provide empirical evidence to show that the final policy still performs well when these conditions are not satisfied. I also provide experimental evaluation of our model in a real network testbed, where I demonstrate the ability to correctly identify real attacks.

4.2 Motivating Domain

Advanced persistent threats can be one of the most harmful attacks for any organization with a cyber presence, as well as one of the most difficult attacks to defend against. While the end goal of such attacks may be diverse, it is often the case that intent of an attack is the theft of sensitive data, threatening the loss of competitive advantage and trade secrets as well as the leaking of confidential documents, and endangerment of national security [Trend Labs, 2013, Intel Security, 2015]. These attacks are sophisticated in nature and often targeted to the vulnerabilities of a particular system. They operate quietly, over long periods of time and actively attempt to cover their tracks and remain undetected. A recent trend in these attacks has relied on *exploiting Domain Name System (DNS) queries* in order to provide channels through which exfiltration can occur [Seth Bromberger, 2011, FarnHam, 2013].These DNS based exfiltration techniques have been used in well-known families of malware; e.g., FrameworkPOS, which was used in the Home Depot data breach involving 56 million credit and debit card information [Rascagneres, 2016].

At a high level, DNS exfiltration involves an attacker-controlled malware inside an organization's network, an external malicious domain controlled by the attacker, and a DNS server authoritative for the domain that is also controlled by the same attacker. The malware leaks sensitive data by transmitting the data via DNS queries for the domain; these queries traverse the DNS hierarchy to reach the attacker controlled DNS server. Attackers can discretely transfer small amounts of data over long periods of time disguised as legitimate user generated DNS queries. Detecting and protecting against such an attack is extremely difficult as the exfiltration attempts are often lost in the high volume of DNS query traffic and any suspicious activity will not be immediately obvious. In both academia and industry, multiple detectors have been proposed to detect DNS exfiltration. However, because of the sophisticated and covert nature of these attacks, detectors designed to protect against these kinds of attacks either often *miss attacks* or are plagued by *high false positive rates, misclassifying legitimate traffic* as suspicious, and potentially *overwhelming a network administrator* with suspicious activity alerts; these issues have been identified with machine learning based detectors [Sommer and Paxson, 2010], pattern matching based detectors [FarnHam, 2013] and information content measuring detector [Paxson et al., 2013].

We focus on the problem of *rapidly determining malicious domains* that could be potentially exfiltrating data, and then *deciding whether to block traffic or not*. In our problem, the defender observes a stream of suspicious DNS based exfiltration alerts (or absence of alerts), and is tasked with inferring which of the domains being queried are malicious, and determining the best response (block traffic or not) policy. Unfortunately, as stated earlier, detectors are inherently *noisy* and each single alert does not provide a high confidence estimate about the security state. Thus, the defender needs to come up

with a sequential plan of actions while dealing with uncertainty in the network and in the alerts, and must weight the cost of deploying detectors to increase their knowledge about malicious domains with the potential loss due to successful attacks as well as the cost of misclassifying legitimate network use. This problem of active sensing is common to a number of cyber security problems; here we focus on the challenge of data exfiltration over DNS queries.

4.2.1 DNS Exfiltration

Sensitive data exfiltration from corporations, governments, and individuals is on the rise and has led to loss of money, reputation, privacy, and national security. For example, attackers stole 100 million credit card and debit card information via breaches at Target and Home Depot [Sidel, 2014]; a cluster of breaches at LinkedIn, Tumblr, and other popular web services led to 642 million stolen passwords [Gooding, 2016]; and the United States Office of Personnel Management (OPM) data breach resulted in 21.5 million records, including security clearance and fingerprint information, being stolen [Barrett, 2015].

In the early days, exflitration happened over well known data transfer protocols such as email, File Transfer Protocol (FTP), and Hypertext Transfer Protocol (HTTP) [Trend Labs, 2013]. The seriousness of the problem has led to several "data loss prevention (DLP)" products from the security industry [Symantec, 2017, McAfee,] as well as academic research for monitoring these protocols [Borders and Prakash, 2004, Hart et al., 2011]. These solutions monitor email, FTP, and other well known protocols for sensitive data transmission by using keyword matching, regular expression matching, and supervised learning.

The increased monitoring of the protocols has forced attackers to come up with ingenious ways of data exfiltration. One such technique used very successfully in recent years is exfiltration over DNS queries [Seth Bromberger, 2011, **?**]. Since DNS is fundamental to all Internet communication, even the most security conscious organizations allow DNS queries through their firewall. As illustrated in Figure 4.1, an adversary establishes a malicious domain, *evil.com*, and infects a client in an organization with malware. To exfiltrate a data file, the malware breaks the file into blocks, b_1, b_2, $\cdots$, b_n, and issues a sequence of DNS queries, b_1.*evil.com*, b_2.*evil.com*, $\cdots$, b_n.*evil.com*. If their responses are not cached, the organization's DNS server will forward them to the nameserver authoritative for *evil.com*; at this point, the adversary controlling the authoritative nameserver can reconstruct the data file from the sequence of blocks.

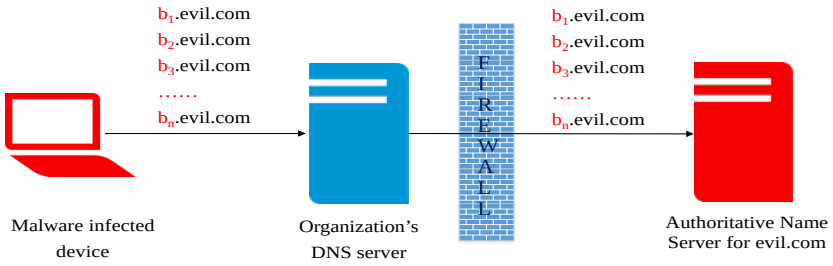


Figure 4.1: Data exfiltration over DNS.

The data transmission is covert and can be accomplished by various means such as a particular sub-domain query meaning bit 1 and another sub-domain query meaning bit 0, or even the timing between queries can leak information. By compressing the data at the client, and by varying query lengths and the time interval between successive queries an adversary can adjust the bandwidth of the communication channel. The adversary could choose to transfer data as quickly as possible (long and rapid domain queries) or slowly (short queries spaced apart in time), depending on the intent behind the attack. To further

hide exfiltration activity, the data blocks can be encrypted by the client before the queries are issued, and decrypted by the adversary. Further, the adversary can encode instructions within its responses to establish a two-way communication tunnel.

Hence building a reliable DNS exfiltration detector is extremely challenging. A recent work on building a detector for DNS exfiltration using measurement of information content of a DNS channel provides techniques that we use to build the low level detector in our problem setting [Paxson et al., 2013]. Apart from this work in academia, there has been some work in the industry that use various heuristics to build low level detectors for DNS exfiltration [FarnHam, 2013]; examples of such heuristics are lengths of DNS queries and responses, sizes of query and response packets, entropy of DNS queries, total volume of DNS queries from a device, and total volume of DNS traffic to a domain. As far as we know, we are the first to build a cost based sequential planning tool that uses the imperfect low level detectors to determine if a domain is involved in exfiltrating data over DNS.

There has been a large amount of work on how to deal with and make decision under uncertainty. Problems such as ours can be well modeled using Partially Observable Markov Decision Process (POMDP) to capture the dynamics of real-world sequential decision making processes, and allow us to reason about uncertainty and compute optimal policies in these types of environments. However a major drawback to these models is that they are *unable to scale* to solve any problem instances of reasonable size. In order to be successful in the cyber domain, such a models needs to be able to handle extremely large problem instances, as networks are often extremely complex, with lots of moving parts. Additionally, due to the salient nature of network states, we need to be able to make *decisions in real time* in order to observe and quickly react to a potential threat.

To address this challenge we make the following key contributions: (1) We provide a formal model of the DNS data exfiltration problem. We propose a new decision making framework using partially observable markov decision processes (POMDPs). (2) We address the scalability issued faced when dealing with large networks by proposing a series of abstractions of the original POMDP. These include using abstract action and observation space. (3) Another step in the abstraction is a *new paradigm* for solving these models by factoring the POMDP into several sub-POMDPs and solving each individual sub-POMDP separately offline; this is motivated by previous work in distributed POMDPs with sparse interactions. We provide techniques for policy aggregation to be performed at runtime in order to combine the abstract optimal actions from each sub-POMDP to determine the final joint action. We denote this model as a *virtually distributed POMDP* (VD-POMDP). We provide conditions under which our methods are guaranteed to result in the optimal joint policy, and provide empirical evidence to show that the final policy still performs well when these conditions do not hold. (4) Finally we provide experimental evaluation of our model in a real network testbed, where we demonstrate the ability to correctly identify real attacks.

4.3 Planning Model

The local computer network can represented as a graph $G(N, E)$, where the nodes N correspond to the set of hosts in the network, with edges E if communication between the hosts is allowed. Each node n has a particular value v_n corresponding to the value of data stored at that computer. At any point in time t each node has a variable traffic volume of requests w_n^t passing through it. We assume there are D domains, where for tractability we

G(N,E)	graph representing network
v_n	value of data at the n^{th} node
$v_{[d]}$	average value of the set of channels to the d^{th} domain
w_n	volume of traffic at the n^{th} node
$w_{[d]}$	total volume of the set of channels to the d^{th} domain
d	the d^{th} domain
X_d	true $\{0, 1\}$ state of the d^{th} domain
M_d	estimated $\{0, 1\}$ state of the d^{th} domain
X	set of all X_d random variables
$c_k =< n, \ldots d >$	k^{th} channel from node n to domain d
$C_{[d]}$	subset of channels ending with the d^{th} domain
C	set of all channels
τ_k	threshold set for channel c_k
a_n	binary variable indicating if node n is sensed or not
z_k	binary variable indicating if channel c_k is sensed or not
Ω_k	$\{0, 1\}$ observation on k^{th} channel
$\Omega_{[d]}$	subset of observations for channels ending with the d^{th} domain

Table 4.1: Notation

assume D is the the number of domains that have ever been queried for the given computer network. DNS queries made from internal nodes in the network are forwarded to special nodes, either access points or internal DNS servers, and then forwarded to external servers from these points. A channel c_k over which exfiltration can occur is then a path, starting at source node, where the query originated, traveling through several nodes in the network and finishing at a target domain d. The total number of channels is K. We use $n \in c_k$ to denote any node in the path specified by c_k.

Let X_d be a random binary variable denoting whether domain d is malicious or legitimate. We assume that a malicious domain will always be malicious, and that legitimate domains will not become compromised; this means that the state X_d does not change with time. Even though legitimate domains get compromised in practice, attackers often

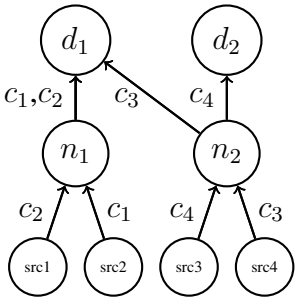


Figure 4.2: Example of a network with two domains, 4 source hosts and 4 channels. Channels $c1$, $c2$, $c3$ go from sources 1, 2, and 3 to domain d_1 while channel c_4 goes from source 4 to domain d_2. We may consider the situation where we can only turn on one detector at any time step, either at node n_1 or n_2, and choose to sense on channels $\{c_1, c_2\}$ or $\{c_3,c_4\}$. We can additionally chose thresholds τ_j for each channel. Each source host has a value v_n and each node n has traffic volume w_n.

use new malicious domains for DNS exfiltration since an attacker needs to control both a domain and the authoritative name server for a domain to successfully carry out exfiltration. In other words, legitimate domains that get compromised are rarely used in DNS exfiltration. Hence in our model, it is reasonable to assume that domains don't change their states. We call the active sensing problem, the challenge of determining the values of X_d. In order to do this we may place detectors at nodes in the network; the state of a detector (off/on) at any node in the network is $a_n \in \{0, 1\}$. Each detector monitors all the channels passing through that particular node, i.e., all $c_k : n \in c_k$. We use the binary variable z_k to indicate if channel c_k is monitored. We can set discrete thresholds individually for each channel; lower thresholds correspond to higher sensitivity to information flow out of any particular channel. Because each channel is associated with a domain, we set a threshold τ_k for each channel. We use $|\tau|$ to denote the number of discrete threshold choices available. We then get observations in the form of alerts for each channel $\Omega_k \in \{0, 1\}$. The probability of receiving an alert for any channel is characterized by some function $\alpha(\tau_k)$

if the channel is malicious and $\beta(\tau_k)$ if the channel is legitimate. Finally, the defender classifies the state of domain d as malicious or legitimate, indicated by M_d.

4.3.1 The POMDP Model

Our POMDP model is a tuple (S, A, T, Ω, O, R) with state space S, action space A, state transition function T, observation space Ω, observation probabilities O and reward function R. Additionally define the average value of the channels to domain d as $v_{[d]} = \sum_{n:n\in C_{[d]}} \frac{v_n}{|C_{[d]}|}$. Below we list the details of components of POMDP model. The state captures the true security state of every domain and the actions specify the thresholds for monitoring each channel, the nodes to be monitored and the decision about which domains are classified as malicious. As we assume the security state of the system does not change, the transition function is straightforward.

$$
\begin{aligned}
\text{States} \quad & \text{S} \quad = \langle X_1, \ldots X_D \rangle \\
\text{Actions} \quad & \text{A} \quad = A_c \times A_n \times A_d \text{ where } \langle \tau_1, \ldots \tau_K \rangle \in A_c, \langle a_1 \ldots a_N \rangle \in A_n \\
& \qquad \text{and } \langle M_1 \ldots M_D \rangle \in A_d \\
\text{Transision} \quad & T(s', s) \quad = \begin{cases} 1 & \text{iff } s' = s \\ 0 & \text{else} \end{cases}
\end{aligned}
$$

Next, we obtain an observation Ω_k for each channel, and as stated earlier for each channel the probability of an alert is given by functions α and β. We state the probability first for the observations for each domains, and then for all the observations using independence across domains.

Observations

$$\Omega \quad = \langle \Omega_1 \dots \Omega_K \rangle$$

Observation Prob

$$O(\Omega_{[d]}|X_d, A) \quad = \begin{cases} \prod_{k:k\in C_{[d]}\wedge z_k=1} \alpha(\tau_k)^{\Omega_k}(1-\alpha(\tau_k))^{1-\Omega_k} & \text{if } X_d = 1 \\ \prod_{k:k\in C_{[d]}\wedge z_k=1} \beta(\tau_k)^{\Omega_k}(1-\beta(\tau_k))^{1-\Omega_k} & \text{if } X_d = 0 \\ 0 & \text{else} \end{cases}$$

$$O(\Omega|X, A) \quad = \prod_d O(\Omega_{[d]}|X_d, A_{[d]})$$

Finally, the reward for the POMDP is given by the following equation:

$$\text{R(S,A)} \quad = -\left(\sum_d \left(X_d(1-M_d)v_{[d]} + (1-X_d)M_d w_{[d]}\right) + \sum_n^N a_n w_n\right)$$

The reward contains two cost components: the first component has two terms for each domain that specify the penalty for mislabeling a domain and the second component is the cost of sensing over the nodes. When a malicious domain d is labeled safe then the defender pays a cost $v_{[d]}$, i.e., the average value of channels going to domain d; in the opposite mislabeling the defender pays a cost $w_{[d]}$, i.e., a cost specified by loss of all traffic going to domain d. While this POMDP model captures all relevant elements of the problem, it is not at all tractable. Consider the input variables to this model, the number of domains D, the number of nodes N and the number of channels k. The state space grows as $O(2^D)$, the action space is $O(2^N|\tau|^K 2^D)$ and the observation space is $O(2^K)$.

This full formulation is exponential in all the input variables and cannot scale to larger, realistic network instances (we also show this experimentally in the Evaluation Section). In order to reduce the combinatorial nature of the observation space, action space and state space, we introduce a compact representation for the observation and action space and a factored representation for the state space that results in splitting the POMDP into multiple POMDPs.

4.4 POMDP Abstraction

We represent the POMDP compactly by using three transformations: (1) we use the same threshold for very channel going to the same domain and change the action space from sensing on nodes to sensing on channels, (2) reduce the observation space by noting that only the number of alerts for each domain are required and not which of the channels generated these alerts and (3) factoring the whole POMDP by domains, then solve a POMDP per domain and combine the solutions at the end. Next, we describe these transformations in details.

Abstract Actions

We can reduce the action space by (1) enforcing that the same threshold is set for all channels going to the same domain and (2) by reasoning about which channels to sense over instead of which nodes to sense on. The first change reduces the action space from a $|\tau|^K$ dependance to $|\tau|^D$, where $|\tau|$ is the discretization size of the threshold for the detector. The new set of threshold actions is then $A_c = \langle \tau_1, \dots \tau_D \rangle$. The second change replaces the set of actions on nodes A_n with a set of actions on channels $A_k = \langle s_{k_{[1]}} \dots s_{k_{[D]}} \rangle$,

where $s_{k_{[d]}}$ is the number of channels to be sensed out of the $|C_{[d]}|$ channels that end in domain d. This changes the action space complexity from 2^N to $|C_{[d]}|^D$. Then the action space is given by

$$\begin{aligned} &\text{Actions} \quad \text{A} = A_c \times A_k \times A_d \\ &\qquad \text{where } \langle \tau_1, \ldots \tau_D \rangle \in A_c, \langle s_{k_{[1]}} \ldots s_{k_{[D]}} \rangle \in A_k \text{ and } \langle M_1 \ldots M_D \rangle \in A_d \end{aligned}$$

In order to properly compute the reward we need to compute the cost of any action in A_k. To do this we need to build a lookup table mapping each action in A_k to an action in A_n, and hence obtain the cost of actions in A_k. Because we will always choose the lowest cost way to sense on a number of channels, the action of sensing a specified number of channels can be mapped to the set of nodes that minimizes the cost of sensing the specified number of channels. We can compute this using the following mixed integer linear program (MILP) mincost($\langle s_{k_{[i]}} \ldots s_{k_{[D]}} \rangle$).

$$\min_{z_k, a_n} \quad \sum_n a_n w_n \tag{4.1}$$

$$z_k \leq \sum_{n \in c_k} a_n \quad \forall k \in \{1, \ldots, K\} \tag{4.2}$$

$$\sum_{c_k \in C_{[d]}} z_k \geq s_{k_{[d]}} \quad \forall d \in \{1, \ldots, D\} \tag{4.3}$$

$$z_k \in \{0, 1\} \quad a_n \in \{0, 1\} \tag{4.4}$$

The mincost($\langle s_{k_{[1]}} \ldots s_{k_{[D]}} \rangle$) MILP needs to be solved for every action $\langle s_{k_{[1]}} \ldots s_{k_{[D]}} \rangle \in A_k$, i.e., we need to fill in a table with $O(|C_{[d]}|^D)$ entries. If we take the example network in Figure 4.2, the old action space is $A_n = \{\{\emptyset\}, \{n_1\}, \{n_2\}, \{n_1, n_2\}\}$, and the new action

space is $A_k = \{\{0,0\},\{1,0\},\{2,0\},\{3,0\},\{0,1\},\{1,1\},\{2,1\},\{3,1\}\}$. In order to map back to the representation using nodes, we build the mapping: $\{0,0\} \rightarrow \emptyset, \{1,0\} \rightarrow \{n_1\}, \{2,0\} \rightarrow \{n_1\}, \{3,0\} \rightarrow \{n_1\}, \{0,1\} \rightarrow \{n_2\}, \{1,1\} \rightarrow \{n_1,n_2\}, \{2,1\} \rightarrow \{n_1,n_2\}, \{3,1\} \rightarrow \{n_1,n_2\}$. However, the problem of converting from number of channels to nodes (stated as mincost($\langle s_{k_{[1]}} \dots s_{k_{[D]}} \rangle$) above) is not easy as the following theorem shows:

Theorem 3. *The problem of converting from number of channels to nodes is NP hard to approximate to any factor better than* $\ln |N|$.

Proof. We perform a strict approximation preserving reduction from the set cover problem. Consider a set cover problem. We are given a universe of m elements E and u subsets of E: U. Form a node n_u for each subset $u \in U$ and a domain d_e for each element $e \in E$. For any particular element e and any node containing that element, connect it to the domain d_e. Then, these connections, say from l nodes, defines l channels each starting from a node and ending in d_m. For any domain d choose $s_{k_{[d]}} = 1$, i.e., at least one channel needs to be sensed. It can be easily seen that for any channel c_k in this network there is a unique node it passes through: call it $n(k)$. Choose $w_n = 1$. Then, the optimization problem to be solved is the following:

$$\min_{z_k, a_n} \quad \sum_n a_n \tag{4.5}$$

$$z_k \leq a_{n(k)} \quad \forall k \in \{1, \dots, K\} \tag{4.6}$$

$$\sum_{c_k \in C_{[d]}} z_k \geq 1 \quad \forall d \in \{1, \dots, D\} \tag{4.7}$$

$$z_k \in \{0,1\} \quad a_n \in \{0,1\} \tag{4.8}$$

First, we prove that the constraints of this optimization specify a choice of subsets (nodes) the union of which equals the set E. Since all channels going to domain d corresponds to a unique element e and at least one channel going to d is chosen (Eq. 4.7), this implies at least one node containing e is selected (Eq. 4.6). Thus, the set of nodes (hence subsets) contains all elements e.

Given, the feasible space is given by a set of subsets (nodes) the union of which produces E, the objective clearly produces the minimum number of such sets. Also, any approximate solution with guarantee α maps to an α approximate solution of the set cover problem. The theorem follows from the lack of better than $\ln n$ approximatability of set cover. □

Abstract Observations

As we only reason about the state of each domain in the network, not each individual channel, we can aggregate the observation in order to reduce the observation space. Thus, instead of recording which channel generated an alert, we only record total number of alerts per domain. Given there are $|C_{[d]}|$ channels going to domain d then the observations for each domain lie in $\{0 \dots |C_{[d]}|\}$. This observation space for each domain is then linear in the number of channels $O(|C_{[d]}|)$. The full joint observation space is exponential in the number of domains $O(|C_{[d]}|^D)$.

The set of observations is then $\Omega = \langle \Omega_1, \dots, \Omega_D \rangle$ where $\Omega_d \in \{0 \dots |C_{[d]}|\}$ corresponding to the number of alerts from all $|C_{[d]}|$ channels going to domain d. Because there is now multiple way for us to get this single observation, the observation probability function for each domain also needs to be modified.

$$O(\Omega_d|X_d, A) = \begin{cases} \binom{s_{k_{[d]}}}{\Omega_d} \alpha(\tau_d)^{\Omega_d}(1-\alpha(\tau_d))^{s_{k_{[d]}}-\Omega_d} & \text{if } X_d = 1 \\ \binom{s_{k_{[d]}}}{\Omega_d} \beta(\tau_d)^{\Omega_d}(1-\beta(\tau_d))^{s_{k_{[d]}}-\Omega_d} & \text{if } X_d = 0 \\ 0 & \text{else} \end{cases}$$

VD-POMDP Factored Representation

Looking at both the observation probability function as well as the belief update, we can consider a factored representation of this POMDP, by factoring these by domains. If we then separate out these factored components and create a new sub-agent for each factor, so that we now have a total of D POMDP's, we can greatly reduce the state space, observation space and action space for each individual sub agent. The model for each of these individual POMDP is then given as follows.

States $\quad \text{S} = X_d$

Actions $\quad \text{A} = \tau_d \times \{0, \ldots, |C_{[d]}|\} \times M_d$

Transition
$$T(s', s) = \begin{cases} 1 & \text{iff } s' = s \\ 0 & \text{else} \end{cases}$$

Observations $\quad \Omega = \langle \Omega_1, \ldots, \Omega_D \rangle$ where $\Omega_d \in \{0, \ldots, |C_{[d]}|\}$

$$O(\Omega_d|X_d, A) = \begin{cases} \binom{s_{k_{[d]}}}{\Omega_d} \alpha(\tau_d)^{\Omega_d}(1-\alpha(\tau_d))^{s_{k_{[d]}}-\Omega_d} & \text{if } X_d = 1 \\ \binom{s_{k_{[d]}}}{\Omega_d} \beta(\tau_d)^{\Omega_d}(1-\beta(\tau_d))^{s_{k_{[d]}}-\Omega_d} & \text{if } X_d = 0 \\ 0 & \text{else} \end{cases}$$

Reward $\quad \text{R(S,A)} = -\Big(X_d(1-M_d)v_{[d]} + (1-X_d)M_d w_{[d]} + \text{mincost}(s_{k_{[d]}})\Big)$

	Full POMDP		VD-POMDP
	Original	Abstract	
State	$O(2^D)$	$O(2^D)$	$O(1)$
Action	$O(2^N\|\tau\|^K 2^D)$	$O(\|C_{[d]}\|^D\|\tau\|^D 2^D)$	$O(2^{\|C_{[d]}\|}\|\tau\|)$
Observation	$O(2^K)$	$O(\|C_{[d]}\|^D)$	$O(\|C_{[d]}\|)$

Table 4.2: Complexities of Full and VD-POMDP models with original and compact representations.

The complexity of the state space is reduced to $O(1)$, the action space is $O(|\tau||C_{[d]}|)$ and the observation space is $O(|C_{[d]}|)$. Table 4.2 shows the comparative complexities of the original POMDP model and the VD-POMDP model. As we use channels as actions for each domain specific POMDP, we still need to construct the lookup table to map channels as actions to nodes as actions in order to obtain the cost of each action on channels. Factoring the model in the way described above also simplifies the construction of this lookup table from actions on channels to actions on nodes, and hence computing mincost$(s_{k_{[d]}})$ can be done in a much simpler way for the VD-POMDPs. We solve a similar (MILP) as in (4.2)-(4.4) but for each VD-POMDP for domain d; thus, we only need to fill in a table with $O(|C|)$ entries, one for each of the $s_{k_{[d]}}$ actions for each domain d. The new MILP formulation is given in equations 4.10-4.12. Observe that unlike the MILP (4.2)-(4.4) used to build the lookup table for the original POMDP, this MILP is solved for a fixed domain d.

$$\min_{z_k, a_n} \quad \sum_n a_n w_n \tag{4.9}$$

$$z_k \leq \sum_{n \in C_k} a_n \tag{4.10}$$

$$\sum_{c_k \in C_{[d]}} z_k \geq s_{k_{[d]}} \tag{4.11}$$

$$z_k \in \{0, 1\} \quad a_n \in \{0, 1\} \tag{4.12}$$

While the above optimization is much more simpler than the corresponding optimization for the original POMDP, it is still a hard problem:

Theorem 4. *The problem of coverting the number of channels to nodes for each VD-POMDP is NP Hard.*

Proof. We reduce from the min knapsack problem. The min knapsack problem is one where the objective is to minimize the value of chosen items subject to a minimum weight W being achieved, which is a well known hard problem. Also, wlog, we can assume weights of items and W to be integers. Given a min knapsack with n items of weights w'_i and value v_i and min weight bound W form an instance of our problem with n nodes (mapped to items) and each node i having w'_i channels going directly to domain d. It can be easily seen that for any channel c_k in this network there is a unique node it passes through: call it $n(k)$. Each node i also has traffic $w_i = v_i$. Also, $s_{k_{[d]}} = W$. Then, the optimization problem being solved is

$$\min_{z_k, a_n} \quad \sum_n a_n v_n \text{ subject to } z_k \leq a_{n(k)}, \sum_{c_k \in C_{[d]}} z_k \geq W,\ z_k \in \{0, 1\},\ a_n \in \{0, 1\}$$

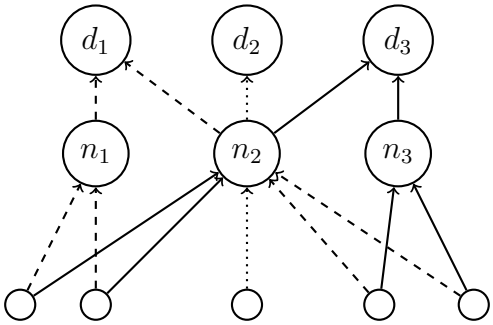


Figure 4.3: Sample network with 3 domains, 3 nodes and 5 sources. The dashed lines are channels to domain d_1 the dotted line is the channel to domain d_2 and the solid lines are channels to d_3.

Note that in the constraints, whenever a node is selected $a_{n(k)} = 1$ then making all w_i channels in it one makes the weight constraints less tight. Thus, any values of a_n, z_k satisfying the constraints specify a set of nodes such that the sum of its weights is $\geq W$. Coupled with the fact that the objective minimizes the values of selected nodes, the solution to this optimization is a solution for the min knapsack problem. □

Policy execution: The solutions to each of these VD-POMDP's give us an action $\langle M_d^*, \tau_d^*, s_{k_{[d]}}^* \rangle$ corresponding to a labeling of malicious or legitimate for that particular domain d, the threshold, and the desired number of channels to sense over. However, at execution time we need to turn on detectors on nodes. Thus, in order to aggregate these factored actions to determine the joint action to take at execution, we need to map the output from each POMDP back to a set of sensing actions on nodes. This can be easily accomplished by solving a single instance of the larger MILP (4.2)-(4.4) with the $s_{k_{[d]}}$ values set to $s_{k_{[d]}}^*$.

We *emphasize* here the importance of using the abstract channels as actions instead of nodes. The possibly alternate approach with nodes as action for each sub-POMDP and

just taking union of the nodes output by each domain specific POMDP, when the channels are not disjoint, can result in over sensing. Consider the example below, where there are 4 channels going to domains d_1 and d_3 and one to d_2 and let us currently be in a belief state where the optimal action for domain d_1 and d_3 would be to sense on 2 channels out of the 4 going to each domain and the optimal action for d_2 is to sense on the one channel. Working in an action space of nodes, the VD-POMDP for d_1 would choose to sense on node n_1, the VD-POMDP for d_3 would choose to sense on n_3 as it has the lowest amount of traffic for 2 channels and the one for d_2 would choose to sense on n_2 as it is the only option. Taking the union of these would result in all the sensors being turned on. However, we can see that choosing only to sense on node n_2 satisfies the sensing requirements of all three separate VD-POMDPs.

Next, we identify a condition under which the solution from the larger MILP is optimal. In the next section, we show empirically that even when this condition is not met our approach is close to optimal.

Theorem 5. *The above described technique of aggregating solutions of the VD-POMDPs is optimal for the original POMDP iff the solution to the MILP (4.2)-(4.4) for any VD-POMPD policy action results in an equality for the constraint (4.3).*

Proof. First, with the standard representation the global value function given in equation 4.13-4.14, cannot generally be fully decomposed by domain due to the $R_n(a_n)$ term which couples the actions for each domain through the sending cost. The decomposition is only possible in special instances of the problem, such as if network of channels were completely disconnected. The action of selecting nodes can be partitioned by domains as $a_{n_{[d]}}$. Then, the cost associated with sensing on the nodes could be written as a sum of domain dependent terms $R_n(a_n) = \sum_d R_{n_{[d]}}(a_{n_{[d]}})$. Also, all actions (threshold, choice of nodes

and decision about each domain) are now partitioned by domain, thus any action a is a combination of actions per domain a_d. Let b_d denote the belief state for domain d. The choice of nodes in this case should just be a union of the nodes chosen by each POMDP as seen from the the value function as each domain dependent component can be optimized separately.

$$V^* = \max_a \left[R(b,a) + \gamma \sum_{\Omega} P(\Omega|b,a) V^*(b,a,\Omega) \right] \tag{4.13}$$

$$= \max_a \left[\sum_d \Big(R_d(b_d, M_d) \Big) + R_n(a_n) + \gamma \sum_{\Omega} \prod_d P(\Omega_d|b_d,\tau_d) V^*(b,a,\Omega) \right] \tag{4.14}$$

$$= \max_a \left[\sum_d \Big(R_d(b_d, M_d) + R_{n_{[d]}}(a_{n_{[d]}}) \Big) + \gamma \sum_{\Omega} \prod_d P(\Omega_d|b_d,\tau_d) V^*(b,a,\Omega) \right] \tag{4.15}$$

$$= \max_a \left[\sum_d \Big(R_d(b_d, a_d) \Big) + \gamma \sum_{d,\Omega_d} P(\Omega_d|b_d,a_d) V_d^*(b_d,a_d,\Omega_d) \right] = \sum_d V_d^* \tag{4.16}$$

$$\text{where } V_d^* = \max_{a_d} \left[R_d(b_d,a_d) + \gamma \sum_{\Omega_d} P(\Omega_d|b_d,a_d) V_d^*(b_d,a_d,\Omega_d) \right] \tag{4.17}$$

If we instead use the compact representation of the action space, and let the actions simply be the number of channels to be sensed on and equality is obtained for the constraint (4.3) for any action from each domain specific POMDP (i.e., each $s_{k_{[d]}}$ can be implemented), then the value function can be decomposed by domain, because the term $R_n(a_n)$ is replaced by the $\sum_d \text{mincost}(s_{k_{[d]}})$, which can be factored by domain and does not couple the VD-POMDP's. Reconstructing the joint policy then just amounts to finding the best

action from each POMDP and taking a union of these individual actions. We then just need to map back to the representation of actions on nodes, by solving the MILP (4.2)-(4.4).

$$V^* = \max_{a_d} \left[\sum_d \left(R_d(b_d, a_d) + \text{mincost}(s_{k_{[d]}}) \right) + \gamma \sum_{d,\Omega_d} P(\Omega_d | b_d, a_d) V_d^*(b_d, a_d, \Omega_d) \right] = \sum_d V_d^*$$

□

4.5 VD-POMDP Framework

Here we explain at a high level, the VD-POMDP framework as applied to the data exfiltration problem, and how it is implemented. With the VD-POMDP, entire planning model is broken up into two parts as depicted in Figure 4.4. The first is the offline factoring and planning, where the POMDP is factored into several sub-agents, and each solved individually. Second is the online policy aggregation and execution, where the policies of each sub-agent are aggregated as each of them choose actions to perform.

In order to build the VD-POMDP for data exfiltration problem, we first construct the network graph, based on the topology of the actual computer network we are modeling as well as the set of domains under consideration, shown at point (a) in Figure 4.4. Then at (b), for each domain in our network, we construct a separate POMDP sub-agent. In order to do this we solve the MILP mincost(s_{k_d}) for each agent, in order to abstract away from the network layer and construct the compact representation of the network. At point (c) each individual POMDP agent is solved, offline, ignoring the presence of the other agents to obtaining a policy for each domain.

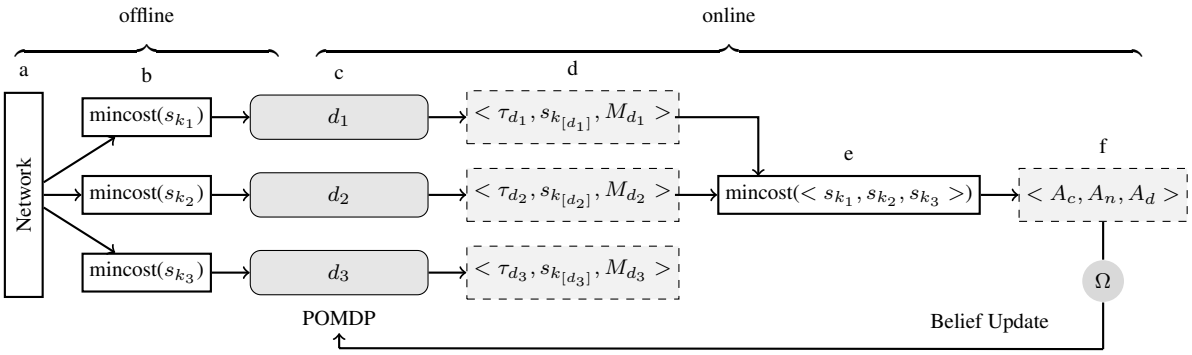


Figure 4.4: Flowchart for the Data Exfiltration VD-POMDP

The policies are then aggregated in an online fashion, shown at point (d) in Figure 4.4 to obtain a joint action (f). At each time step the agents receive observations from the network and update their beliefs individually. Each agent then presents the individual action to be performed consisting of a number of channels to be sensed on, a threshold for sensing and a classification of malicious or legitimate for their respective domain. The required number of channels for each agent is then fed into the MILP mincost($\langle s_{k_{[i]}} \dots s_{k_{[D]}} \rangle$) to determine the set of nodes to be sensed on. The agents then again receive observations from the resulting set of detectors and iterate through this process again.

Policy aggregation is performed online as it would be infeasible to do offline policy aggregation for all but the smallest policies. If aggregation were to be performed offline, we would need to consider every possible combination of actions from each policy and then solve the MILP mincost($\langle s_{k_{[i]}} \dots s_{k_{[D]}} \rangle$) for each of these, in order to compute the combinatorially large joint policy. Because the MILP is fast to solve, it does not result in much overhead when these joint actions are computed in an online fashion.

It is important to note here that the policy we compute is not an optimal sequence of actions, but rather a mapping of belief state to actions. This distinction is important, as it

may be the case that, upon policy aggregation, there is no feasible implementation of the individual action. In such a scenario, an agent may choose an action to sense on a subset of k channels; however, given the sensing requirements of the other agents, the agent in question may actually get to sense on more channels than they had initially wanted. The agent may then end up in a belief state that they had not originally planned for, but because we are solving for the entire belief space, we still know how to behave optimally. Additionally, from Theorem 5, we know that the joint action will only be optimal if we can exactly implement each individual action, and no agent get to sense on more channels than it requests. Our policy aggregation may then result in a suboptimal joint action being taken, however, we show later in section 6, that even when the optimality condition does not hold, we can still achieve good performance.

4.6 Evaluation

We evaluate our model using three different metrics: runtime, performance, and robustness. We first look at the runtime scalability of VD-POMDP model, varying the size of several synthetic network as well as the number of domains and compare to the standard POMDP model. We then evaluate the performance of the VD-POMDP, measuring how quickly it can classify a set of domains as malicious or legitimate, as well as computing the accuracy of correct classifications. For small network sizes, we compare the performance of the VD-POMDP to the original model and look at the performance of the VD-POMDP on larger synthetic networks.

Synthetic Networks

In order to test a variety of network sizes we created synthetic networks using a tree topology. Leaf nodes in the tree network correspond to source computers. Channels travel upwards from these nodes to the root of the tree; for each domain we create one such channel on each source computer. The size of the network is varied by varying the depth and branching factor of the tree.

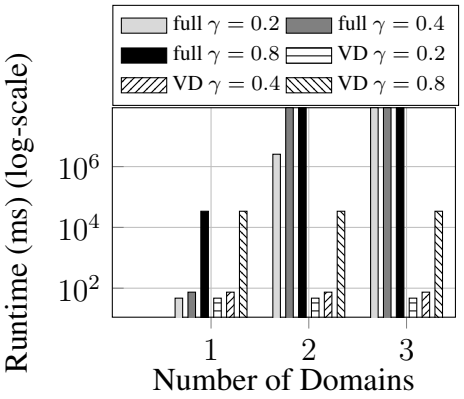
4.6.1 DETER Testbed Simulation

We also evaluated the performance of our model using a real network, by running simulations on the DETER testbed. The DETER testbed provides capabilities of simulating a real computer network with virtual machines and simulating agents that perform tasks on each computer. Every agent is specified in a custom scripting language, and allows simulating attackers, defender and benign users. For our simulation we simulated legitimate DNS queries as well as launched real attacks. We performed a simple attack, by attempting to exfiltrate data from a file to a chosen malicious domain by embedding data from the file into the DNS queries. We conducted the attack using the free software Iodine [Kryo, 2014] which allows for the creation of IP tunnels over the DNS protocol in order to generate these DNS queries. We were provided with 10 virtual machines, from which we formed a tree topology with 7 of them as host computers and sources of traffic. We then built and implemented a real time data exfiltration detector based off of the techniques proposed in [Paxson et al., 2013]. The detector uses off the shelf compression algorithms like *gzip* in order to measure the information content of any channel in the network. We then set a cut off threshold for the level of allowable information content in any channel.

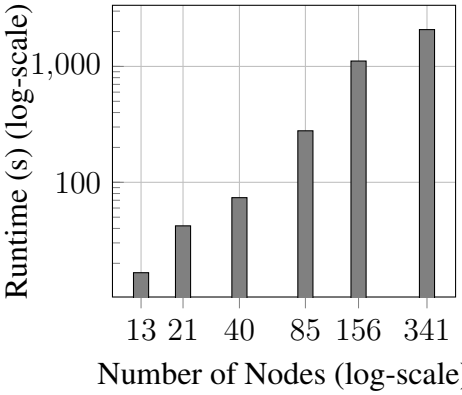
Channels exceeding this threshold are flagged as malicious. While we chose to use this specific detector to generate observations for our model, it is important to note that any other methods for detecting exfiltration would have been equally viable.

4.6.2 Runtime

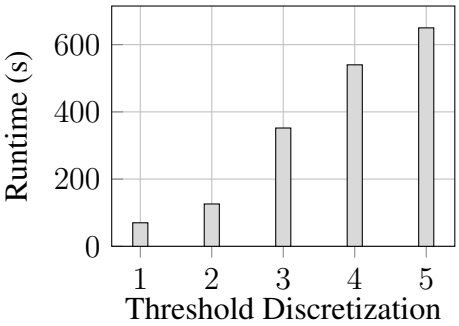
We first look at runtimes needed to solve the original model compared to our VD-POMDP model with increasing number of domains. Unless otherwise stated, all test used a threshold discretization of 2. We used an offline POMDP solver ZMPD [Smith, 2007] to compute policies; however, any solver which computes policies for the entire belief space may be used. The largest network we were able to solve for with the original model was one of only 3 nodes. For larger than 2 domains with discount factors $\gamma = -0.2$ and all cases with $\gamma = -0.4$ and $\gamma = -0.8$ the original POMDP did not finish solving in 24 hours and is shown cut off at the 24hr mark in Figure 4.5a. Consistent with the complexities in Table 4.2, in Figure 4.5a we see the runtimes on the y-axis increase exponentially with the number of domains on the x-axis, for the original POMDP. If the VD-POMDP models are solved in parallel, runtimes do not vary with increasing domain. If the models are solved sequentially, then we would see only a linear increase in runtime. However in the case where networks have the channels uniformly distributed among all hosts, i.e. there exists one channel from every host to every domain, then the models become identical, and it becomes only necessary to solve one of them.



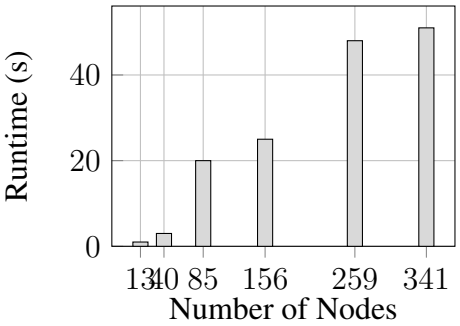
(a) Comparing runtimes of the POMDP to the VD-POMDP for a small network of 3 nodes and varying discount factor γ. On the y-axis, we have runtime in log scale and on the x-axis, we have increasing number of domains.



(b) Log-Log graph of runtimes for a single VD-POMDP for networks of increasing size. Both axes are in log scale, with runtimes on the y-axis in seconds and number of nodes on the x-axis, and showing a linear increase with network size.



(c) Runtime for a single VD-POMDP with increasing action space. Test were run on a tree network with branching factor 4 and depth of 3 for a total of 85 nodes.



(d) Runtime required to build network for a single VD-POMDP. Time in seconds is shown on the y-axis and size of network in number of nodes is shown on the x-axis.

Figure 4.5: Runtime results

We show the scalability of computing policies for the VD-POMDP in Figure 4.5b. On the y-axis, in log scale we have the runtime in seconds, and on the x-axis, also in log scale we have the number of nodes in the network, achieved by varying both the depth and branching factor of our tree network structure. We can see that there appears to be a linear scaling with the size of the network. We also show in Figure 4.5d the time it takes to build the network, the lookup table of costs computed by repeatedly solving (4.10)-(4.12) and pomdp files to be fed to the solver. This time corresponds to the steps (a) and (b) in Figure 4.4. On the y-axis we have again the runtime and on the x-axis the number of nodes in the network.

Figure 4.5c shows the runtime for computing the policy of a single factored agent with increasing action space. The runtime is shown on the y-axis in seconds, while the increasing action space is measured by the threshold discretization on the x-axis. We first divided the space of possible true positive and true negative rates into a number of segments equal to the discretization number. For each discretization level, we then combinatorially formed all the true positive and true negative pairs possible within that discretization number and averaged over the runtimes, in order to ensure that we were not only testing easy cases, where one choice threshold was dominant over another.

4.6.3 Performance

We evaluate the performance of the model by looking at the reward, the number of time steps taken to classify all domains and the accuracy of the classifications. For each test, we averaged the values over 100 simulations. Table 4.3 compares the performance of the original POMDP model with the VD-POMDP model. The largest tests we could run using the full POMDP were on a network of 3 nodes with a total of two domains, while solving

the model with a discount factor of $\gamma = 0.2$. The VD-POMDP model performs as well in terms of accuracy and time compared to the full POMDP model. We show a normalized average reward, computed by dividing the total reward by the number of time steps taken to classify the domains to better compare the models. Since we stop the simulation after all the domains have been classified, the total reward is not the expected infinite horizon reward, so simulations which run for different amounts of time will have had the chance to accumulate different amounts of reward. The normalized reward is meant to give a better indication of what the long term average reward would be, which would be a much fairer comparison. We also looked at the VD-POMDP solved with a discount factor of $\gamma = 0.8$, where we can clearly see the benefit of longer term planning. Although this VD-POMDP takes longer to classify both domains, it has a perfect accuracy and lower normalized reward than the other two models. This shows that the model is placing more value on potential future information, by preferring to wait and collect more alerts before making a final decision. This is clearly the better choice as we see a much better accuracy. It is clear that it is necessary to be able to plan for the future to perform well in this kind of domain; it is therefore necessary to be able to solve for large planning horizons, something that we cannot do using just the original POMDP model. This demonstrates the merit of the VD-POMDP framework, as solving this problem with a simple POMDP framework is clearly infeasible.

Looking at just the VD-POMDP we test performance on a variety of larger networks in Table 4.4. Each of the synthetic networks are tested with 50 domains, averaged over 30 trials. The DETER network is tested with 100 domains averaged over 30 trials. For the DETER network, we used two thresholds, and determined the true and false positive rates of our detector by letting it monitor traffic at each threshold setting and observing the

Model	Timesteps to Classify	Attack Traffic Accuracy	User Traffic Accuracy	Normalized Reward
Full POMDP $\gamma = 0.2$	11.814	0.948	0.979	-470.594
VD-POMDP $\gamma = 0.2$	11.144	0.944	0.961	-675.100
VD-POMDP $\gamma = 0.8$	29.044	1.0	1.0	-386.982

Table 4.3: Comparing performance of full POMDP model to factored model on a small test network of 3 nodes, with 2 domains. One domain is malicious and the other domain is legitimate.

number of alerts obtained for each channel. We found our simple implementation of the detector had true positive rates of $\alpha(\tau 1) \simeq 0.35$, $\alpha(\tau 2) \simeq 0.45$ and true negative rates of $\beta(\tau 1) \simeq 0.8$, $\alpha(\tau 2) \simeq 0.7$, and these were the parameters used in the model for this experiment as well as all the synthetic ones. We can see that, although the synthetic simulations all perform extremely well, and have a perfect accuracy, the deter simulation occasionally misclassifies legitimate traffic. This is due to the uncertainty in the characterization of the detector, as network traffic is variable and may not always follow a static distribution. Observations for the synthetic experiments were drawn from the distributions that the VD-POMDP had planned for, while in the DETER experiments, traffic did not always follow the input distributions. However, even with this uncertainty, the model still performs well in this realistic network setting. A more sophisticated implementation of this detector along with a more intensive characterization would even further boost the performance.

We also show an example of the diversity of actions chosen by the VD-POMDP. In Table 4.6 we show a trace of the actions taken by a single agent planning for a single domain. We show the number of channels chosen to sense on on, the choice of threshold,

Network	Timesteps to Classify	Attack Traffic Accuracy	User Traffic Accuracy	Normalized Reward
Synthetic 40 Nodes	4.079	1.0	1.0	-13523275.239
Synthetic 85 Nodes	3.252	1.0	1.0	-15514333.580
Synthetic 156 Nodes	3.235	1.0	1.0	-22204095.194
Synthetic 341 Nodes	3.162	1.0	1.0	-21252069.929
DETER	5.3076	1.0	0.995	-6835.588

Table 4.4: Performance of the factored model on larger networks.

Time	Action			Observations
	# Channels	τ	M_d	
1	64	1	0	23
2	62	1	0	23
3	3	0	0	0
4	5	0	0	2
5	8	0	0	2
6	18	0	0	4
7	0	0	0	0

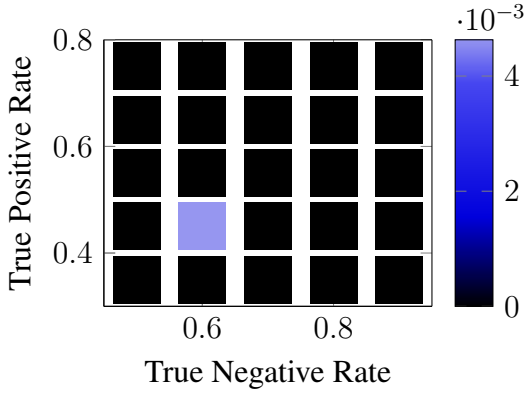
Figure 4.6: Trace of a run on a network of 85 nodes of a single legitimate domain.

along with the classification of the domain. We also show the observations, which the number of channels that triggered alerts. The simulation ends when no more channels are to be sensed on. We can see the agent varying the number of channels as well as the threshold of the detector, as they become more and more sure that they domain is legitimate.

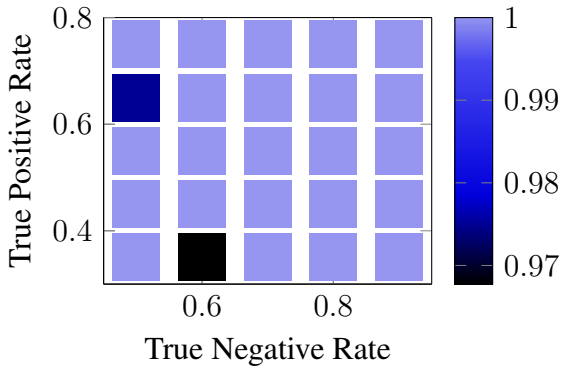
4.6.4 Robustness

Lastly, we looked at the robustness of our model to errors in the input parameter. As evidenced with the DETER experiment, the model requires known false positive and true positive rates for the detector. While it may be reasonable to assume that with enough

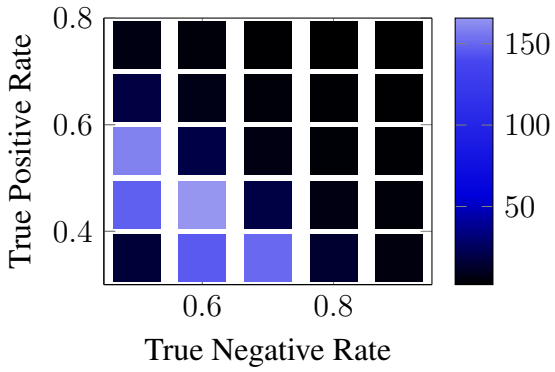
monitoring, it is possible to get accurate measures of false positive rates in a network by simply running the detector on known legitimate traffic for long periods of time, it is more difficult to characterize the true positive rates, as attacks can take many forms and exfiltration can occur over varying rates. In order to test the robustness of our model, we solved for the policy using one set of rates and then tested the model in simulation against a variety of actual rates. For our tests, the model was solved with a true negative rate of 0.8 and true positive rate of 0.55. We then drew alerts from a range of distributions for the true positive and negative rates as shown in Figures 4.7 on the y-axis and x-axis respectively. The metrics used to measure robustness are shown as a heat-map for each true positive, true negative pair.



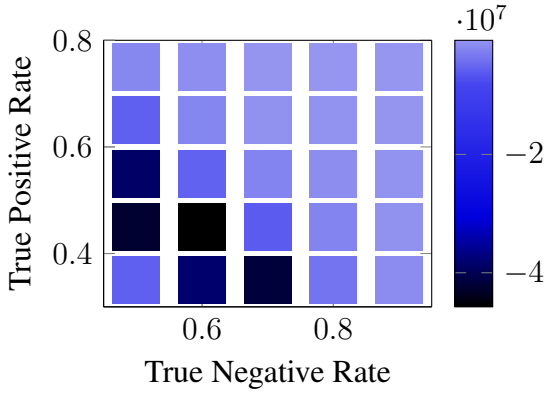
(a) Percent of Incorrect Legitimate Domain Classifications



(b) Percent of Correct Malicious Domain Classifications



(c) Timesteps to Classify Domain



(d) Average Reward

Figure 4.7: Testing the robustness with respect to error in the planned true positive and true negative rate.

In Figure 4.7a performance of the model was tested by looking at the percent of incorrect legitimate domain classifications ie. the percent of legitimate domains flagged as malicious. In all cases except for one, all legitimate domains were correctly flagged as non-malicious, and in one case legitimate domains were misclassified in only 0.4% of the trials. In Figure 4.7b the percent of correct malicious domain classifications is shown, where in all but two cases, the correct domain was always identified. Figure 4.7c shows

the number of time steps taken to classify all the domains, while Figure 4.7d shows the average reward (in this case a penalty) for the the simulations. We can see that the model is robust to mischaracterization of the detectors, where the only dips in performance occur when either the detector has a low true negative rate and when the error in both the true positive and negative rates are large.

4.7 Conclusion and Future Work

We demonstrated the effectiveness of POMDP based planning tool in making intelligent decisions to tackle the problem of DNS based data exfiltration. These decisions were made by aggregating information from multiple noisy detectors and using sequential planning under uncertainty based reasoning. In doing so, we also proposed a new class of POMDPs called VD-POMDP that uses domain characteristics to split the POMDP into sub-POMDPs and allows for abstract actions in each sub-POMDP that can then be easily converted to a full joint action at execution time. VD-POMDP allows scaling up our approach to real world sized networks. The approach also detects attacks in near real time, thereby providing options to minimize the damage from such attacks. More generally, we believe that our approach applies to other security detection and response problems such as exfiltration over other protocols like HTTP and intrusion detection. While this work is an important step in addressing the problem of APT's in a realistic and scalable manner, we recognize that having a non-adaptive adversary is a simplification of the potentially complex interaction between attacker and defender in this environment. Building an appropriate adversary model, and considering the underlying game in this domain is a key avenue for future work in this area.

Part III

Tactical Planning

Chapter 5

Tactical Planning in Security Games

In this chapter I discuss the types of tactical planning problems which give rise for the need of hierarchical planning and how these types of tactical problems have been largely ignored in the security game literature. This chapter will focus in particular on resource heterogeneity and team formation in security games. Here I will first touch on why resource heterogeneity is important and then focus on a particular class of games known as Threat Screening Games, one of the only game theoretic security models which models resource heterogeneity and teams.

5.1 Tactical Planning with Heterogeneous Resources

While strategic planning can allow the defender to design more expressive security systems and achieve higher solution quality and therefore better security there is additional computation cost associated with performing this additional planning. As such it is important to recognize when this additional work is necessary or beneficial. We see from the results in section 3.4.2 that the amount of solution quality gained by performing strategic planning varies with the heterogeneity of the resources; as the number of possible unique teams which can be formed grows the value in strategic planning increases. When we looks at how the strategic design variables y affect the problem formulation, it is through constraints on the resource deployments $x_r^i \in G_r(y)$. When there is only one resource

type, the design variables only appear in a single constraint... and while the design variables themselves may be subject to some set of constraints, these are now completely decoupled from the problem, and we now only have to choose y to optimize the single value x

Example 3. *Consider the SORT problem from example 1. This becomes a trivial problem when there is only one resource type, as the solution is simply to purchase as many of that single resource type as possible, subject to the budget B. We see this from the problem formulation*

$$\max_{y} \{F(y) : yb \leq B\}$$

Where, since $F(y)$ corresponds to the optimal objective value of a game we know that $F(y) \geq F(y')$ if $y > y'$ and thus the optimal solution is $y = \frac{B}{b}$.

This shows us that strategic planning is particularly relevant to games with *teams of heterogenous resources.* However, to date the security game literature has focused mainly on problems with homogenous sets of resources. Thus, in order to take full advantage of the benefits of strategic planning it is important to be able to solve game models which capture this added complexity. In section 3.3.2 this thesis extends the NSG model to include heterogenous resources and provides a novel double oracle solution approach for solving this problem. Here we provide a more thorough discussion of resource heterogeneity in security games and how it changes the complexity of the tactical planning problem. I then look at one of the only types of games to include heterogeneous resource and provide a framework for solving these games in a more realistic setting.

5.1.1 Heterogenous Resource Efficiency

A common source of resource heterogeneity is effectiveness in interdicting, detecting or preventing an attack. This appears in the security game literature in areas such as cyber security, Other sources of heterogeneity exists, usually due to the constraint structure of the problem. For example, in the problem of network interdiction of Section 3.2.1 we have additional heterogeneity due to the differences in the path lengths the resources can cover; this additional distinction between resources is only possible due to the underlying network structure and how the network constrains affect each resource. However, the effectiveness of resource is ubiquitous to all security games, and so has the potential to appear regardless of the game structure. Thus it is a source of heterogeneity that is important to address as it is common to all security game problems.

The general problem of heterogeneity in efficiency can be framed in the following way. Consider a game G with a set of targets $\mathcal{V}$ and set of heterogeneous resource types $\mathcal{R}$ available. Let it be the case that each resource type $r \in \mathcal{R}$ has a probability $P_{r,v}$ of successfully averting an attack on target v so that, as in section 3.2.1 the probability of successfully averting an attack on target v is given by $P_v(x) = 1 - \prod_{r \in \mathcal{R}(t)}(1 - P_{r,t})$ similarly to the edge coverage equation 3.2, where $\mathcal{R}(t)$ corresponds to the set of all resources in team t. (Note that this can be equivalently expresses using a decision variable $x_{r,v}$ which corresponds to the number of resources of type r assigned to target t as $P_v(x) = 1 - \prod_{r \in \mathcal{R}}(1 - P_{r,t})^{x_{r,\mathcal{V}}}$). Let $R_{d,v}$ be the value of target v to the defender and let a_v be the adversary's mixed strategy over the targets (note that under the Stackelberg model, the optimal adversary mixed strategy is equivalent to a deterministic strategy, being a choice of a single target). We look at the defender best response to gain intuition on the hardness of the problem. The defender best response problems is a combinatorial optimization problem over all the defender pure

strategies, given fixed adversary strategy. It has been shown in [Xu, 2016] that there is an equivalence in the problems of computing the defender best response and computing the minimax equilibrium for zero-sum security games, the strong Stackelberg equilibrium for securitygames and computing the best or worst (for the defender) Nash equilibrium for security games over the pure strategy space of the defender. Since we are looking to compute SSE for security games, the the hardness of the defender best response implies the hardness the full problem.

Theorem 6. *For a security game with heterogeneous resources, the defender best response problem is a problem of maximizing a submodular function subject to matroid constraints.*

Proof. We first show that the probability of successfully defending a target is submodular. A function $f : [0,1]^{|D|} \rightarrow \mathbb{R}$ is submodular if for every $X, Y \subseteq DX, Y \subseteq D$ with $X \subseteq Y$ and every $x \in D \setminus Y$ we have that $f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y))$.

Let $f := P_v(t)$, where $t \in \mathcal{T}$ denotes a team or set of resources. Let $D = \mathcal{T}$. We have that:

$$
\begin{aligned}
P_v(X) &= 1 - \prod_{r \in X}(1 - P_{r,t}) \\
P_v(X \cup \{r'\}) &= 1 - \prod_{r \in X}(1 - P_{r,t})(1 - P_{r',t}) \\
P_v(Y) &= 1 - \prod_{r \in Y}(1 - P_{r,t})^{x_{r,v}} = 1 - \prod_{r \in X}(1 - P_{r,t}) \prod_{r \in Y \setminus X}(1 - P_{r,t}) \\
P_v(Y \cup \{r'\}) &= 1 - \prod_{r \in X}(1 - P_{r,t}) \prod_{r \in Y \setminus X}(1 - P_{r,t})(1 - P_{r',t})
\end{aligned}
$$

$$
\begin{aligned}
P_v(X \cup \{r'\}) - P_v(X) &= \left(1 - \prod_{r \in X}(1 - P_{r',t})\right) - \left(1 - \prod_{r \in X}(1 - P_{r,t})\right) \\
&= \prod_{r \in X}(1 - P_{r,t}) P_{r',v} \\
P_v(Y \cup \{r'\}) - P_v(Y) &= \prod_{r \in X}(1 - P_{r,t}) \prod_{r \in Y \setminus X}(1 - P_{r,t}) P_{r',t}
\end{aligned}
$$

$$
P_v(Y \cup \{r'\}) - P_v(Y) - P_v(X \cup \{r'\}) - P_v(X)
$$

$$
\begin{aligned}
&= \prod_{r \in X}(1 - P_{r,v}) \prod_{r \in Y \setminus X}(1 - P_{r,v}) P_{r',v} - \prod_{r \in X}(1 - P_{r,t}) P_{r',v} \\
&= \prod_{r \in X}(1 - P_{r,v}) P_{r',v} \left(\prod_{r \in Y \setminus X}(1 - P_{r,v}) - 1 \right)
\end{aligned}
$$

Since $\prod_{r \in X}(1 - P_{r,v}) \leq 1$ and $\prod_{r \in Y \setminus X}(1 - P_{r,v}) \leq 1$ we have that $\left(\prod_{r \in Y \setminus X}(1 - P_{r,v}) - 1\right) \leq 0$ therefore $P_v(Y \cup \{r'\}) - P_v(Y) - P_v(X \cup \{r'\}) - P_v(X) \leq 0$ which means that $P_v(X)$ is submodular.

Next we show that the constraints on the problem form a partition matroid. We have a number of disjoint sets $\mathcal{R}_i$ corresponding to all of the resources of type i and a set of integers y_i corresponding to the maximum number of resources of type i which may be deployed. Define the set of independent sets X to be the sets such that $|X \cap \mathcal{R}_r| \leq y_i \ \ \forall i$. These set for the independent sets of a partition matroid.

The defender's best response for a security game with heterogenous resources can be formulated as the following optimization problem:

$$\begin{aligned} \text{maximize}_x \quad & \sum_v a_v R_{d,v} P_v(X) \\ \text{subject to} \quad & P_v(X) = 1 - \prod_{r \in \mathcal{R}(X)} (1 - P_{r,v}) \\ & |X \cap \mathcal{R}_i| \leq y_i \forall i \end{aligned}$$

Thus the defender best response problem is a problem of maximizing a submodular function subject to matroid constraints. □

Solving games with resource heterogeneity proves to be a challenge when the heterogeneity is due to resource efficiency. In general, these problems are NP hard, and NP hard to approximate to within any better than a factor of $(1 - \frac{1}{e})$ [Feige, 1998, Calinescu et al., 2007].

5.1.2 Heterogenous Schedule Types

Another source of resource heterogeneity is in the types of schedules each resource may be assigned. For example, in the problem of network interdiction of Section 3.2.1 we have additional heterogeneity due to the differences in the path lengths the resources can cover. Different schedules here correspond to different patrols of various lengths. The different schedules types here come from the underlying structure of the game. Schedules are distinct if they cover different targets, as shown in the example in Figure 5.1.

Heterogeneity in resource schedules alss makes the problem difficult. In fact when resources are heterogeneous in their schedule types and schedules are not singletons (meaning that they may cover multiple targets) finding an optimal Stackelberg strategy is NP-hard, even when the game is zero-sum [Korzhyk et al., 2010b].

Standard techniques for solving large scale games such as incremental strategy generation techniques fail to properly address these challenge. These techniques, such as column generation and double oracle methods, require the ability to solve efficiently solve the best response problem, as they involve many oracle calls where the oracles must compute the best response strategies. As we have shown, for games with heterogeneous resource efficiency, this is a hard problem, so that any incremental strategy generation technique will be inefficient as it require solving this hard problem many times. Indeed, as an example of this we turn to one of the few games which model heterogenous resources: a class of games known as Threat Screening Games which will be the focus of the remainder of this chapter. Using these games as a motivating example, this thesis provides a framework for solving the tactical planning problem in games with heterogeneous resource types

5.1.3 Existing Games with Resource Types

Threat Screening Games (TSG) introduced in [Brown et al., 2016] are a game theoretic model to address the challenge of screening a flow of incoming items for threats. These

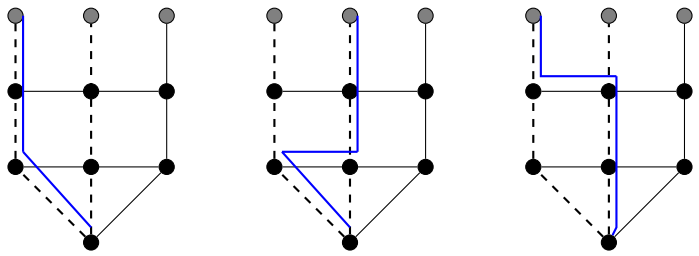


Figure 5.1: Three schedules on a graph. The targets are shown in gray, with the adversary paths to the targets shown by the dotted lines. The schedules are shown in blue overlayed on the graph. The first schedule is a singleton as it covers only target 1 by covering the first path. The second schedule covers target 1 and 2 by covering both paths. The third schedule covers both targets as well, however as it covers the same paths as the second schedule, they are considered equivalent.

games model situations, wherein a strategic attacker attempts to penetrate a secure area, while the screener has the opportunity to screen for threats using limited resources. Optimizing the defender (mixed) strategy by means of the TSG captures the strategic behavior of attackers and thus yields more effective screening strategies.

As mentioned previously, this model which inspired by previous work in security games [Tambe, 2011, Korzhyk et al., 2010a, Yin et al., 2015, Balcan et al., 2015, Basilico et al., 2009], is fundamentally difference in that it is one of the few games which reasons about *heterogenous resource types* each with varying efficacies and efficiencies with regards to deployment and capacity for use. While this enables TSGs to model a larger variety of resource allocation problems the resulting problem requires fundamentally new algorithms in order to scale up to real problem sizes. In [Brown et al., 2016] it is demonstrated that standard optimization techniques such as column generation are suboptimal in terms of solution quality and scalability. However, despite promising results, previous work in TSG fails in its mission to realistically model real-world settings. Its fundamental limitation is its assumption of perfect fore-knowledge of screenee arrival times (e.g., arrival times of passengers at airports). However, in the real-world there is significant uncertainty in arrival times. Addressing this challenge is difficult, as it requires reasoning about all the possible realizations of the uncertainty and coming up with an optimal plan for each of those scenarios. When dealing with a large number of screenees, this result in millions of possible scenarios, making the planning problem extremely difficult.

To address this shortcoming, this work provides a new model *Robust Threat Screening Games (RTSG)*, which expresses the required uncertainty in screenee arrival times. In RTSG, we model the problem faced by a screener as a robust multistage optimization problem. We present a tractable solution approach with three key novelties that contribute to its

efficiency: *(i)* compact linear decision rules; *(ii)* robust reformulation; and *(iii)* constraint randomization. We present extensive empirical results that show that our approach outperforms the original TSG methods that ignore uncertainty, and the exact solution methods that account for uncertainty.

5.2 Motivating Domain and Game Model

Screening for threats is an important security challenge, be it inspecting cargo at ports, passengers at airports, or fans entering a stadium. Given a strategic adversary capable of exploiting gaps in security measures, along with a large number of screenees, it becomes critical to optimize the allocation of limited screening resources. Indeed, to improve airport screening efficiency and effectiveness, the US Transportation Security Administration (TSA) recently launched the Dynamic Aviation Risk Management Solution (DARMS) initiative [AAAE, 2014] to incorporate adaptive screening. The goal of this project is to move towards more dynamic and adaptive risk based screening for passengers, through projects such as TSA pre-check in which passengers can choose to submit to background checks in order to receive expedited screening. Thus fewer resources can be dedicated to screening these lower risk passengers. Within the larger DARMS project there has been work on developing a multi-attribute utility model in order to assigns each passenger a risk level based on available information such as flight history, frequent flyer membership, TSA Pre-check status, as well as assigning values to each flight that measures its attractiveness as a target. This was done by decomposing the assessment of the probability of a successful attack using probability trees [Burns et al., 2015]. Our goal is exploit this flexibility and

information on passengers and flights to perform more effective and efficient passenger screening.

The original TSG model was developed in collaboration with the TSA as a part of this initiative, and so I first will present the problem of screening for threats in the context of airport security with no uncertainty as modeled in [Brown et al., 2016] and then describe the new model with explicit uncertainty.

5.2.1 The Case when Screenee Arrivals are Known

We consider a finite planning horizon consisting of W time windows (periods) $\mathcal{W} := \{1, \ldots, W\}$. During each period, a *known* number of screenees arrive, each from a *known* category $\kappa := (\rho, \phi)$, $\rho \in \mathcal{P} := \{1, \ldots, P\}$, $\phi \in \mathcal{F} := \{1, \ldots, F\}$. The first (second) component of their category, ρ (ϕ), represents the *uncontrollable* (resp. *controllable*) part of the screenee's category. Thus, each screenee can decide the controllable part of their category, however, they cannot decide the uncontrollable part of their category, which stems from their inherent characteristics. For notational convenience, we let $\mathcal{K} := \mathcal{P} \times \mathcal{F}$. We assume that each screenee knows their own category. As an example, in the context of passenger screening at airports, ρ can represent the risk category of the passenger (e.g., normal boarding versus TSA pre-check), while ϕ can represent a flight type (e.g., international with given departure time) – note that both these components are known to the passenger. We let N_κ^w denote the number of screenees in category κ to arrive in time window w. Since the category and arrival time of each screenee is known, the quantities N_κ^w are perfectly known. Without loss of generality, we assume that $N_\kappa^w > 0$ for all w and κ.

One of the screenees is planning on conducting an attack using an attack method m of his choosing from the set $\mathcal{M}$. For this reason, the screener is operating a checkpoint

comprised of T teams indexed by $t \in \mathcal{T}$ and can decide which team should screen each screenee based on their category. Each of these teams consists of various resource types. The set of all available resource types is denoted by $\mathcal{R}$. The subset of resources composing team t is denoted by $\mathcal{R}(t) \subseteq \mathcal{R}$. If a screenee is assigned to team t, then he must be screened by all resource types allocated to that team. Unfortunately, not all screenees can be screened by the most effective resources as each resource has a capacity C_r on the number of screenees that it can process in each time window. The attack will be averted if the attack method is identified by any one of the resources screening the attacker. We let $E_{t,m}$ denote the effectiveness (ie. probability of interception) of team t at detecting attack method m, determined by the effectiveness of each resource, see Example 4.

Example 4. *Assuming independence of the effectiveness of the resources that make up each team and letting $E^r_{r,m}$ denote the probability of detecting an attack of type m using resource r, we have $E_{t,m} = 1 - \prod_{r \in \mathcal{R}(t)}(1 - E^r_{r,m})$.*

Following the (by now standard) approach in the literature, we formalize this problem as a *Threat Screening Game*, i.e., a Stackelberg game in which the screener, as the leader, commits to mixed strategies, and the attacker acts as the follower [Brown et al., 2016, Schlenker et al., 2016]. The rationale is that the screener acts first by selecting a (randomized) screening strategy, i.e., a feasible assignment of screenees to teams. In response to the choice of screening strategy, the attacker (after observing the screenee allocation) selects: *a)* his attack method m, *b)* his attack window w, and *c)* the components of his category that he can control in κ, so as to cause maximum harm. We refer to such a choice as an attack (m, w, κ). If the attack is caught, the screener receives a utility U^+_κ, which depends on the category of the adversary. Accordingly, if the screener is unsuccessful at preventing the attack, he receives the (negative) utility U^-_κ. The attacker's utilities

are assumed to be negative of the screener's utilities, so that the game is zero-sum. We assume that the defender knows the *probability* that the attacker's uncontrollable category is ρ, denoted by P_ρ and we have $\sum_{\rho \in \mathcal{P}} P_\rho = 1$.The objective of the screener is then to select the best randomized allocation (i.e., mixed strategy), in anticipation of the attacker's best response.

Example 5. *Using the domain of passenger screening at an Airport, we will now provide an illustrative example to instantiate the model. Imagine we are tasked with planning the screening at one of the terminals in a large airport for a 12 hour time period. Using a granularity of 1 hour, we have 12 time windows in the game, so that* $\mathcal{W} = \{1 \dots 12\}$. *During this time there are 5 flights that are departing from this terminal, so that* $\mathcal{F} = \{1 \dots 5\}$. *Each of the flights has a value associated with it, which are given by positive numbers e.g.*$\{100, 100, 200, 100, 500\}$. *If an attack on a flight is successful the defender receives a utility of negative the value of that flight. So if flight* $\phi = 1$ *is successfully attacked the defender receives a utility of* -100 *regardless of the risk level of the passenger, so that* $U_\kappa^{-/+} = U_\phi^{-/+}$. *Contrarily the defender receives a reward of* 0 *for successfully preventing an attack so that* $U_1^- = -100$ *and* $U_1^+ = 0$.

Risk levels are determined by wether a passenger has TSA pre-check or not, and so there are two risk levels $\mathcal{P} = \{1, 2\}$. *Each risk level has a probability associated with it which corresponds to the probability* P *that the adversary may be of that risk level e.g.* $P_1 = 0.8$ *and* $P_2 = 0.2$. *There are then* $2 \times 5 = 10$ *categories of passenger.*

We are concerned about two different types of attacks: explosives and fire-arms, so that $\mathcal{M} = \{1, 2\}$.

In order to screen passengers, at the security checkpoint there are three types of resources: walk-through metal detector (WTMD), and pat-downs to screen people, and

x-ray machines to screen luggage. Each resource has a probability of detecting each attack method $E^r_{r,m}$ and each resource has a capacity for the number of passengers which may be processed in an hour C_r given by:

Resource	*Probability of detection* $\left(E^r_{r,m}\right)$		*Capacity* C_r	*Teams*		
	m = explosive	*m = firearm*	*(per time window)*	t_1	t_2	t_3
x-ray	*0.4*	*0.6*	*50*	✓	✓	✓
WTMD	*0.5*	*0.7*	*30*		✓	✓
Pat-down	*0.9*	*0.9*	*10*			✓

There are three teams which consist of $t_1 = \{$x-ray$\}$, $t_2 = \{$x-ray, WTMD$\}$, $t_3 = \{$x-ray, WTMD, pat-down$\}$. Passengers are assigned to teams of resources, so for example, if a passenger were to be assigned to t_2 they would be screened by the x-ray machine and the WTMD. Teams must share the pool of available resources, so that if there is only one x-ray machine, then all teams must share the same machine. The efficiency of each team is given by the probability of each resource detecting an attack (computed as shown in example 4) so that:

Team	*Probability of detection* $(E_{t,m})$	
	m = explosive	*m = firearm*
t_1	*0.4*	*0.6*
t_2	*0.8*	*0.88*
t_3	*0.97*	*0.988*

We are now ready to provide a mathematical formulation of the problem in the spirit of [Brown et al., 2016].

Defender Pure Strategy Set. An assignment of screenees to teams occurs at the beginning of each period $w \in \mathcal{W}$, and corresponds to a decision on the number of screenees from each category κ to allocate to each team t out of the N_κ^w screenees that arrive in that time window. Letting $\nu_{\kappa,t}^w$ denote this assignment, the defender pure strategy set is given by

$$\mathcal{S} := \left\{ \nu \; : \; \nu_{\kappa,t}^w \in \mathbb{N}_+ \; \forall t \in \mathcal{T}, \; \sum_{t \in \mathcal{T}} \nu_{\kappa,t}^w = N_\kappa^w \; \forall \kappa \in \mathcal{K}, \right.$$
$$\left. \sum_{t : r \in \mathcal{R}(t)} \sum_{\kappa \in \mathcal{K}} \nu_{\kappa,t}^w \leq C_r \; \forall r \in \mathcal{R}, w \in \mathcal{W} \right\}.$$

The first constraint in the set stipulates that the number of screenees must be a non-negative integer. The second ensures that all the screenees are allocated to a team. The third guarantees that resource capacities are not exceeded. Note that $\mathcal{S}$ has finite cardinality, i.e., there are finitely many pure strategies available to the screener. The probability of detecting an attack (m, w, κ) given defender strategy s is given by

$$D_{\kappa,m}^{w,s} := \sum_{t \in \mathcal{T}} E_{t,m} \nu_{\kappa,t}^{w,s} / N_\kappa^w,$$

where $\nu_{\kappa,t}^{w,s}$ denotes the number of screenees in category κ screened by team t in window w according to pure strategy s.

Defender Mixed Strategies. A mixed strategy corresponds to a distribution over pure strategies, i.e., to a choice

$$q \in \mathcal{Q} := \left\{ (q_s)_{s \in \mathcal{S}} : \sum_{s \in \mathcal{S}} q_s = 1,\ q_s \geq 0 \right\}.$$

The probability of detecting an attack (m, w, κ) is given by $\sum_{s \in \mathcal{S}} q_s D^{w,s}_{\kappa,m}$.

Robust Linear Programming Formulation. Since the attacker can select his attack (m, w, κ), but cannot select the uncontrollable aspect of his category, the problem faced by the screener is expressible as the following robust optimization problem in variables z and q

$$\begin{aligned} \text{maximize} \quad & \min_{w,m,\phi} \sum_{\rho \in \mathcal{P}} P_\rho [z^w_{\kappa,m} U^+_\kappa + (1 - z^w_{\kappa,m}) U^-_\kappa] \\ \text{subject to} \quad & z^w_{\kappa,m} = \sum_{s \in \mathcal{S}} q_s D^{w,s}_{\kappa,m} \qquad \forall \kappa,\ m,\ w \\ & q \in \mathcal{Q}. \end{aligned} \tag{5.1}$$

We have omitted the sets of the variables κ, m, w and ϕ to minimize notational overhead. The variable $z^w_{\kappa,m}$ is the probability of detecting an attack (m, w, κ). Accordingly, the objective function corresponds to the worst-case expected utility of the screener. The expectation is taken with respect to the uncontrollable component of the attacker's category. The minimum is taken across all choices available to the attacker.

The cardinality of the strategy set $\mathcal{S}$ (and accordingly the number of decision variables in Problem (5.1)) is exponential in the number of time windows and Problem (5.1) is $\mathcal{NP}$-hard [Brown et al., 2016]. We thus consider a relaxation to Problem (5.1) obtained by

performing the change of variables $\pi^w_{\kappa,t} := \sum_{s\in\mathcal{S}} q_s n^{w,s}_{\kappa,t}/N^w_\kappa$. The variable $\pi^w_{\kappa,t}$ can be interpreted as the (marginal) probability of allocating a screenee in category κ to team t in window w. We obtain the following robust linear problem in variables z and π whose size is polynomial in the number of time windows

$$\begin{array}{lll} \text{maximize} & \min\limits_{w,m.\phi} \sum\limits_{\rho\in\mathcal{P}} P_\rho[z^w_{\kappa,m}U^+_\kappa + (1-z^w_{\kappa,m})U^-_\kappa] & \\ \text{subject to} & z^w_{\kappa,m} = \sum\limits_{t\in\mathcal{T}} E_{t,m}\pi^w_{\kappa,t} & \forall\kappa, m, w \\ & \pi \in \Pi. & \end{array} \tag{5.2}$$

The first constraint is a direct consequence of the first constraint in Problem (5.1) combined with the change of variables, and

$$\Pi := \left\{ \pi \; : \; \begin{array}{ll} \sum\limits_{t:r\in\mathcal{R}(t)} \sum\limits_{\kappa\in\mathcal{K}} \pi^w_{\kappa,t} N^w_\kappa \le C_r & \forall r, w \\ \sum\limits_{t\in\mathcal{T}} \pi^w_{\kappa,t} = 1 & \\ 0 \le \pi^w_{\kappa,t} \le 1 \quad \forall t & \end{array} \right\} \quad \forall w, \kappa. \Bigg\}$$

denotes the set of all marginal strategies. We note that Problem (5.2) is equivalent to a moderately sized linear program obtained by linearizing the piecewise linear concave objective function using the standard epigraph reformulation approach.

5.2.2 The Case of Uncertain Screenee Arrivals

Insofar, we have assumed that screenee arrival times are perfectly known. Unfortunately, this assumption fails to hold in most threat screening problems. Moreover, ignoring uncertainty in the screenee arrivals during optimization may yield severely suboptimal or even infeasible allocations, see Section 5.4. We thus develop a novel modeling and solution framework for threat screening that is robust to uncertainty in screenee arrival times. Our framework builds upon formulation (5.2) which enjoys better tractability properties than Problem (5.1).

Model of Uncertainty. We model the number of screenees from each category to arrive in each time window as random variables that are defined on the probability space $(\Xi, \mathcal{F}, \mathbb{P})$, which consists of the sample space Ξ, the Borel σ-algebra $\mathcal{F}$ and the probability measure $\mathbb{P}$. The elements of the sample space are denoted by $\xi := (\xi_0, \xi_1, \ldots, \xi_W)$ where the subvector $\xi_w := (\xi_{w,\kappa})_{\kappa \in \mathcal{K}}$ is observed at the end of period w and $\xi_{w,\kappa}$ represents the number of people from category κ that arrive in window w. We also let $\xi^w := (\xi_0, \ldots, \xi_w)$ denote the portion of ξ that has been observed by the end of time window w. We assume that Ξ is a bounded set expressible as

$$\Xi \;:=\; \{\xi \;:\; \xi_{w,k} \in \mathbb{N},\; V\xi \leq h\} \tag{5.3}$$

for some matrix $V \in \mathbb{R}^{\ell \times WK}$ and vector $h \in \mathbb{R}^{\ell}$, where ℓ corresponds to the number of constraints in the uncertainty set. Thus Ξ corresponds to the intersection of the set of all non-negative integers with a polyhedral set. Without loss of generality, we assume that $\Xi \subset \{\xi \;:\; \xi_0 = 1\}$ (since $w = 0$ is not a valid time period, we let ξ_0 be a constant,

so that affine functions of $(\xi_w)_{w\in\mathcal{W}}$ can be represented compactly as linear functions of ξ). We assume that Ξ is bounded. In the spirit of robust optimization, we refer to Ξ as the *uncertainty set*. We note that polyhedral uncertainty sets allow for a lot of modeling flexibility and enable us to capture a wide variety of constraints of practical relevance such as in the airport screening domain.

Example 6 (Airport Screening)**.** *In the context of security screening at airports, the total number of people to travel in category κ on a given day, denoted by N_κ is known from the flight manifests. At the same time, passenger arrival times are conditioned by the time of their flight category ϕ. It is thus natural to assume that all passengers in category κ will arrive in some window $w \in \Delta_\kappa \subseteq \mathcal{W}$ (covering e.g., a couple of hours before their flight time). A suitable choice of uncertainty set is then given by*

$$\Xi_{\mathrm{AS}} := \left\{ \xi \ : \ \xi_{w,k} \in \mathbb{N}_+, \ \sum_{w\in\Delta_\kappa} \xi_{\kappa,w} = N_\kappa \ \forall \kappa \right\},$$

which we denote by AS *for Airport Screening.*

In this paper, we take the view of a risk-averse screener that wishes to be immunized against all possible realizations of $\xi \in \Xi$. This view point is very natural for the set of applications under consideration that fall under the realm of security. This implies that the attacker can in some sense "strategize with nature" to devise a maximally harmful attack. Equivalently, it can be interpreted as the desire to be immunized against an attacker who would, by his own fortune, select the maximally harmful attack relative to uncertainty in arrivals.

Adaptive Screening. As information about screenee arrivals is revealed sequentially over time, the screener has the opportunity to adjust his screening policy in an adaptive fashion, at the beginning of each time window, in response to these observations. In particular, at the beginning of time window w, the screenee has observed the sequence of past arrivals ξ^{w-1} and can use that information to reason about uncertainty in remaining time windows and adjust his screening strategy accordingly. Mathematically, the screening decisions made at the beginning of time window w (i.e., π_w) in Problem (5.2) must be modeled as functions of the history of screenee arrivals ξ^{w-1}. Given a realization $\tilde{\xi}^{w-1}$ of ξ^{w-1}, the screener will allocate $\pi^w_{\kappa,t}(\tilde{\xi}^{w-1})$ percent of screenees of category κ to team t in window w. Accordingly, the probability of intercepting an attacker from category κ using attack method m in time window w (i.e., $z^w_{\kappa,m}$) also depends on the realization of ξ^{w-1} and must be modeled as a function of the history of observations, i.e., we have $z^w_{\kappa,m}(\xi^{w-1})$.

Resource Overflow. When arrivals are uncertain, the resource capacity constraint in (5.2) reads

$$\sum_{t:r\in\mathcal{R}(t)} \sum_{\kappa\in\mathcal{K}} \pi^w_{\kappa,t}(\xi^{w-1})\xi_{w,\kappa} \leq C_r \ \forall r \in \mathcal{R}, w \in \mathcal{W}, \xi \in \Xi.$$

It requires that *for all possible realizations of screenee arrivals*, the allocation must be such that all screenees be screened by available resources in the window in which they arrive. This may lead to highly conservative strategies that allocate most (if not all) screenees to the team with the highest capacity. To mitigate such over-conservatism, we propose to allow each resource $r \in \mathcal{R}$ to *overflow* from one time window to the next at a cost F_r per screenee that is delayed. Thus, each screenee is allocated to a team in the window

in which they arrive. However, screening by some (or all) of the resources in that team may take place in a future time window if that resource is over-capacity. The higher the overflow fine F_r, the least likely that resource r will be overcapacity. We note that similarly to the screening policy, the number of screenees to overflow in each resource from time window w to time window $w+1$, denoted o_r^{w+1}, must be modeled as functions of the history of screenee arrivals, ξ^w. Under these considerations, the resource capacity constraint becomes

$$\sum_{t:r\in\mathcal{R}(t)} \sum_{\kappa\in\mathcal{K}} \pi_{\kappa,t}^{w}(\xi^{w-1})\xi_{w,\kappa} \leq C_r - o_r^w(\xi^{w-1}) + o_r^{w+1}(\xi^w) \tag{5.4}$$

and is enforced for all $r \in \mathcal{R}$, $w \in \mathcal{W}$, and $\xi \in \Xi$.

Adaptive Robust Optimization Formulation. We now formulate the screener's problem as a multi-stage robust optimization problem. We note that if the attacker chooses category κ and time window w for his attack, at least one screenee in category κ (corresponding to the attacker) must arrive in that time window, i.e., it must hold that $\xi_{w,\kappa} > 0$. The screener's problem may be formulated in epigraph form as

$$\begin{aligned}
\text{maximize} \quad & \theta \\
\text{subject to} \quad & \theta \leq \sum_{\rho\in\mathcal{P}} P_\rho u_\rho - \sum_{w\in\mathcal{W}} \sum_{r\in\mathcal{R}} F_r o_r^w \quad \forall \xi \\
& u_\rho \leq z_{\kappa,m}^w U_\kappa^+ + (1 - z_{\kappa,m}^w) U_\kappa^- \quad \forall \xi : \xi_{w,\kappa} > 0 \\
& z_{\kappa,m}^w = \sum_{t\in\mathcal{T}} E_{t,m} \pi_{\kappa,t}^w \qquad\qquad \forall \xi, \kappa, m, w \\
& \pi \in \Pi_{\text{o}}.
\end{aligned} \tag{$\mathscr{P}$}$$

The decision variables of Problem ($\mathscr{P}$) are $\theta \in \mathbb{R}$, $u_\rho(\xi)$, $o_r^w(\xi^{w-1})$, $z_{\kappa,m}^w(\xi^{w-1})$, $\pi_{\kappa,t}^w(\xi^{w-1}) \in \mathbb{R}$, and

$$\Pi_{\mathrm{o}} := \left\{ \begin{array}{ll} & \exists o \text{ with } o_r^w \geq 0 : \text{Constraint (5.4) } \forall \xi, r, w \\ \pi \ : & \sum_{t \in \mathcal{T}} \pi_{\kappa,t}^w = 1 \\ & 0 \leq \pi_{\kappa,t}^w \leq 1 \quad \forall t \end{array} \right\} \quad \forall \xi, w, \kappa \Bigg\}.$$

We omit the dependence on ξ to minimize notational overheard. The variables $u_\rho(\xi)$ express the utility of the screener in scenario ξ when the uncontrollable category of the screener is ρ. The remaining variables admit the same interpretation as in Section 5.2.1. In the present setting they are however *adaptive*. The first set of constraints is used to linearize the piecewise linear concave objective function. The second set of constraints determines the worst-case value of $u_\rho(\xi)$ for each scenario ξ. For any given choice of (κ, w, m) by the attacker, this constraint is only enforced over those $\xi \in \Xi$ for which $\xi_{w,\kappa} > 0$ since at least one screenee must arrive in the attacker's chosen category and attack window.

Example 7. *Continuing the scenario from example 5, the defender then must choose how many of each passenger from each of the 10 categories of passenger to send to each of the 3 teams, for each of the 12 time windows, given by $\pi_{\kappa,t}^w$. But since the defender does not know how many people will arrive in any time window they do this by choosing a percentage of the people who will arrive to screen. So for example, in time window 2, given $\xi = x$ number of passengers have already arrived, if the defender chooses to screen 5 passengers of category 1, corresponding to risk level 1 (no tsa pre-check) and on flight 1,*

with team 2, we would have $\pi^2_{1,2}(x) = 0.5$*. This may vary with the number of passengers who have arrived in the previous time window, so that for a different* $\xi = x'$ *the defender may choose a different number of passengers to screen* $\pi^2_{1,2}(x') = 0.7$*.*

The following Proposition establishes correctness of the above formulation by showing equivalence of Problem ($\mathscr{P}$) and an appropriately constructed robust dynamic program.

Proposition 1. *The multi-stage robust optimization problem* ($\mathscr{P}$) *computes the optimal defender screening strategy, which maximizes his worst-case expected utility when screenee arrivals are uncertain. It is always feasible.*

Proof. We derive the game formulation using backward induction starting from the last time window of the TSG, where there is no uncertainty.

Base Case: Last time window. At the last time window W there is no uncertainty, the defender knows that ξ^{W-1} passengers have arrived, and there are currently o^{W-1} passengers overflowed. This is a single stage deterministic TSG given in problem (2) taken at a single time window W, with the addition of overflow. Since this is the last time window, it must be that the adversary has not yet attacked and thus will attack in this time window.

The adversary then has a choice of κ and m so that the actions space is

$$A = \left\{ a_{\kappa,m} \in \{0,1\} : \sum_{\kappa,m} a_{\kappa,m} \right\}$$

.

The defender chooses a screening strategy $\pi \in \Pi_o^w(\overline{\xi}^{w-1})$ where

$$\Pi_o^w(\overline{\xi}^{w-1}) = \left\{ \pi_{\kappa,t}^w : \begin{array}{ll} \sum_{t:r\in\mathcal{R}(t)} \sum_{\kappa\in\mathcal{K}} \pi_{\kappa,t}^w(\overline{\xi}^{w-1})\xi_{w,\kappa} \leq & C_r - o_r^w(\overline{\xi}^{w-1}) \\ & \quad + o_r^{w+1}(\xi^w) \;\; \forall \xi_w : \xi^{w-1} = \overline{\xi}^{w-1} \\ \sum_{t\in\mathcal{T}} \pi_{\kappa,t}^w = 1 & \\ 0 \leq \pi_{\kappa,t}^w \leq 1 \quad \forall w,\kappa,t & \end{array} \right\}$$

we can equivalently express the first constraint as: so that the equilibrium utility for the defender is then:

$$\begin{array}{rll} f^W(\xi^{W-1}, o_r^{W-1}) = & \max\limits_{\pi^W} & \sum\limits_{\kappa,m} P_\rho a_{\kappa,m} u_{\kappa,m}^W - \sum\limits_r F_r o_r^W \\ & \text{st.} & u_{\kappa,m}^W = (z_{\kappa,m}^W U_\kappa^+ + (1 - z_{\kappa,m}^W) U_\kappa^-) \\ & & z_{\kappa,m}^W = \sum\limits_{t\in\mathcal{T}} E_{t,m} \pi_{\kappa,t}^W \\ & & \pi_{\kappa,t}^W \in \Pi_o^W(\xi^{W-1}) \end{array}$$

Adding in Uncertainty: Time Window w. In the second to last time window $W-1$ we know that ξ^{W-2} screenees have arrived and there are o^{W-2} screenees overflowed. The problem is no longer deterministic, as we are now unsure of how many screenees will arrive. All we know us that the number of screenees must be in the range $[0, N_\kappa - \sum_{i=1}^{W-1} \xi_{W-2,\kappa}]$ so that $\xi^{W-1} \in \Xi^{W-1}(\overline{\xi}^{W-2})$ for this time window.

$$\Xi^{W-1}(\overline{\xi}^{W-2}) = \{\xi^{W-1} \in \Xi,\ \xi_{W-1,\kappa} \leq N_\kappa - \sum_{i=1}^{W-2} \overline{\xi}_{i,\kappa}\}$$

We must now plan for the remaining two time windows, and we assume the adversary will attack either in this time window $W-1$ or in the last time window W. The adversary has an

additional choice $a_w \in \{0, 1\}$ of whether to attack in time window w or not. We know that if the adversary attacks in the next time window, the utility we will get is $f^W(\xi^{W-1}, o_r^{W-1})$. If the attack occurs in this time window then we will get a utility $\sum_\kappa P_\rho u_\kappa^{W-1} - \sum_r F_r o_r^{W-1}$ determined by our strategy in this time window. We can define a recurrence relation for any time window w where:

$$
\begin{aligned}
f^w(\xi^{w-1}, o^{w-1}) \quad &= \max_{\pi^w} \min_{a^w} \min_{a_{\kappa,m}} \min_{\xi^w} a_w \Big[\sum_\kappa P_\rho a_{\kappa,m} u_{\kappa,m}^W - \sum_r F_r o_r^W \\
&\quad + (1 - a_w) f^{w+1}(\xi^w, o^w) \Big] \\
&\quad \text{s.t.} \quad \pi_w \in {\Pi_o}^w(\xi^{w-1}) \quad a^w \in \{0, 1\} \quad a_{\kappa,m} \in A \\
&\qquad \xi^w \in \Xi^w = \{\xi_\kappa^w \in \Xi : \ \xi_{w,\kappa} \leq N_\kappa - \textstyle\sum_{i=1}^{w-1} \xi_{i,\kappa}\}
\end{aligned}
\tag{$\mathcal{B}$}
$$

$$
f^W(\xi^{W-1}, o^{W-1}) \quad = \max_{\pi^W} \min_{a_{\kappa,m}} \sum_{\kappa,m} P_\rho a_{\kappa,m} u_{\kappa,m}^W - \sum_r F_r o_r^W
$$

So that the optimal defender utility is given by $f^1(\emptyset, \emptyset)$. By then expanding out the recurrence relation we get a sequence sequence $a_1 + (1 - a_1)(a_2 + (1 - a_2)(\dots a_{W-1} + (1 - a_{W-1}))$ with W terms. Since each a_i is binary valued, we can equivalently can express

the sequence as $a_1 + a_2 + \dots a_W$ with the additional constraint that and $\sum a_i = 1$. The objective can equivalently be written:

$$\max_{\pi \in \mathbf{\Pi}} \min_{\mathbf{a} \in \mathbf{A}} \min_{\xi \in \mathbf{\Xi}} \big[\sum_{w} \big[a_w \sum_{\kappa,m} P_\rho a_{\kappa,m} u^w_{\kappa,m} - \sum_{w',r} F_r o^{w'}_r \big] \big]$$

$$\max_{\pi \in \mathbf{\Pi}} \min_{\mathbf{a} \in \mathbf{A}} \min_{\xi \in \mathbf{\Xi}} \big[\sum_{w} \sum_{\kappa,m} P_\rho a_w a_{\kappa,m} u^w_{\kappa,m} - \sum_{w} a_w \sum_{w',r} F_r o^{w'}_r \big]$$

$$\max_{\pi \in \mathbf{\Pi}} \min_{\mathbf{a} \in \mathbf{A}} \min_{\xi \in \mathbf{\Xi}} \big[\sum_{w} \sum_{\kappa,m} P_\rho a^w_{\kappa,m} u^w_{\kappa,m} - \sum_{w',r} F_r o^{w'}_r \big]$$

$$A = \{a \in \{0,1\}^{|\mathcal{C}||\Delta||\mathcal{M}|} \ : \ \mathbb{1}^\top a = 1\}$$

$$\Xi = \{\xi : \mathbb{1}^\top \xi = N, \xi \geq 0 \quad \forall w \in W\}$$

$$\Pi_o = \{\cup_w^W \Pi_o^w(\xi^{w-1}) \quad \forall \xi \in \Xi\}$$

Where we have combined the two variables a_w and $a_{\kappa,m}$ into a single variable $a^w_{\kappa,m}$. We can express the minimization over Ξ equivalently using the standard epigraph reformulation:

$$\begin{aligned} \max_{\pi \in \mathbf{\Pi}} \min_{\mathbf{a} \in \mathbf{A}} \quad & \theta \\ & \theta \leq \big[\sum_{\kappa,m} P_\rho a^w_{\kappa,m} u^w_{\kappa,m} - \sum_r F_r o^w_r \big] \quad \forall \xi \in \Xi \\ & u^w_{\kappa,m} = (z^W_{\kappa,m} U^+_\kappa + (1 - z^W_{\kappa,m}) U^-_\kappa) \end{aligned}$$

In order to get rid of the integer valued variables a, we can do the same reformulation for the minimization over the A using big M constraints, knowing that :

$$\max_{\pi \in \mathbf{\Pi}} \quad \theta$$

$$\theta \leq \big[\sum_{\rho} P_\rho u_\rho - \sum_{r} F_r o_r^w \big] \qquad \forall \xi \in \Xi$$

$$0 \leq u_\rho - (z^w_{\kappa,m} U^+_\kappa + (1 - z^w_{\kappa,m}) U^-_\kappa) \leq (1 - a^w_{\kappa,m}) M \quad \forall \xi \in \Xi,\ a^w_{\kappa,m} \in A$$

For some w', κ', m' we will have $a^{w'}_{\kappa',m'} = 1$ and $a^w_{\kappa,m} = 0 \quad \forall w \neq w', \kappa \neq \kappa', m \neq m'$. So that:

$$a^w_{\kappa,m} \leq \xi_{w,\kappa} \quad \forall w, \kappa, m \Leftrightarrow 1 \leq \xi_{w',\kappa'}, \ \ 0 \leq \xi_{w,\kappa} \ \ \forall w \neq w', \kappa \neq \kappa', m \neq m'$$

The first constraint can then be expressed:

$$0 \leq u_\rho - (z^{w'}_{\kappa',m'} U^+_{\kappa'} + (1 - z^{w'}_{\kappa',m'}) U^-_{\kappa'}) \leq 0 \qquad 1 \leq \xi_{w',\kappa'}, \xi \in \Xi$$

$$0 \leq u_\rho - (z^w_{\kappa,m} U^+_\kappa + (1 - z^w_{\kappa,m}) U^-_\kappa) \leq M \qquad \forall w \neq w', \kappa \neq \kappa', m \neq m'$$

$$\xi \in \Xi$$

Any feasible solution must have $u_\rho = (z^{w'}_{\kappa,m} U^+_\kappa + (1 - z^{w'}_{\kappa,m}) U^-_\kappa)$ with $\xi_{w',\kappa} \geq 1$ and that $u_\rho \leq (z^w_{\kappa,m} U^+_\kappa + (1 - z^w_{\kappa,m}) U^-_\kappa) \quad \forall w$ so we can re-write this constraint as $u_\rho \leq (z^w_{\kappa,m} U^+_\kappa + (1 - z^w_{\kappa,m}) U^-_\kappa) \quad \forall w, \xi_{w,\kappa} \geq 1$ The problem then becomes:

$$
\begin{aligned}
\text{maximize} \quad & \theta \\
\text{subject to} \quad & \theta \leq \sum_{\rho \in \mathcal{P}} P_\rho u_\rho - \sum_{w \in \mathcal{W}} \sum_{r \in \mathcal{R}} F_r o_r^w \quad \forall \xi \\
& u_\rho \leq z_{\kappa,m}^w U_\kappa^+ + (1 - z_{\kappa,m}^w) U_\kappa^- \quad \forall \xi \in \Xi' \qquad (\mathscr{P}) \\
& z_{\kappa,m}^w = \sum_{t \in \mathcal{T}} E_{t,m} \pi_{\kappa,t}^w \qquad \forall \xi, \kappa, m, w \\
& \pi \in \Pi_{\text{o}}.
\end{aligned}
$$

□

Complexity. Since Ξ is discrete and bounded, Problem ($\mathscr{P}$) is equivalent to a deterministic linear program obtained by enumerating all possible realizations of $\xi \in \Xi$ and imposing appropriate non-anticipativity constraints, in the spirit of scenario-based stochastic programming [Birge and Louveaux, 1997]. While the numbers of decision variables and constraints in that problem is linear in the number of scenarios, the number of scenarios (cardinality of Ξ) can grow very large, as illustrated by the following example.

Example 8 (Airport Screening)**.** *Consider the uncertainty set Ξ_{AS} from Example 6. For any fixed screenee category κ, the number of possible ways in which these screenees may arrive is*

$$g := \binom{N_\kappa + |\Delta_\kappa| - 1}{N_\kappa}.$$

For fixed $|\Delta_\kappa|$ this quantity is $\mathcal{O}(N_\kappa^{|\Delta_\kappa|})$; and for fixed N_κ, it is $\mathcal{O}(|\Delta_\kappa|^{N_\kappa})$. Since passenger arrivals are independent across different categories, the cardinality of $\Xi_{\mathcal{AS}}$ is given by $g^{|\mathcal{K}|}$ and is thus exponential in the number of categories. In the context airport screening,

the number of scenarios is thus exponential in the number of flight categories. In addition, both the number of flight categories and corresponding number of passengers are generally linear in the number of time windows. This implies that the size of the corresponding scenario problem is exponential in the number of time windows.

5.2.3 Defender Strategies as Markov Decision Processes

The robust optimization problem ($\mathscr{P}$) emits solutions which are adaptive policies. These policies are similar to the policy solutions to the decision theoretic models known as Markov Decision Processes (MDP), which are mathematical model for sequential decision making under uncertainty. In order to illustrate the connection to this existing work, we derive the corresponding MDP representation of the robust threat screening game model. We present here a formal model of *Robust Threat Screening Games (RTSG)* expressed using MDPs.

Defender

We can represent the set of valid defender strategies Π as the set of valid policies strategy for the following finite horizon MDP($\mathcal{S}, \mathcal{A}, T, R$):

States: Let the set of states be $\mathcal{S} := \{(\xi, o)\}$ where a state $s \in \mathcal{S}$ is a tuple of ξ, the number passengers in each category κ who have already arrived, and size of the overflow

queues o on each resource r at time window w.

$$\mathcal{S} := \left\{ (\xi, o) \ : \ \xi_{\kappa,w}, o_{r,w} \in \mathbb{N}_+ \ \xi_{\kappa,w} \leq N_\kappa \right.$$
$$\left. o_{r,w} \leq o_{r,w-1} + \sum_{\kappa} \xi_{\kappa,w} - C_r \ \ \forall r \in \mathcal{R}, w \in \mathcal{W} \right\}.$$

Where C_r is the capacity of resource r and N_κ is the total number of passengers in category κ.

Actions: Let the set of action $\mathcal{A}$ be the set of screening assignments $a^w = (a^w_{1,1,1} \dots a^w_{n,\kappa,t})$, indicating the assignment in time window w, of the n^{th} arriving passenger in each category κ to be screened with particular team t.

$$\mathcal{A} := \left\{ \ a \ : \ a^w_{n,\kappa,t} \in \{0, 1\} \ \right\}.$$

Because we don't know exactly how many passengers will arrive in each time window, the defender pure strategy is a matching for all passenger *which may arrive*. An example of the possible ξ components of the state space and corresponding policy is shown in Figure 5.2 for an example with 2 passengers. In the first time window, the defender does not know if 0, 1, or 2 people will show up in the next time window. The defender needs to come up with a plan, but cannot condition the plan on information they do no have (ie. how many people will show up), thus the plan needs to be the same in all three of these scenarios. If nobody shows up in the next time window, none of the plan is executed. If one person shows up the defender should screen them according to their plan, but since they do no know if this will be the only person who will show up or if more passengers will

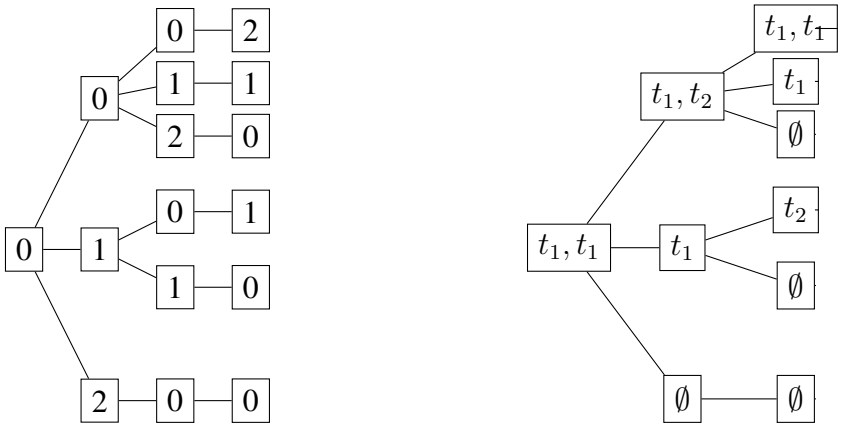
come, they need to screen them the same way, regardless of how many passengers arrive after them. In each state they need to commit to a sequence of screening assignments, which corresponds to an assignment for each passenger which may show up, having a length equal to the the maximum number of possible passengers which may show up. The original RTSG solved directly for the defender's mixed strategy. We will later show how this mixed strategy can be derived from this pure strategy representation.

Transition function: Let $T(s^w, a, s^{w+1})$ defines the probability of transitioning to state $s^{w+1} \in \mathcal{S}$ from $s^w \in \mathcal{S}$ given the defender chooses action $a^w \in \mathcal{A}$. This function is determined by the stochastic process that governs the passenger arrivals as well as the and can be written $T(s^w, a^w, s^{w+1}) = P(o_{w+1}|o_w, \xi_w, \xi_{w+1}a^w)P(\xi_{w+1}|\xi_w)$. The defender's action a^w affects the state transition in that it will affect the overflow queues in the next round o_{w+1}.

Reward: Let the immediate reward $R(s^w, a^w, s^{w+1})$ be the penalty the defender pays in time window w for the length of the overflow queues on all the resources r, where ϕ_r is the fee for keeping one passenger waiting on resource r. This is the reward the defender obtains independent of their interactions with the adversary.

$$R(s^w, a^w, s^{w+1}) = \sum_r \phi_r o_r^{w+1}$$

$$o_r^{w+1} = \max\left(\sum_{t:r\in t,\kappa} \sum_{n=1}^{\xi_\kappa^{w+1}} a_{n,\kappa,t}^w + o_r^w - C_r, 0\right)$$



(a) Possible ξ states in the MDP (b) An example pure strategy or policy.

Figure 5.2: Example state transition, and corresponding MDP policy for a small game with 4 time windows, one category, two teams (t_1, t_2), and two passengers.

Given a particular series of states $(s_1 \dots s_W)$

$$\sum_{s_w} R(s^w, a^w, s^{w+1})$$

measures the throughput efficiency of a defender policy.

Adversary

The adversary is a separate agent which takes actions in the world along side the defender. The adversary may be one of several types ρ, which corresponds to their risk level. An adversary has an action space $Q = \{q^w_{m,\phi}\}$ corresponding to a choice of attack method m, flight ϕ and time window w. The risk level along with the flight choice ϕ can be generalized to an assigned risk category $\kappa = (\rho, \phi)$, as these are treated as a single feature by the defender. The probability distribution over the adversary types is given by P_ρ.

The actions available to the adversary are denoted $q^w_{m,\phi} \in \{0,1\}$ and represent the whether the adversary is choosing to attack flight ϕ with attack method m in time window w. Note that the adversary can only choose a single, flight, attack method and time window, and must attack at some point in the game, so that $\sum_{w,m,\phi} q^w_{m,\phi} = 1$. The adversary cannot control how other passengers in their time window arrive and therefore cannot choose what order they arrive in. If they choose category κ as part of their attack, there will be some probability $P(n, \xi_{w+1,\kappa})$ that they are the n^{th} passenger in category κ to arrive, where $\sum_n P(n, \xi_{w+1,\kappa}) = \sum_m q^w_{m,\phi} \quad \forall \rho$. The immediate joint reward received from the defender-adversary interaction is:

$$C(s^{w+1}, a^w, q^w) = q^w_{m,\phi} \left(\sum_{m,\kappa} P_\rho u^{w+1}_{\kappa,m}(\xi^{w+1}) \right)$$

$$u^{w+1}_{\kappa,m}(\xi^{w+1}) = z^{w+1}_{\kappa,m}(\xi^{w+1}) U^+_\kappa + (1 - z^{w+1}_{\kappa,m}(\xi^{w+1})) U^-_\kappa$$

$$z^{w+1}_{\kappa,m}(\xi^{w+1}) = \sum_{n,t} E^t_m a^w_{n,\kappa,t} P(n, \xi_{w+1,\kappa})$$

$$q^w_{m,\phi} = \begin{cases} 0 & \text{if no-attack} \\ 1 & \text{if attack} \end{cases}$$

We assume that the adversary is able to conduct multiple rounds of surveillance and observe the defender's mixed strategy over all time windows; however the adversary still does not know the future arrival of passengers, and thus their policy is a function of

the state and defender policy $\pi_q : (\pi_d, \xi) \rightarrow q$. Given a particular sequence of states $(s_1 \dots s_W)$

$$\sum_{s_w} C(s^{w+1}, a^w, q^w)$$

measures the screening efficiency of the defender policy.

Nature

Nature determines the arrival rate of passengers and influences the state transitions and the immediate reward received by the defender $R(s^w, a^w, s^{w+1})$. In the event that we do not know what the transition matrix looks like, we can choose to be robust and assume the worst case possible transition matrix which would minimize the cumulative rewards we receive at each time step and correspondingly the value function at the start state. Let $\mathcal{T}$ be the set of all valid transition matrices. The problem them becomes:

$$\min_{T \in \mathcal{T}} V(s^0)$$

Since we are now taking a robust approach, we also assume that the adversary also does not know the transition matrix but knows we are optimizing for the worst case terminal state. With this it is as if the adversary and nature are 'colluding' to minimize the value function. In the orignal Robust Threat Screening Game formulation, equal weight is given to both the screening efficiency and the throughput efficiency. Assuming the adversary chooses to follow an optimal policy, we can express the value function as:

$$V^{\pi_d}(s^0) = \min_{q \in Q} \min_{T \in \mathcal{T}} \sum_{s^{w+1}} T(s^w, a^w, s^{w+1}) \left(R(a^w, s^{w+1}) \right.$$

$$\left. + C(s^{w+1}, a^w, q^w) + V(s^{w+1}) \right)$$

Compact Policy

Randomized Representation Because the adversary's policy is a function of the defender's policy, the optimal defender policy is a randomized policy. Rather than representing this randomized policy as a mixture over individual actions or deterministic policies $\pi(s^w)_i \rightarrow (a^w)_i$, where each policy is played with probability Θ_i we can more compactly represent the policy by marginalizing over the actions. For each action a^w, the fraction of passengers in each category κ sent to each team t is $\sum_n a^w_{n,\kappa,t} P(n, \xi_{w+1,\kappa})$, which depends on the current number of passengers arriving $\xi_{w+1,\kappa}$.

However, because we are being robust to nature, for any passenger arrival $\xi_{w+1,\kappa}$ we can assume that the ordering of passengers will be such that the adversary arrives in position $n' = \text{argmin}_n \sum_t E^t_m a^w_{n,\kappa,t}$ to minimize the defender's screening efficiency. For any arrival of passengers $\xi_{w+1,\kappa}$, because the adversary ends up arriving in the worst position for the defender we need randomize over the permutations of assignments. What this means is that if there is a deterministic policy π_i in the support of our randomized policy which assigns people to teams in some order eg. (t_1, t_2, t_1, t_2) in order to make the adversary indifferent to the position they arrive in, the randomized policy must also randomize over a set other policies such that each position has the same probability of being screened by each team.

Let this probability be denoted $\pi^w_{\kappa,t}$. We can then compactly represent the randomized policy by reasoning directly about these probabilities rather than the distribution over actual deterministic policies so that $\pi^w_{\kappa,t} = \sum_i \Theta_i(a^w_{n',\kappa,t})_i$.

Compact Reward The reward functions can equivalently expressed with this new representation:

$$R(s^w, \pi^d(s^w), s^{w+1}) = \sum_r \phi_r o_r^{w+1}$$

$$o_r^{w+1} = \max\left(\sum_{t:r\in t,\kappa} \sum_{n=1}^{\xi^w_\kappa} \pi^w_{n,\kappa,t}\xi_{w+1,\kappa} + o_r^w - C_r, 0\right)$$

$$C(s^{w+1}, \pi^d(s^w), q^w) = q^w_{m,f}\left(\sum_{m,\kappa} P_\rho u^{w+1}_{\kappa,m}(\xi^{w+1})\right)$$

$$u^w_{\kappa,m}(\xi^{w+1}) = z^{w+1}_{\kappa,m}(\xi^{w+1})U^+_\kappa + (1 - z^{w+1}_{\kappa,m}(\xi^{w+1}))U^-_\kappa$$

$$z^{w+1}_{\kappa,m}(\xi^{w+1}) = \sum_{n,t} E^t_m \pi^w_{n,\kappa,t}(\xi^w)$$

Which is the exact randomized policy solved for in Problem ($\mathscr{P}$).

Compact Transitions The transition matrix is now a function of π so that:

$$T(s_w, s_{w+1}, \pi) = P(o_{w+1}|o_w, \pi, \xi_w, \xi_{w+1})P(\xi_{w+1}|\xi_w)$$

The last term is determined by the arrival distribution of passengers and governs the transition probability of the ξ component of the state. The first term is determined by the defender action and determines the transition probability of the o component. We are

interested in computing the probability $P(o_{w+1}|o_w, \pi, \xi_w, \xi_{w+1})$. Since we decide how to screen passengers before observing the state transition, $\pi^w_{t,\kappa}$ be the probability that a single passenger of type κ is sent to team t in time window $w+1$. The probability that a single passenger of type k is sent to resource r in time window $w+1$ is:

$$\pi^w_{r,\kappa} = \sum_{t:r\in\mathcal{R}(t)} \pi^w_{t,\kappa}.$$

Let π^w_κ be the probability that a passenger arriving during time window $w+1$ is of type κ given ξ_w and ξ_{w+1} (assuming $\xi_{w+1} \geq \xi_w$):

$$\pi^{w+1}_\kappa = \frac{\xi_{\kappa,w+1} - \xi_{\kappa,w}}{\sum_{\kappa'}(\xi_{\kappa',w+1} - \xi_{\kappa',w})},$$

where $\xi_{\kappa,w+1} - \xi_{\kappa,w}$ is the number of passengers of type κ during time window $w+1$. Now the probability that a single passenger is sent to resource r during time window $w+1$ is:

$$\pi^w_r = \sum_{\kappa'} \pi_{r,\kappa'}\pi^w_{\kappa'}.$$

Given the state of overflow queue o_{w+1}, let n^{w+1}_o be the set of all possible number of people sent to each resource which could result in that overflow queue given o_w. The size

of the overflow queue in the next round is given by $o_{w+1,r} = \max(o_{w,r} - C_r + n_r, 0)$, so that n_o^{w+1} is given by:

$$n_o^{w+1} = \left\{ n : \begin{array}{l} n \in \mathbb{R}^{|\mathcal{R}|} \\ n_r \in \begin{cases} o_{w+1,r} - o_{w,r} + C_r & o_{w+1,r} > 0 \\ \{0 \dots o_{w,r} - C_r\} & o_{w+1,r} = 0 \end{cases} \end{array} \right\}$$

Let N_ξ^{w+1} be the total number of passengers arrived in time window $w+1$ so that

$$N_\xi^{w+1} = \sum_{\kappa'} (\xi_{\kappa',w+1} - \xi_{\kappa',w})$$

. Then for any $n \in n_o^{w+1}$, $p(n)$ follows a multinominal distribution with parameters N_ξ^{w+1} and $\{\pi_r^{w+1}\}_r$:

$$p(n) = N_\xi^{w+1}! \prod_r \frac{(\pi_r^{w+1})^{n_r}}{n_r!}.$$

Now, the transition probabilities of the overflow queue can be expressed by:

$$P(o_{w+1}|o_w, \pi, \xi_w, \xi_{w+1}) = \sum_{n \in n_o^{w+1}} p(n)$$

This defines the defender's problem as an MDP where due to the robust nature of the problem and the adversarial agent the transition matrix is also a function of the defender policy. The state space of this MDP is very large as it is the product space of total number of ways passengers may arrive Ξ and the number of possible passenger overflows O.

5.3 Proposed Solution Approach

Problem ($\mathscr{P}$) can become computationally expensive to solve for realistic size instances where the cardinality of Ξ is exponential in the number of time windows, see Example 8. We thus propose a solution approach that results in a tractable problem even when Ξ has exponentially many scenarios. In what follows, we describe our approach and main results.

5.3.1 Linear Decision Rule Approximation

Information Aggregation. In Problem ($\mathscr{P}$), the decision variables π^w are modeled as functions of the entire vector of past arrival realizations ξ^{w-1}. As a first step to obtain a tractable problem we propose to reduce information available to the screener and only allow his screening policy to adapt to the *aggregate number of screenees* that have arrived in past windows. Thus, we model the screening policy π^w for time window w as a function of the aggregate information $\zeta_{w-1} := \{\zeta_{w-1,\kappa}\}_{\kappa\in\mathcal{K}}$, where $\zeta_{w,\kappa} := \sum_{w'=1}^{w} \xi_{w',\kappa}$. The following proposition shows that this results in a conservative approximation to the optimal screening policy, since the restricted policy lies within the space of feasible policies.

Proposition 2. *Restricting the adaptive decision variables π^w and z^w for each time window $w \in \mathcal{W}$ to be functions of the aggregate information vector ζ_{w-1} provides a lower bound on the optimal objective value of Problem ($\mathscr{P}$).*

However in practice aggregating the information vector often does not result in any loss in optimality. We present here an example where one might expect a policy based on aggregate information to do poorly, and explain why the solution quality is robust to such a scenario.

Example 9. *Consider a situation with 2 teams each composed of a single resource r_1 and r_2, with capacities $c_1 = 5$ and $c_2 = 10$ and where r_2 and has a marginally worse capability of discovering the attacker than r_1 so that $E^m_{r_1} > E^m_{r_2} \quad \forall m$. The game takes place over 3 time windows $|W| = 3$ with one plane containing 15 people arriving. The fine for overflowing is $\phi < 0$ and the flight value is U so that $U^+ = 0$ and $U^- = -U$. Assume two different realizations for the first two periods – one realization, where in the first period there are 4 passengers, and there are 6 passengers in the second period; in the second realization, there are 5 passengers in each of the periods. Clearly, in the second realization, all passengers are checked by the better team T1 – i.e., even for the third period. However, in the first realization, it may be the case that if the passengers are assigned to T1, there is already 1 cost due to overflow and moreover, if T1 is assigned for screening additional passenger has to wait. Therefore, it actually may be optimal for the defender to use the allocation for T2 that is only marginally worse in case there is an overflow. However, this information is lost if only the aggregated number of passengers is used since both scenarios appear identical to the defender.*

Why this fails: *The scenario described looks at two different arrival distributions call them ξ and η so that:*

$$\xi = (5, 5, 5) \quad \eta = (4, 6, 5)$$

The optimal screening strategy for ξ is $\pi_1(\xi) = (1, 1, 1)$ and $\pi_2(\xi) = (0, 0, 0)$ so that there is no overflow in any time window.

For η the counter example assumes that it is optimal for time windows 1 and 2 to screen all passengers with resources 1, so that $\pi_1^1(\eta) = 1$ and $\pi_1^2(\eta) = 1$. We will now show that if this strategy is optimal for time windows 1 and 2, it must also be optimal for time window 3.

First we consider the potential amount of overflow which would be caused by any screening strategy $\pi_1^1(\eta), \pi_1^3(\eta), \pi_1^3(\eta)$. For brevity we ignore the screening strategy for resource 2 as there will never be any overflow on this resources in scenario η. We will also use π_1^1 as shorthand for $\pi_1^1(\eta)$.

In time window 1, no matter what our screening strategy π_1^1 is we would never get any overflow so that $o^1 = 0$.

In time window 2, using π_1^2 the overflow would be

$$\begin{aligned} o^2 &= \max(\textstyle\sum_r \pi_r^2 \eta_2 - C_r, 0) = \max(\pi_1^2 6 - 5, 0) \\ &= \begin{cases} 6\pi_1^2 - 5 & \textit{if } 1 \geq \pi_1^2 \geq \frac{5}{6} \\ 0 & \textit{else} \end{cases} \end{aligned}$$

In time window 3, using π_1^3 the overflow would be

$$\begin{aligned} o^2 &= \max(\textstyle\sum_r \pi_r^3 \eta_3 - C_r + o^2, 0) = \max(\pi_1^3 5 - 5 + \max(\pi_1^2 6 - 5, 0), 0) \\ &= \begin{cases} 6\pi_1^2 - 5 - (1 - \pi_1^3)5 & \textit{if } 1 \geq \pi_1^2 \geq \frac{5}{6} \;\&\; 6\pi_1^2 + 5\pi_1^3 \geq 10 \\ 0 & \textit{else} \end{cases} \end{aligned}$$

The total overflow would then be $O = o^1 + o^2 + o^3 \leq 12\pi_1^2 - 5\pi_1^3 - 15$

If we were only keeping track of the aggregate number of passengers, we would match the strategy of ξ *in the third time window so that* $\pi_1^3(\eta) = \pi_1^3(\xi) = 1$. *The counter example assumes that in the second time window* $\pi_1^2(\eta) = 1$ *is optimal. This gives us a total overflow* $O = 2$.
If the adversary were to attack in this time window we would receive a utility:

$$U3 = \phi O + (1 - E_1)\pi_1^3(-U) + (1 - E_2)\pi_2^3(-U) = 2\phi - (1 - E_1)U$$

The assumption here is that we may be able to increase this utility by decreasing the total overflow to $O = 1$ *using an alternate screening strategy. To decrease the overflow* o^3 *in the third round we need to use screening strategy* $\pi_1^3 = \frac{4}{5}$, $\pi_2^3 = \frac{1}{5}$. *This results in a new utility:*

$$U3' = \phi O + (1 - E_1)\pi_1^3(-U) + (1 - E_2)\pi_2^3(-U) = \phi - (1 - E_1)\frac{4U}{5} - (1 - E_2)\frac{U}{5}$$

The claim is that $U3 < U3'$. This implies that

$$\begin{aligned} 2\phi - (1-E_1)U &< \phi - (1-E_1)\tfrac{4U}{5} - (1-E_2)\tfrac{U}{5} \\ \phi &< (1-E_1)\tfrac{U}{5} - (1-E_2)\tfrac{U}{5} \\ \phi &< (E_2-E_1)\tfrac{U}{5} \end{aligned}$$

Assume that we then used this new strategy in the third time window so that $O = 1$. Consider now the utility in the second time window:

$$U2 = \phi O + (1-E_1)\pi_1^2(-U) + (1-E_2)\pi_2^2(-U) = \phi - (1-E_1)U$$

Since trading screening utility for less overflow was the better choice in the third time window then it must also be better in this time window. We can set $\pi_1^2 = \frac{5}{6}$ and achieve a new utility:

$$U2' = (1-E_1)\frac{-5U}{6} + (1-E_2)\frac{-U}{6}$$

Using the fact that $\phi < (E_2 - E_1)\frac{U}{5}$ and $E_1 > E_2$ we have that:

$$\begin{aligned} U2 = \phi - (1-E_1)U &< (1-E_1)\tfrac{U}{5} - (1-E_2)\tfrac{U}{5} - (1-E_1)U \\ &= -(1-E_1)\tfrac{4U}{5} - (1-E_2)\tfrac{U}{5} \\ &< (1-E_1)\tfrac{-5U}{6} + (1-E_2)\tfrac{-U}{6} = U2' \\ U2 &< U2' \end{aligned}$$

So that in fact $\pi_1^2 = \frac{5}{6}$ is optimal for time window 2. If we then return our attention to time window 3, we are now faced with no overflow. Therefore the screening strategy $p_1^3 = 1$ then results in no additional overflow and would be optimal, and the same as in the scenario where the passenger arrival is (5,5,5).

The key insight here is that this works well because we are performing robust optimization, and while the loss of information may change the utility of solutions in some of the realizations of ξ, if it does not change the worst case realization, then the objective value will not change.

However, even when restricting π to be functions of the aggregate arrival ζ, o^w and u_ρ are still functions of the full passenger arrival ξ^{w-1}. The overflow in time window w is a function of not only ξ_w but all $\xi_i \ \forall i \leq w$ since $o^w \geq \sum_{i=1}^{w}(\pi_{\kappa,t}^i \xi_{i,\kappa} - C_r)$. Additionally, u_ρ depends on ζ_w for all w, which is equivalent to knowing the actual passenger arrival ξ. Since restricting o^w and u_ρ to be functions of ζ^{w-1} would result in further loss of optimality we avoid it here.

Linear Decision Rule. In Problem ($\mathscr{P}$), the decision variables of the problem are arbitrary (bounded) functions of the uncertain parameter realizations. As a second step to obtain a tractable problem, we propose to restrict the space of feasible adaptive decisions to those that exhibit affine dependence on the data in the spirit of [Ben-Tal et al., 2004]. Thus, we let

$$
\begin{aligned}
\pi^w_{\kappa,t}(\xi^{w-1}) &= (\pi^w_{\kappa,t})^\top \zeta_{w-1} \quad \forall \kappa, t, w, \xi \\
z^w_{\kappa,m}(\xi^{w-1}) &= (z^w_{\kappa,m})^\top \zeta_{w-1} \quad \forall \kappa, m, w, \xi \\
o^w_r(\xi^{w-1}) &= (o^w_r)^\top \xi^{w-1} \quad \forall r, w, \xi \\
u_\rho(\xi) &= u_\rho^\top \xi \quad \forall \rho, \xi
\end{aligned}
$$

where the vectors $\pi^w_{\kappa,t}$, $z^w_{\kappa,m} \in \mathbb{R}^K$, $o^w_r \in \mathbb{R}^{K(w-1)}$ and $u_\rho \in \mathbb{R}^{KW}$ represent the new decision variables of the problem. Following the decision rule approximation, the number of decision variables of the problem is polynomial in the number of time windows, categories, resources, and teams. Also, it is independent of the number of scenarios. Since the linear functions lie in the space of all feasible functions the decision rule results in a conservative approximation. We denote the resulting conservative approximation by $(\mathscr{P}_1)$.

Proposition 3. *Problem $(\mathscr{P}_1)$ provides a lower bound on the optimal objective value of problem $(\mathscr{P})$.*

This is trivially shown by noting that the space of possible policies of problem $(\mathscr{P}_1)$ is a subset of the policy space of $(\mathscr{P})$.

Proposition 4. *If resource overflow is not permitted (i.e., $o^w_r(\xi^{w-1}) = 0 \; \forall r \in \mathcal{R}$, $w \in \mathcal{W}$, and $\xi \in \Xi$), then there exists an optimal solution to the adaptive robust optimization problem $(\mathscr{P})$ such that: π, z, and u are all constant in ξ.*

Proof. Let

$$
\overline{\xi} := \underset{\xi \in \Xi}{\arg\min} \left\{ \sum_{\rho \in \mathcal{P}} P_\rho u_\rho(\xi) \right\}.
$$

In the definition of $\overline{\xi}$, we have assumed that the maximization problem admits a unique solution. This is without loss of generality. If this is not the case, we can choose the solution that comes first using lexicographic ordering.

Define $(\tilde{\pi}, \tilde{z}, \tilde{u})$ through:

$$\begin{aligned}
\tilde{\pi}^{w'}_{\kappa,t}(\xi^{w'-1}) &:= \pi^{w}_{\kappa,t}(\overline{\xi}^{w-1}) \\
\tilde{z}^{w'}_{\kappa,m}(\xi^{w'-1}) &:= \sum_{t\in\mathcal{T}} E_{t,m}\tilde{\pi}^{w'}_{\kappa,t}(\xi^{w'-1}) \\
\tilde{u}_{\rho}(\xi) &:= \min_{w',m} \min_{\kappa:\xi_{w',\kappa}>0} \tilde{z}^{w'}_{\kappa,m}(\xi^{w'-1})U^{+}_{\kappa} + (1-\tilde{z}^{w'}_{\kappa,m}(\xi^{w'-1}))U^{-}_{\kappa}
\end{aligned}$$

where we use use the convention that an infeasible minimization problem has optimal objective equal to ∞.

$\forall \xi \in \Xi$, $w' \in \mathcal{W}$

The policy $(\theta, \tilde{\pi}, \tilde{z}, \tilde{u})$ trivially satisfies the second and third constraints in the problem and the requirement that $\tilde{\pi} \in \Pi_{\text{o}}$. It remains to verify that it satisfies the first set of constraints. Define

$$\begin{aligned}
\theta(\xi) &:= \sum_{\rho\in\mathcal{P}} P_{\rho}u_{\rho}(\xi) \\
\tilde{\theta}(\xi) &:= \sum_{\rho\in\mathcal{P}} P_{\rho}\tilde{u}_{\rho}(\xi)
\end{aligned}$$

The first constraint of problem ($\mathscr{P}$) can be equivalently expressed as $\theta \leq \theta(\xi)\ \forall \xi \in \Xi$. Optimality of $(\theta, \tilde{\pi}, \tilde{o}, \tilde{z}, \tilde{u})$ requires that $\theta \leq \tilde{\theta}(\xi) \quad \forall \xi$. If we can guarantee $\tilde{\theta}(\xi) \geq \theta(\xi) \quad \forall \xi$ then it trivially follows that $\theta \leq \tilde{\theta}(\xi)$.

By definition of $\bar{\xi}$ we know that $\theta(\bar{\xi}) \leq \theta(\xi) \quad \forall \xi$ and therefore that $\theta \leq \theta(\bar{\xi})$. By definition of $\tilde{u}_\rho(\xi)$ we know that $\tilde{u}_\rho(\xi) = u_\rho(\bar{\xi})$ so that $\tilde{\theta}(\xi) = \theta(\bar{\xi})$. Therefore $\theta \leq \tilde{\theta}(\xi)$ and policy $(\theta, \tilde{\pi}, \tilde{z}, \tilde{u})$ is optimal. □

5.3.2 Robust Counterpart

Problem $(\mathscr{P}_1)$ exhibits only a moderate number of decision variables but still a very large number of constraints. In what follows, we propose to mitigate the number of constraints by using techniques inspired from modern robust optimization [Ben-Tal et al., 2004]. The key observation is that under the linear decision rule approximation, all constraints in the problem (except from (5.4)) are linear in ξ, thus being expressible in the form $a(x)^\top \xi \leq 0$ $\forall \xi \in \Xi$, for some linear function a that maps the collection of all decision rule coefficients (denoted by x) to coefficients of ξ. The following proposition enables us to reformulate these constraints in a compact fashion.

Proposition 5. *For any $y \in \mathbb{R}^k$, define:*

(i) $y^\top \xi \leq 0 \quad \forall \xi \in \Xi$

(ii) $\exists \lambda \in \mathbb{R}^\ell$ *with* $\lambda \geq 0$, $V^\top \lambda \geq y$, *and* $h^\top \lambda \leq 0$.

Then $ii)$ implies $i)$.

Proof. Following a standard duality argument, we have

$$\begin{aligned}
& y^\top \xi \leq 0 \quad \forall \xi : \xi \in \mathbb{N}_+,\ V\xi \leq h \\
\Leftarrow & y^\top \xi \leq 0 \quad \forall \xi : \xi \geq 0,\ V\xi \leq h \\
\Leftrightarrow & \{\max\ y^\top \xi\ :\ V\xi \leq h,\ \xi \geq 0\} \leq 0 \\
\Leftrightarrow & \{\min\ h^\top \lambda\ :\ V^\top \lambda \geq y,\ \lambda \geq 0\} \leq 0 \\
\Leftrightarrow & \exists \lambda \in \mathbb{R}^\ell\ :\ h^\top \lambda \leq 0,\ V^\top \lambda \geq y,\ \lambda \geq 0,
\end{aligned}$$

which concludes the proof. □

To represent these constraints efficiently, we apply the above result to each constraint in Problem $(\mathscr{P}_{\rm l})$ (except from (5.4)) and denote the resulting problem by $(\mathscr{P}_{\rm l-rc})$. For general uncertainty sets, we obtain a conservative approximation to Problem $(\mathscr{P}_{\rm l})$. The following Proposition establishes that with uncertainty set $\Xi_{\rm AS}$, as defined as in Example 6, the reformulation of these constraints is exact.

Proposition 6. *Suppose that we have the uncertainty set $\Xi_{\rm AS}$ as defined in Example 6. Then, statements $i)$ and $ii)$ in Proposition 5 are equivalent and Problems $(\mathscr{P}_{\rm l})$ and $(\mathscr{P}_{\rm l-rc})$ are equivalent.*

Proof. We first show that statements $i)$ and $ii)$ are equivalent for $\Xi = \Xi_{\rm AS}$. From Proposition 4, we have that $ii)$ implies $i)$. We now show that it also holds that $i)$ implies $ii)$. Define the LP relaxation of the uncertainty set $\Xi_{\rm AS}$ as

$$\mathcal{LP}(\Xi_{\rm AS}) := \left\{ \xi\ :\ \xi \geq 0,\ \sum_{w \in \Delta_\kappa} \xi_{w,\kappa} = N_\kappa\ \ \forall \kappa \right\},$$

and let $\mathcal{CH}(\Xi_{\text{AS}})$ denote its convex hull. It suffices to show that $\mathcal{LP}(\Xi_{\text{AS}})$ and $\mathcal{CH}(\Xi_{\text{AS}})$ coincide, or equivalently that the extreme points of $\mathcal{LP}(\Xi_{\text{AS}})$ are integer. In what follows, we make the assumption that $N_\kappa > 0$ for all κ. This assumption is non-restrictive and can always be guaranteed by eliminating any categories κ for which $N_\kappa = 0$ from the problem (since no people arrive in this category, there is no need to screen them).

We note that $\mathcal{LP}(\Xi_{\text{AS}})$ is a non-empty standard form polyhedron and thus has at least one extreme point. We now argue that at any basic feasible solution of $\mathcal{LP}(\Xi_{\text{AS}})$, there must be one and only one element of $\{\xi_{w,\kappa}\}_{w\in\Delta_\kappa}$ that is distinct from zero for each κ which will conclude the proof. Let $K := |\mathcal{K}|$. The LP relaxation of the uncertainty set involves WK uncertain parameters and K equality constraints. At any basic solution, there must be at least $WK - K$ uncertain parameters at zero level (at most K non-zero uncertain parameters) and the equality constraints must hold. Since $N_\kappa > 0$ by assumption, there must be at least one of $\{\xi_{w,\kappa}\}_{w\in\Delta_\kappa}$ that is non-zero for each κ at any basic solution. Suppose that there is a basic solution at which more than one of $\{\xi_{w,\kappa}\}_{w\in\Delta_\kappa}$ is non-zero for some κ. Then the total number of non-zero parameters is strictly greater than K, a contradiction. This concludes the first part of the proof.

We now show that statements $i)$ and $ii)$ are equivalent for $\Xi = \Xi_{\text{AS}} \cap \{\xi_{w',\kappa'} > 0\} = \Xi_{\text{AS}} \cap \{\xi_{w',\kappa'} \geq 1\}$, $w' \in \mathcal{W}$, $\kappa' \in \mathcal{K}$. The LP relaxation of this uncertainty set is given by

$$\mathcal{LP}(\Xi_{\text{AS}} \cap \{\xi_{w',\kappa'} \geq 1\}) = \left\{ \xi : \xi \geq 0,\ \sum_{w\in\Delta_\kappa} \xi_{w,\kappa} = N_\kappa\ \forall \kappa,\ \xi_{w',\kappa'} \geq 1 \right\}.$$

Note that this set has at least one extreme point since it is non-empty and does not contain a line. We now show that at any extreme point of $\mathcal{LP}(\Xi_{\text{AS}} \cap \{\xi_{w',\kappa'} \geq 1\})$, there must be one and only one element of $\{\xi_{w,\kappa}\}_{w\in\Delta_\kappa}$ that is distinct from zero for each κ which

will conclude the proof. First, it follows from the assumption $N_\kappa > 0$ that at least one of $\{\xi_{w,\kappa}\}_{w\in\Delta_\kappa}$ must be non-zero at an extreme point. We now proceed by contradiction to show that no more than one component of $\{\xi_{w,\kappa}\}$ can be non-zero at an extreme point. Suppose that ξ is an extreme point such that for some $\overline{\kappa}$, it holds that $\xi_{w_1,\overline{\kappa}} > 0$ and $\xi_{w_2,\overline{\kappa}} > 0$ for $w_1, w_2 \in \Delta_{\overline{\kappa}}$ with $w_1 \neq w_2$. Define the points ξ^1 and ξ^2 through

$$\xi^1_{w,\kappa} := \begin{cases} \xi_{w_1,\overline{\kappa}} + \xi_{w_2,\overline{\kappa}} & \text{if } \kappa = \overline{\kappa} \text{ and } w = w_1 \\ 0 & \text{if } \kappa = \overline{\kappa} \text{ and } w = w_2 \\ \xi_{w,\kappa} & \text{else,} \end{cases}$$

and

$$\xi^2_{w,\kappa} := \begin{cases} 0 & \text{if } \kappa = \overline{\kappa} \text{ and } w = w_1 \\ \xi_{w_1,\overline{\kappa}} + \xi_{w_2,\overline{\kappa}} & \text{if } \kappa = \overline{\kappa} \text{ and } w = w_2 \\ \xi_{w,\kappa} & \text{else.} \end{cases}$$

It can be readily verified that ξ^1 and ξ^2 are both elements of $\mathcal{LP}(\Xi_{\mathrm{AS}} \cap \{\xi_{w',\kappa'} \geq 1\})$ and that

$$\frac{\xi_{w_1,\overline{\kappa}}}{\xi_{w_1,\overline{\kappa}} + \xi_{w_2,\overline{\kappa}}}\xi^1 + \left(1 - \frac{\xi_{w_1,\overline{\kappa}}}{\xi_{w_1,\overline{\kappa}} + \xi_{w_2,\overline{\kappa}}}\right)\xi^2 = \xi,$$

so that ξ is expressible as a strict convex combination of two elements of $\mathcal{LP}(\Xi_{\mathrm{AS}} \cap \{\xi_{\kappa',w'} \geq 1\})$, a contradiction. □

5.3.3 Constraint Randomization

Problem $(\mathscr{P}_{\mathrm{l-rc}})$ still involves constraint (5.4) enforced over a set Ξ of potentially very large cardinality. We obtain a tractable approximation to $(\mathscr{P}_{\mathrm{l-rc}})$ by replacing Ξ with

subsets $\Xi^N \subset \Xi$ of cardinality N. We denote the resulting problem by $(\mathscr{P}^N_{\mathrm{l-rc}})$. The following theorem shows that a randomly sampled subset Ξ^N of moderate cardinality N will lead a good approximation.

Theorem 7 ([Campi and Garatti, 2008]). *Suppose that $(\mathscr{P}_{\mathrm{l-rc}})$ is feasible and accommodates n decision variables. For a prespecified violation probability $\epsilon \in (0,1)$ and confidence $\beta \in (0,1)$, define*

$$N(\epsilon,\beta) := \min\left\{ N \in \mathbb{N} \ : \ \sum_{i=0}^{n-1} \binom{i}{N} \epsilon^i (1-\epsilon)^{N-i} \leq \beta \right\}$$

Then, the probability mass of all $\xi \in \Xi$ whose associated constraints are violated by an optimal solution of $(\mathscr{P}_{\mathrm{l-rc}})$, for $N \geq N(\epsilon,\beta)$, does not exceed ϵ with confidence $1-\beta$.

The parameter ϵ describes the probability that an optimal solution to $(\mathscr{P}_{\mathrm{l-rc}})$ violates the overflow constraint. A violation of the overflow constraint implies that the overflows are calculated incorrectly for some samples so that the part of the objective associated with overflow is calculated incorrectly. The theorem states that such miscalculations are rare. Moreover, the size of the resulting sampled problem is polynomial in the number of time windows, categories, resources, and teams, see [Vayanos et al., 2012]. In order to solve the resulting problem more efficiently we employ a cutting plane method, in the spirit of [Fischetti and Monaci, 2012].

5.3.4 Full Robust Threat Screening Game Formulation

We use the uncertainty set Ξ_{AS}. Substituting in the linear decision rules for π and o the constraints read as follows:

$$
\begin{aligned}
\text{maximize} \quad & \theta \\
\text{subject to} \quad & \theta \leq \sum_{\rho \in \mathcal{P}} P_\rho u_\rho^\top \xi - \sum_{w \in \mathcal{W}} \sum_{r \in \mathcal{R}} F_r (o_r^w)^\top \xi^{w-1} && \forall \xi \\
& u_\rho^\top \xi \leq \sum_{t \in \mathcal{T}} E_{t,m} (\pi_{\kappa,t}^w)^\top \zeta_{w-1} U_\kappa^+ + (1 - \sum_{t \in \mathcal{T}} E_{t,m} (\pi_{\kappa,t}^w)^\top \zeta_{w-1}) U_\kappa^- && \forall \xi_{w,\kappa} > 0 \\
& \sum_{t : r \in \mathcal{R}(t)} \sum_{\kappa \in \mathcal{K}} (\pi_{\kappa,t}^w)^\top \zeta_{w-1} \xi_{w,\kappa} \leq C_r - (o_r^w)^\top \xi^{w-1} + (o_r^{w+1})^\top \xi^w && \forall r, w, \xi \\
& (\pi_{\kappa,t}^w)^\top \zeta_{w-1} \geq 0 && \forall \kappa, t, w, \xi \\
& (\pi_{\kappa,t}^w)^\top \zeta_{w-1} \leq 1 && \forall \kappa, t, w, \xi \\
& \sum_{t \in \mathcal{T}} (\pi_{\kappa,t}^w)^\top \zeta_{w-1} = 1 && \forall \kappa, w, \xi \\
& (o_r^w)^\top \xi^{w-1} \geq 0 && \forall r, w, \xi
\end{aligned}
$$

We show an example of how to take the robust counterpart of the last constraint $(o_r^w)^\top \xi^{w-1} \geq 0 \quad \forall \xi$. The robust reformulation of constraints (1) (4) (5) are done in the same way. By proposition 5, we know that $\mathcal{CH}(\Xi_{\text{AS}}) = \mathcal{LP}(\Xi_{\text{AS}})$ so that we may write:

$$
\begin{aligned}
& (o_r^w)^\top \xi^{w-1} \geq 0 && \forall \xi \in \mathcal{CH}(\Xi_{\text{AS}}) \\
\Leftrightarrow & \min (o_r^w)^\top \xi^{w-1} && \\
& \sum_{w \in \Delta_\kappa} \xi_{w,\kappa} = N_\kappa && \forall \kappa \\
& \xi \geq 0 && \\
\Leftrightarrow & \max (y_r^w)^\top N && \\
& (y_r^w)_\kappa \leq (o_r^w)_{w',\kappa} && \forall \kappa, w' \in \Delta_\kappa, w' < w \\
& (y_r^w)_\kappa \leq 0 && \forall \kappa \\
& y_r^w \in \mathbb{R}^\kappa && \\
\Leftrightarrow & (y_r^w)^\top N \geq 0 && \\
& (y_r^w)_\kappa \leq (o_r^w)_{w',\kappa} && \forall \kappa, w' \in \Delta_{\kappa'}, w' < w \\
& (y_r^w)_\kappa \leq 0 && \forall \kappa
\end{aligned}
$$

We reformulate constraint (6) by matching coefficients. We have that $\zeta_{w,\kappa} = \sum_{i=1}^{w} \xi_{i,\kappa}$ and $\zeta_0 = 1$ so the constraint is equivalently expressed

$$\sum_{t \in \mathcal{T}} (\pi_{\kappa,t}^w)_\kappa \zeta_{w-1,\kappa} + (\pi_{\kappa,t}^w)_0 = 1 \quad \forall \kappa, w, \zeta$$

This can only be satisfied for all ζ if $\sum_{t \in \mathcal{T}} (\pi_{\kappa,t}^w)_\kappa = 0$ and $(\pi_{\kappa,t}^w)_0 = 1$ and so we have that:

$$\sum_{t\in\mathcal{T}}(\pi^w_{\kappa,t})^\top\zeta_{w-1}=1 \quad \forall\kappa,w,\zeta$$
$$\Leftrightarrow\sum_{t\in\mathcal{T}}(\pi^w_{\kappa,t})_\kappa=0,\quad (\pi^w_{\kappa,t})_0=1 \quad \forall\kappa,w$$

For constraint (2) the uncertainty set has the additional constraint $\xi_{w,\kappa}>0$. We show an example of how to take the robust counterpart of such a constraint in the form $(a^w_{\kappa,m})^\top\xi\geq 0 \quad \forall\xi:\xi_{w,\kappa}>0$ where for constraint (2):

$$(a^w_{\kappa,m})_{w',\kappa'}=U^+_\kappa\sum_{t\in\mathcal{T}}E_{t,m}(\pi^w_{\kappa,t})_{\kappa'}+U^-_\kappa(1-\sum_{t\in\mathcal{T}}E_{t,m}(\pi^w_{\kappa,t})_{\kappa'})-(u_\rho)_{w',\kappa'}$$

By proposition 5, we know that $\mathcal{CH}(\Xi_{\text{AS}}\cap\xi_{w,\kappa}\geq 1)=\mathcal{LP}(\Xi_{\text{AS}}\cap\xi_{w,\kappa}\geq 1)$ so that we may write:

$$\begin{aligned}
&(a^w_{\kappa,m})^\top\xi\geq 0 && \forall\xi\in\mathcal{CH}(\Xi_{\text{AS}}\cap\xi_{w,\kappa}\geq 1)\\
\Leftrightarrow&\min(a^w_{\kappa,m})^\top\xi\\
&\sum_{w'\in\Delta_{\kappa'}}\xi_{w',\kappa'}=N_{\kappa'} && \forall\kappa'\\
&\xi_{w,\kappa}\geq 1\\
&\xi_{w',\kappa'}\geq 0 && \forall,\kappa',w'\neq w
\end{aligned}$$

$$\begin{aligned}
\Leftrightarrow&\max(v^w_{\kappa,m})^\top N+q^w_{\kappa,m}\\
&(v^w_{\kappa,m})_{\kappa'}\leq(a^w_{\kappa,m})_{w',\kappa'} && \forall\kappa',w'\in\Delta_{\kappa'}\\
&(v^w_{\kappa,m})_\kappa+q^w_{\kappa,m}\leq(a^w_{\kappa,m})_{w,\kappa}\\
&q^w_{\kappa,m}\geq 0\\
&v^w_{\kappa,m}\in\mathbb{R}^\kappa,\ \ q^w_{\kappa,m}\in\mathbb{R}
\end{aligned}$$

$$
\begin{aligned}
\Leftrightarrow & (v^w_{\kappa,m})^\top N + q^w_{\kappa,m} \\
& (v^w_{\kappa,m})_{\kappa'} \leq (a^w_{\kappa,m})_{w',\kappa'} && \forall \kappa', w' \in \Delta_{\kappa'} \\
& (v^w_{\kappa,m})_{\kappa} + q^w_{\kappa,m} \leq (a^w_{\kappa,m})_{w,\kappa} \\
& q^w_{\kappa,m} \geq 0
\end{aligned}
$$

The final robust multistage optimization problem can be expressed as:

$$
\begin{aligned}
\text{maximize} \quad & \theta \\
\text{subject to} \quad & \theta \leq \sum_{\rho \in \mathcal{P}} P_\rho(u_\rho)_0 - \sum_{w \in \mathcal{W}} \sum_{r \in \mathcal{R}} F_r(o_r^w)_0 - s^\top N \\
& \sum_{\rho \in \mathcal{P}} P_\rho(u_\rho) - \sum_{w \in \mathcal{W}} \sum_{r \in \mathcal{R}} F_r(o_r^w) - s \geq 0 \\
& (U_\kappa^+ - U_\kappa^-) \sum_t E_{t,m}(\pi_{\kappa,t}^w)_0 + U_\kappa^- \leq (u_\rho)_0 - (v_{\kappa,m}^w)^\top N + q_{\kappa,m}^w && \forall \kappa, w, m \\
& (U_\kappa^+ - U_\kappa^-) \sum_t E_{t,m}(\pi_{\kappa,t}^w)_{\kappa'} \leq (u_\rho)_{w',\kappa'} - (v_{\kappa,m}^w)_{\kappa'}^\top && \forall \kappa, w, m, \kappa' \\
& && w' \neq w, w' \in \Delta_{\kappa'} \\
& (U_\kappa^+ - U_\kappa^-) \sum_t E_{t,m}(\pi_{\kappa,t}^w)_\kappa \leq (u_\rho)_{w,\kappa} - (v_{\kappa,m}^w)_\kappa^\top - q_{\kappa,m}^w && \forall \kappa, w, m, \kappa' \\
& && w' = w, w' \in \Delta_{\kappa'} \\
& \sum_{t : r \in \mathcal{R}(t)} \sum_{\kappa \in \mathcal{K}} \pi_{\kappa,t}^w(\xi^{w-1}) \xi_{w,\kappa} \leq C_r - o_r^w(\xi^{w-1}) + o_r^{w+1}(\xi^w) && \forall r, w, \xi \\
& q_{\kappa,m}^w \geq 0 && \forall \kappa, w, m \\
& \sum_{t \in \mathcal{T}} (\pi_{\kappa,t}^w)_{\kappa'}^\top = 0 && \forall w, \kappa, t, \kappa' \\
& \sum_{t \in \mathcal{T}} (\pi_{\kappa,t}^w)_0^\top = 1 && \forall w, \kappa, t \\
& (x_{\kappa,t}^w)^\top N + (\pi_{\kappa,t}^w)_0 \geq 0 && \forall w, \kappa, t \\
& (x_{\kappa,t}^w)_{\kappa'} \leq (\pi_{\kappa,t}^w)_{\kappa'} && \forall w, \kappa, t, \kappa' \\
& -(d_{\kappa,t}^w)^\top N + (\pi_{\kappa,t}^w)_0 \leq 1 && \forall w, \kappa, t \\
& (d_{\kappa,t}^w)_{\kappa'} \leq -(\pi_{\kappa,t}^w)_{\kappa'} && \forall w, \kappa, t, \kappa' \\
& (d_{\kappa,t}^w)_{\kappa'} \leq 0 && \forall w, \kappa, t, \kappa' \\
& (y_r^{w+1})^\top N \geq 0 && \forall w, r \\
& (y_r^{w+1})_{\kappa'} \leq (o_r^{w+1})_{w',\kappa'} && \forall w, r, \kappa' \\
& w' \in \Delta_{\kappa'}, w' \leq w \\
& (y_r^{w+1})_\kappa \leq 0 && \forall w, r, \kappa
\end{aligned}
$$

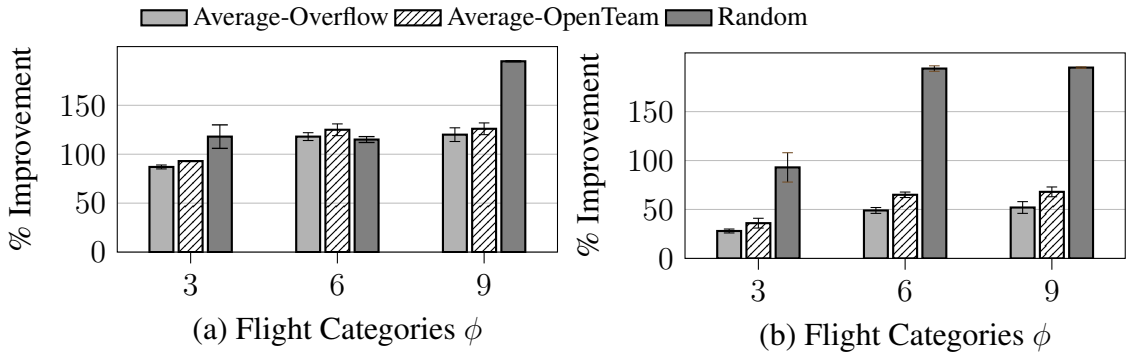


Figure 5.3: Utility improvement over averaged sample and random uniform in (a) worst case and (b) average case.

5.4 Evaluation

We evaluate our framework on airport passenger screening problems with uncertainty set Ξ_{AS}.

5.4.1 Solution Quality

The optimal objective of our solution gives us the performance on the training set of samples we use. We evaluate the solution quality *out of sample* (both on average and in the worst case) by generating a large test set. We also use the test set to compute an experimental *violation probability*. We assume that the arrival of passengers is normally distributed in the range Δ_κ. Each data point is averaged over 30 trials, each with randomized parameter settings, with error bars giving the 90% confidence intervals. For each of these trials we generate 10,000 samples from the distribution of passenger arrivals, and evaluate the computed strategy on each sample so that each data point corresponds to 300,000 evaluations.

Uncertainty model vs Averaged Model. We compare our solution method to the TSG model for problems with increasing numbers of flights with $W = 10$. The TSG model optimizes against only the average ξ, so there will be many scenarios where the strategy becomes infeasible. We consider two heuristics to adjust an infeasible strategy: *(1) Overflow Heuristic:* add excess passengers to the existing overflow queue, or *(2) Open-Team Heuristic:* send excess passengers to any team with available capacity. Figure 5.3 summarizes our results. Against both heuristics, we outperform the TSG in worst case (average) by more than 100% (50%). The average violation probability was $98 \pm 2\%$ for the averaged sample solutions and $0.5 \pm 0.02\%$ for the solution to $(\mathscr{P}^N_{\mathrm{l-rc}})$.

Uncertainty Model vs Uniform Random. We compare to a baseline where passengers are assigned to teams uniformly at random. Figure 5.3 shows our results. In both the average and worst cases, the solution quality of random screening can be arbitrarily bad– we reach around 200% improvement.

Full Stochastic Program. We also compare the quality of the solution of $(\mathscr{P}^N_{\mathrm{l-rc}})$ to that of the optimal solution to the full stochastic program associated with $(\mathscr{P})$. Because the full program is exponential in the number of categories, we can only solve for very small problem instances. We fix the number of time windows, with an arrival period of 2 time windows for any flight, and show runtime and solution quality for a small range of categories. The results are shown in Table 5.1 where near-optimal performance is exhibited.

(ϕ, ρ)	% Diff Solution	Solve Time (ms)		Wall Time (ms)	
		$(\mathscr{P}^N_{\mathrm{l-rc}})$	$(\mathscr{P})$	$(\mathscr{P}^N_{\mathrm{l-rc}})$	$(\mathscr{P})$
(1,2)	1.3(0.1)	7.4(0.1)	13.1(0.4)	22.8(0.2)	79(9)
(1,3)	0.29(0.1)	40(1)	320(10)	110(10)	2500(230)
(1,4)	-	110(40)	-	640(10)	-
(2,1)	1.1(0.03)	2.5(0.07)	7(0.2)	10.9(0.4)	44.9(0.6)
(2,2)	0.8(0.01)	87(1)	2130(90)	260(10)	70500(90)
(2,3)	-	340(50)	-	2700(100)	-

Table 5.1: Comparing the $(\mathscr{P}^N_{\mathrm{l-rc}})$ to full stochastic program $(\mathscr{P})$. Blank entries correspond to instances where the full stochastic program could not be held in memory.

5.4.2 Scalability

Figure 5.4 shows total solve and wall times for problems with increasing number of flight categories. We are able to efficiently solve for a very large number of flight categories, with polynomial growth with respect to flight categories.

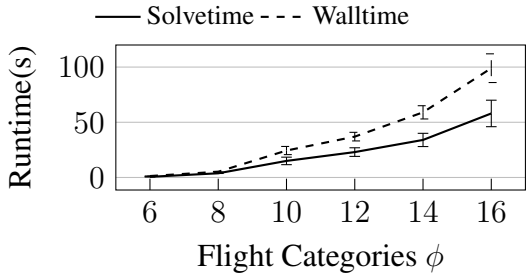


Figure 5.4: Solve & Wall time with increasing number of flights.

Table 5.2 summarizes our findings, showing the experimental violation probability as well as the number of decision variables needed for each problem. We see that even for

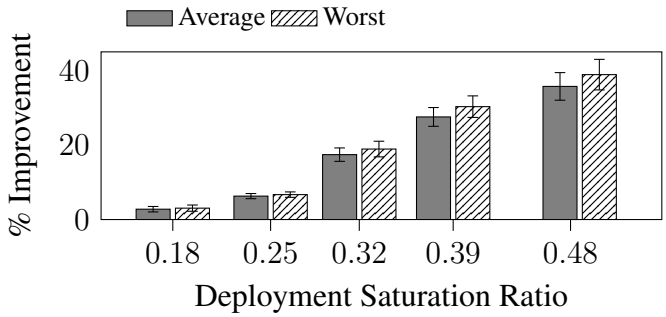


Figure 5.5: Utility improvement using adaptive decision rules.

very large problems, where the cardinality of Ξ_{AS} is very large, the computed strategies have very low violation probability.

ϕ	Violation Probability (%)	Decision Variables ($\times 10^2$)
10	0.27 ± 0.03	40 ± 0.5
12	0.18 ± 0.03	54 ± 0.5
14	0.14 ± 0.02	70 ± 0.6
16	0.19 ± 0.04	100 ± 2

Table 5.2: Experimental violation probability with increasing problem size.

Deployment to Saturation Ratio. In Figure 5.5 we explore the space in which the decision problem becomes difficult by comparing the linear decision rule to a constant decision rule, where we make the same decisions regardless of the past arrival of passengers. It is a known phenomenon in security games, that the problem difficulty increases as the deployment to saturation ratio (ratio of defender resources to targets) approaches 0.5 [Jain et al., 2012b].

We measure the ratio by comparing the number of passengers to the capacity, for a single flight, so that the maximum number of passengers which can be screened in any time window is clearly defined. Figure 5.5 shows that as the problem difficulty increases, the gap in solution quality becomes large and the adaptive screening greatly outperforms the constant strategy.

5.5 Chapter Summary

This chapter makes several contributions in the area of tactical planning for security games by extending an important class of games to model heterogeneous resources, and enriching on of the few existing game models with teams of resources. In this chapter, I address a significant limitation in the area of Threat Screening Games, where the previous unrealistic assumption of complete certainty in screenee arrival times renders its solution unusable in real-world settings, by proposing a scalable framework that provides good solution quality and works for generalized models of uncertainty. This framework can additionally be applied to other problems where existing techniques such as column generation may fail to scale due to the more complex nature of modeling heterogenous resources. As such this addresses an important limitation of

Part IV

Operational Planning

Chapter 6

Operational Planning in Security Games

In the last chapter of this thesis I formalize the operational planning problem for security and game theory and how it relates to the usability of game theoretic models. I discuss why usability and consequently operationalization is important and when it becomes necessary to take operational planning into consideration.

Previous chapters of this thesis discuss many ways in which security game models can be made more expressive, with the goal of making them more realistic and more applicable to solve real world problems. In fact a large portion of the past work in this area is dedicated to developing and extending the security game frame work to better model the real world. As there has been much success in this goal, security agencies have begun adopting these more sophisticated game-theoretic strategies making *ease of practical use and implementation* a key concern. These game models have found purchase in many real world security systems and organizations such as the Los Angeles International airport [Pita et al., 2008b], the United States Coast Guard [Shieh et al., 2012b], the Federal Air Marshal Service [Tsai et al., 2009], police departments in Los Angeles [Yin et al., 2012], Mumbai [Jain et al., 2010a] and Singapore [Brown et al., 2014], the Transportation Security Administration [Brown et al., 2016] and well as many non governmental organizations such as the World Wildlife Fund, the Wildlife Conservation Society, and Panthera [Fang et al., 2017, Ford et al., 2014b, Nguyen et al., 2016a] from preliminary field tests

being conducted to actual deployed applications. However, as the use of these models continues to grow, it becomes ever more pertinent to consider the *operationalization* of the plans generated from these models, i.e. how the mixed strategy solutions will be actually be implement and deployed by the security personnel.

In this next section I will formalize the problem of usability of mixed strategy, discuss the tradeoffs in usability and utility of solutions.

6.1 Usability in the Real World

As mentioned in the previous section, many game theoretic models have been deployed and field tested in real world systems by real world security agencies. Solutions to these games are mixed strategies, being randomizations over a set of pure strategies. In practice, each pure strategy can be viewed as a separate *security protocol*. Thus, mixed strategies with large support sets can be problematic to operationalize as they require security agents to be familiar with a large variety of protocols to execute them all properly. These types of complex tasks increase the cognitive load in individuals [Hogg, 2007] increasing the likelihood that mistakes are made [Paas, 1992, Cooper and Sweller, 1987] and making the system vulnerable to exploitation. While such usability concerns have always been present in deployed security games, these have often been addressed in an ad-hoc fashion, and not explicitly discussed in the literature.

Interest in the topic of usability for security games, in particular the computation of small support mixed strategies developed due to interactions and feedback from these agencies when actually attempting to deploy the strategies produced by these models. A prime example of this was work that was done previously for the United States Coast

Guard, a branch of the United States Armed Forces, who, among many other duties, are tasked with protecting ports all around the United States. Game theoretic models were developed in order to compute patrol routes the coast guard to use when deciding what areas of the port should be monitored [Fang et al., 2013b]. However the mixed strategy solutions so these had support over a very large number of pure strategies, in the thousands. The coast guard patrols operate normally by breaking down the patrols they are going to conduct into a sequence of maneuvers to ensure proper execution of the patrols. Requiring then that the coast guard personnel be familiar with thousands of sequences of maneuvers, in the thousands of pure strategies, is not reasonable in practice. In fact when these strategies were first presented to the coast guard, they were rejected for these exact reasons, where the coast guard responded that they would only be willing to use four different patrols [Fang private communication 2018].

Thus there is a real world demand for small numbers of effective strategies, and if security game models are to be adopted by security agencies, they need to be able to address this requirement in a principled way. In the next section I describe how to formally address this usability concern for security games.

6.2 The Price of Usability and Operationalizable Mixed Strategies

Security games deal with the challenge of allocating scarce security resources among heterogeneous targets to avert a possible attack. Optimal solutions to security games can involve randomizing over large numbers of pure strategies, each corresponding to a different security protocol. Thus, while they are by far preferable to deterministic strategies

from an efficiency perspective, they are difficult to operationalize, requiring the security personnel to be familiar with numerous protocols in order to execute them.

Example 10 (Usability in a Real-World Security Game)**.** *Consider the real world security game presented in the original SORT problem [Mc Carthy et al., 2016], where the defender must build a team of resources, to be deployed on patrols to protect an endangered forest area. Within this area there is a set of $N+1$ targets indexed in the set $\{0,\ldots,N\}$ to be protected. Let the first target, t_0, have a value $v_0 := 10 - \frac{1}{N}$ and the remaining targets, t_i, have value $v_i := 1$, $i = 1,\ldots,N$. In a pure strategy, the defender may either choose to concentrate their resources to statically protect a single target (no patrolling) or may conduct a patrol, which splits their time equally between a set of m targets so that each target is covered during $\frac{1}{m} - \frac{m}{100}$ of the time (here $\frac{m}{100}$ corresponds to the amount of coverage lost due to travel between m targets.*

The unique optimal defender strategy is a mixture over $N+1$ pure strategies covering each target statically such that t_0 is covered statically with probability 0.9 and each t_i, $i = 1,\ldots,N$, is covered statically with probability $\frac{0.1}{N}$.

Suppose now that to facilitate the operationalization of his defense strategy, the defender restricts themselves to using only two strategies: one which covers t_0 statically and one which patrols between the remaining N targets, playing the first with 0.9 probability and the second with probability 0.1. This strategy, which has only two support points achieves an ϵ approximation to that of the best mixed strategy. At the same time, this strategy is far easier to operationalize.

The operational planning problem explicitly formulates planning for usability in security games and motivate our definition to limit the cognitive load placed on security personel, and refer to such strategies as being *operationalizable*.

Operationalizable Security Games Motivated by our discussions with practitioners in the security domain, we propose a model for usability in security games which we refer to as *Operationalizable Security Games* that admits solutions whose mixed strategy support cardinality is a design parameter selected by the defender; the choice of cardinality enables explicit trade-off between ease of implementation and efficiency. Rather than pre-committing to a fixed subset of pure strategies to be used in the randomization, our model decides on the *best* subset of policies to employ. Our hope (which we confirm with extensive experiments, see Section 6.6) is that the *price of usability*, i.e., the loss in efficiency due the restriction of the space of feasible mixed strategies, will not be high even if only a moderate number of strategies is employed.

Definition 1 (k-Operationalizable Mixed Strategy)**.** *A mixed strategy p is k-operationalizable if the cardinality of the support of p is limited to k, i.e. $|\{i \in \{1, \ldots, S\} : p_i > 0\}| \leq k$.*

For usability's sake, we propose to restrict solutions of security games to be k-operationalizable. A large k produces solutions that randomizes over a large number of pure strategies (maximizes optimal utility but not easy to operationalize) and low k produces more deterministic strategies (easy to operationalize, but exploitable by an intelligent adversary). We can balance between usability and efficiency using the single parameter k. The following theorem postulates that unfortunately usability comes at a *computational price*.

Theorem 8. *Let $\mathcal{G}$ be a zero sum game with pure strategy space $\mathcal{Q}$. The problem of finding optimal solutions that are k-operationalizable is NP-Hard to solve even if $\mathcal{G}$ can be solved in polynomial time.*

Proof. Given an arbitrary instance of the set cover problem, a number k, a universe $\mathcal{U}$ and a collection of sets $S = \{S_1, S_2, \ldots, S_m\}$ we want to determine if there is a set covering of size k or less ie. a subset $C \subseteq S$, such that $|C| \leq s$. Let G be a game with a number of uniform resources equal to $\max_i |S_i|$, with target set $\mathcal{U}$ and scheduling constraints such that $Q = S$. Let each target have value V, so that the defender receives a utility of $-V$ if the target is unprotected and a utility of zero if the target is protected. Let x_i be the probability that target i is covered by a resource. The defender's expected utility is then $E_x[U] = -\max_{i \in \mathcal{U}} V(1 - x_i)$. If there exists an k-operationalizable solution to G with expected utility $E_x[U] > -V$ that means that it is possible for the defender to cover all the targets with some probability using only k pure strategies. These pure strategies in support of the k-operationalizable strategy then give a set cover if size k (or less). Therefore if the corresponding game has an optimal objective value greater than $-V$ there exists a set cover $C \subset S$ such that $|C| \leq k$. □

For a player in such a game $\mathcal{G}$, an optimal mixed strategy which is k-operationalizable is one which minimizes that player's *Price of Usability.*

Definition 2 (Price of Usability)**.** *Let G be a game with optimal mixed strategy solution p and utility $U(p)$. Let p^k be an optimal k-operationalizable mixed strategy solution to G. We define the price of usability (PoU) as the ratio between the utilities of p and p^k so that $PoU := \frac{U(p)}{U(p^k)}$.*

6.3 Operational SORT Problems

The SORT problem admits additional usability concerns; not only can the mixed strategy have a very large support, but the number of types of resource configurations (teams) used

by any pure strategy may also be very large (as the number of team types grows combinatorially with the number of resources). This can also pose the same operationalization issues, and so we also propose to limit the number of possible resource configurations that may be used in any pure strategy. Formally, we say that a mixed strategy solution to a SORT problem is operationalizable if the following property holds.

Definition 3 ((k, τ)-Operationalizable Mixed Strategy)**.** *A mixed strategy p is said to be (k, τ)-operationalizable if the support size of p is less than k, and each pure strategy uses no more than τ unique teams, i.e., if l_t is a binary variable indicating the formation of a team of type t then $\sum_{t=1}^{T} l_t \leq \tau$.*

We can compute *operationalizable strategies* for the SORT problem by constructing a new set of allowed pure strategies $\mathcal{Q}_\tau$ by adding the following additional constraints to the set Q which enforce that each pure strategy i may use no more than than τ resource configurations:

$$\sum_{t=1}^{T} x_{t,i} \leq \tau \tag{6.1}$$

We call l_t an operational variable so that $t_t = 1$ if a strategy uses team t at any point.

6.4 Motivating Domain and Game Model

As a motivating example we take the Threat Screening Game introduced in Chapter 2. This model was developed in collaboration with the US Transportation Security Administration (TSA) to tackle the problem of screening for threats at checkpoints. As development of this model continues it is important that the usability and implementability of the solutions

to this model is taken into account, to ensure that it is successfully integrated into the TSA security system.

The first challenges we address here relates to the practical difficulty of integrating decisions made by different levels of authority: *a)* higher level strategic decisions related to the design of teams of security resources and the assignment of personnel to shifts and teams; and so we provide a model of the SORT problem for TSG in order to explicitly reason about staff scheduling constraints and how to form teams of screening resources, subject to capacity and staff scheduling constraints.

6.4.1 SORT for Threat Screening Games

Addressing the two usability limitations of TSG [Brown et al., 2016, Schlenker et al.,] discussed earlier, in this section we first present (1) the model of Simultaneous Optimization (SORT) for TSGs and second (2) the problem of computing operationalizable strategies for TSGs. We assume a zero-sum game. Throughout this section we use the example of passenger screening at airports, but emphasize that the TSG applies to generalized screening problems. The TSA TSG is a game between the screener (defender) and adversary over a finite planning horizon $\mathcal{W}$ consisting of W time windows. The defender is operating a checkpoint through which screenees (passengers) arrive at during each time window. Each screenee belongs to a category $c \in \mathcal{C}$ where a category $c := (\rho, f)$ consists of components which are controllable and uncontrollable. In the airport security domain, the controllable component f corresponds to a flight type, dictated by the adversary's choice of flight to attack, while the uncontrollable element ρ describes the risk level assigned to each passenger (i.e. if they are TSA pre-check). It is assumed that the number of passengers of category c arriving during each time window, N_c^w, is known.

The adversary attempts to pass through screening by disguising themselves as one of the screenees. He has a choice of flight to attack, and thus can choose his flight type category, a time window w to arrive in and an attack method $m \in \mathcal{M}$. The adversary cannot control his risk level ρ and we assume a prior distribution P_ρ over the risk level of the adversaries.

At the checkpoint, the defender has a set of $r \in \mathcal{R}$ resources which are combined into teams indexed in the set $\mathcal{T}$ to which incoming passengers are assigned. If a passenger is assigned to be screened by a team $t \in \mathcal{T}$, they must be screened by all resources $\mathcal{R}(t) \subset \mathcal{R}$ in that team. The efficiency of a team, $E_{t,m}$, denotes the probability that an attacker carrying out an attack of type m be detected when screened by team t. This efficiency depends on the resources in that team: $E_{t,m} = 1 - \prod_{r \in \mathcal{R}}(1 - e_{r,m})$, where $e_{r,m}$ is the efficiency of resource r against attack method m.

Each resource $r \in \mathcal{R}$ has a fixed capacity C_r for the number of passenger which it can process in any time window. In the case that it is not possible to screen all passengers in a single time window, we allow these passengers to be screened in the next time window by their assigned resources, at a cost ϕ_r per passenger overflowing to the next window. Each resource r maintains an overflow queue o_r^w corresponding to the number of passengers waiting to be processed by that resource in the beginning of time window w.

To speed up processing, the defender can increase the number of resources of each type that are available in a particular window (by e.g., opening up more lanes). However, the number of resources of each type r that can be operated at any given time is limited by the number of resources of that type that are available in the arsenal of the defender, denoted by $M_r \in \mathbb{R}$, and by the number of operators that are working in that window. Specifically, to operate each resource of type r, A_r workers are needed. The workforce of the defender

consists of S workers and the defender can decide on the number of workers available in any window. However, the workers must follow shifts: they can start in arbitrary time windows but must work for δ consecutive time windows.

6.4.2 SORT-TSG Problem Formulation

We now formulate the SORT problem for TSG as a mixed-integer linear optimization problem. For convenience, we first introduce the pure strategy spaces related to the strategic and tactical decisions of the defender, respectively, and then go on to formulate the optimization problem which randomizes over these strategies.

The core strategic decisions of the SORT-TSG problem correspond to the number of resources of each type $r \in \mathcal{R}$ to operate in each window, which we denote by $y_r^w \in \mathbb{N}^+$. They also include the number of workers $b^w \in \mathbb{N}^+$ to start their shift in window w and the number of workers s^w available in window w. The space of pure strategic strategies can then be expressed as:

$$\mathcal{Y} := \left\{ y : \exists (b, s) : \begin{array}{ll} s^w = \sum_{w'=\max(1,w-\delta+1)}^{\min(w,W-\delta+1)} b^{w'} & \forall\, w \\ \sum_{w=1}^{W-\delta+1} b^w \leq S & \\ \sum_{r \in \mathcal{R}} y_r^w A_r \leq s^w & \forall\, r \\ y_r^w \leq M_r & \forall\, w, r \\ y_r^w, b^w, s^w \in \mathbb{N}^+ & \forall\, w, r \end{array} \right\}.$$

The first constraint above counts the total number of workers with shifts currently in progress at time window w. The second constraint stipulates that the total number of workers assigned to each shift cannot exceed the size of the workforce. The third and fourth constraints enforces that in each time window there must be enough workers to operate each resource, and that the number of operating resources cannot exceed the maximum number available for each type.

The core tactical decision variables of the SORT-TSG problem correspond to the number of passengers of each type c to screen with team t in window w, denoted by $n^w_{t,c}$. For any choice y of strategic decisions, the space of pure tactical strategies is expressible as:

$$\mathcal{X}_y := \left\{ (n,o) : \begin{array}{ll} \displaystyle\sum_{t\in\mathcal{T}} n^w_{c,t} = N^w_c & \forall\, c,w \\ \displaystyle\sum_{t:r\in R(t)}\sum_{c} n^w_{c,t} \le y^w_r C_r - o^{w-1}_r + o^w_r & \forall\, w,r \\ n^w_{t,c}, o^w_r \in \mathbb{N}^+ & \forall\, t,c,w,r \end{array} \right\},$$

where the two constraints above stipulate that all arriving passengers must be assigned to be screened by a team and enforce the capacity constraints on each of the resource types. Note that the capacity is determined by the number of operating resources of each type. The full defender pure strategy space can be expressed compactly as:

$$\mathcal{Q} := \{(y,n,o) : y \in \mathcal{Y},\ (n,o) \in \mathcal{X}_y\}.$$

Next, given the probability distribution as the defender's mixed strategy, we denote by $E_p[\ ]$ the expectation operator with respect to p (the mixed strategy). Thus, the expected number $n^w_{t,c}$ of passengers in category c screened by team t in time window w and the

expected number o_r^w of passengers waiting to be screened by a resource of type r in time window w are given by:

$$E_p[n_{t,c}^w] := \sum_{i=1}^{S} p_i n_{t,c}^{w,i} \quad \text{and} \quad E_p[o_r^w] := \sum_{i=1}^{S} p_i o_r^{w,i}. \tag{6.2}$$

The utility of the defender is linear in the pure strategies, so the defender's optimization problem can be expressed as:

$$\begin{aligned} \max_{p} \quad & \textstyle\sum_\rho P_\rho \theta_\rho - \sum_w \sum_r \phi_r E_p[o_r^w] \\ s.t. \quad & \theta_\rho \leq z_{c,m}^w U_c^+ + (1 - z_{c,m}^w) U_c^- \quad \forall\, c, m, w \\ & z_{c,m}^w = \textstyle\sum_t E_{t,m} \frac{E_p[n_{c,t}^w]}{N_c^w} \qquad \forall\, c, m, w \\ & p \in \mathcal{P}, \end{aligned} \tag{$\mathscr{P}$}$$

where $z_{c,m}^w$ is the adversary's detection probability for an adversary of type c, using attack m during w and θ_ρ is the expected utility when the passenger's risk level is ρ. We denote this formulation of the SORT-TSG as problem $\mathscr{P}$.

Theorem 9. *Problem $\mathscr{P}$ is NP-Hard to solve.*

Proof. Reduces to [Brown et al., 2016] when there is no overflow and when strategic decisions are fixed. □

6.4.3 Operationalizable Strategies for SORT-TSG

To address the usability concerns related to *operationalizable mixed strategies*, we here provide a formulation for computing $(k, \tau)-$*operationalizable strategies* for the SORT-TSG problem. We construct a new set of allowed pure strategies $\mathcal{Q}_\tau$ by adding the following additional constraints to the orginal set Q which enforce that each pure strategy may use no more than than τ resource configurations:

$$\sum_{t=1}^{T} l_t \leq \tau; \quad \tfrac{n^w_{c,t}}{N^w_c} \leq l_t, \quad \forall\, t, w, c; \quad l_t \in \{0, 1\}, \quad \forall\, t. \tag{6.3}$$

Where the second constraint enforces that $l_t = 1$ if a strategy uses team t at any point, i.e., if $\exists\ w, c, t \ \ : \ \ n^w_{c,t} > 0$ We then enforce that the support of the mixed strategy has maximum cardinality k by replacing equations (6.2) with:

$$E_p[n^w_{t,c}] = \sum_{i=1}^{k} p_i n^{w,i}_{t,c} \quad E_p[o^w_r] = \sum_{i=1}^{k} p_i o^{w,i}_r \quad \forall\, w, t, c, \tag{6.4}$$

such that $p \in \mathcal{P}_k = \{p_i \geq 0, i = 1, ..., k,\ \sum_{i=1}^{k} p_i = 1\}$. Lastly, for the TSG problem it is undesirable to have many different schedules for staff members and have employees work different shifts throughout the week. For this reason we specifically enforce that the scheduling decisions s should be the same across all k pure strategies i.e.

$$s_i = s_j \ \ \forall i, j \in \{0 \ldots k\}. \tag{6.5}$$

These additions (2,3,4) to $\mathscr{P}$, define the operationalizable SORT-TSG problem, we refer to the problem as $\mathscr{P}\mathrm{k}$.

6.5 Solving the Operationalizable SORT-TSG

The SORT-TSG problem is expressible as a mixed integer linear program (MILP). However the resulting operationalizable problem $\mathscr{P}\mathrm{k}$ is non-linear, with bilinear terms introduced in (3). Since the domains of n and o are finite we can express each integer variable n and o as a sum of binary variables, and the bilinear terms can be easily linearized using standard optimization techniques. However, the resulting program has a number of binary variables which grows with the number of passengers, making the full MILP formulation intractable to solve. Other standard approaches for dealing with these types of problems, such as column generation, also do not work well as we show in Section 6.6. In the following, we provide our new solution approach for efficiently solving $\mathscr{P}\mathrm{k}$.

For convenience we define the following notation. Let $\mathscr{P}$ be an optimization problem with integer variables $x_i \in \mathbb{N} \ \ \forall i$. We denote the LP relaxation of $\mathscr{P}$, i.e., the problem obtained by letting $x_i \in \mathbb{R} \ \ \forall i$, as $\mathscr{P}^{\,LP}$. Additionally let the LP relaxation of a problem $\mathscr{P}$ with respect to a single variable x_j, i.e., the problem obtained by letting $x_j \in \mathbb{R}$, be denoted by $\mathscr{P}^{\,LP_{x_j}}$. Let the marginal value of x_j (i.e., the expectation $E_p[x_j]$) be denoted $\tilde{x}_j$. Lastly we denote the problem with a fixed variable x_j as $\mathscr{P}\,|x_j$.

Our novel solution approach $\mathscr{P}\mathrm{k}$ is based on the two following ideas: (1) we allow the k pure strategies to form a *multiset* (so that a single strategy may appear multiple times) and (2) we restrict the mixed strategy to be a *uniform distribution* over the multiset of k pure strategies. The intuition behind this approach is that the multiset allows us to approximate any optimal mixed strategy $\mathcal{P}$ using multiples of the fractions $\frac{1}{k}$. If $p_i \geq \frac{1}{k}$ (probability of playing strategy i), then strategy i will appear multiple times in the multiset, and thus will be played with probability $\frac{a}{k}$ where a is the number of times it appears. If

$p_i < \frac{1}{k}$ then as k grows large enough, the loss in utility from not playing strategy i becomes negligible.

This intuition is formalized in Theorem 10 which stipulates that we can compute approximate equilibria (with approximation error ϵ) for any choice of k by fixing a uniform distribution over the multiset of k pure strategies.

Theorem 10. *Given a game G with Stackelberg equilibrium x^*, z^* and game value $(x^*)^\top R z^*$ there exists a solution x', z' such that x' is k-operationalizable and is* uniformly distributed over its support *where for $k > \frac{4\log(1+n)}{\epsilon^2}$ (where n is the size of the adversary's action space) we have that x', z' is an ϵ-Stackelberg equilibrium with game value $(x^*)^\top R z^* - (x')^\top R z' \leq \epsilon$.*

Proof. Let G be a zero sum matrix game with payoff matrix R. Let x^*, z^* be a Stackelberg equilibrium solution to game G with game value $(x^*)^\top R z^*$ for the leader. Construct an s-finite adaptive solution x' by sampling from x^*, s times. We want to show that there exists an x' with uniformly distributed support over s strategies and adversary best response y' such that, (x', y') form an ϵ-equilibrium:

$$(x')^\top R z_i - (x')^\top R z' < \epsilon \qquad \forall i \in Q_a, \;\; |\mathcal{Q}_a| = n \tag{6.6}$$

And the corresponding game value is ϵ away from the Stackelberg equilibrium game value:

$$(x^*)^\top R z^* \geq (x')^\top R z' + \epsilon \tag{6.7}$$

Let x' be the defender mixed strategy. We construct x' from x^* by drawing s iid samples from the defender pure strategy space according to distribution x^*, and assigning each sample probability $\frac{1}{s}$ in x'. We define the events:

$$E_1 = \{(x^*)^\top R z^* \geq (x')^\top R z' + \epsilon\}$$

$$E_{2,i} = \{(x')^\top R z_i - (x')^\top R z' < \epsilon\}$$

If we can show that the probability of all the events $E_1 \cap_i^n E_{2,i}$ occurring is non zero then there must exist an (x', y') such that these equations hold. What we will do is show that:

$$P(E_1 \cap_i^n E_{2,i}) = 1 - P(\neg(E_1 \cap_i^n E_{2,i})) = 1 - P(\neg E_1) - \sum_i^n P(\neg E_{2,i}) > 0$$

Following [Lipton *et al.,* 2003] we define several auxiliary events. Let

$$E_{1,a} = \{(x')^\top R z^* - (x^*)^\top R z^* \geq \frac{\epsilon}{2}\} \tag{6.8}$$

$$E_{1,b} = \{(x')^\top R z' - (x^*)^\top R z' \geq \frac{\epsilon}{2}\} \tag{6.9}$$

We note (as in [Lipton *et al.,* 2003]) that

$$\begin{aligned} E_1 &\supseteq E_{1,a} \cap E_{1,b} \\ P(E_1) &\geq P(E_{1,a} \wedge E_{1,b}) \\ P(\neg E_1) &\leq P(\neg_{1,a}) + P(\neg E_{1,b}) \end{aligned} \tag{6.10}$$

Similarly define:

$$E_{2,a} = \{(x')^\top Rz' - (x^*)^\top Rz' < \frac{\epsilon}{2}\}$$

$$E^b_{2,i} = \{(x')^\top Rz^i - (x^*)^\top Rz^i < \frac{\epsilon}{2}\}$$

so that:

$$\begin{aligned} E_{2,i} \supseteq &\quad E_{2,a} \cap E^b_{2,i} \\ P(E_{2,i}) \geq &\quad P(E_{2,a} \wedge E^b_{2,i}) \\ P(\neg E_{2,i}) \leq &\quad P(\neg E_{2,a}) + P(\neg E^b_{2,i}) \end{aligned} \tag{6.11}$$

Again following [Lipton *et al.,* 2003] we note that $(x')^\top Rz^?$ is a sum of k independent random variables each of expected value $(x^*)^\top Rz^?$ with each random variable having value between 0 and 1. We can apply a standard concentration bounds (using Hoeffding's inequality) and get:

$$P(\neg E_{1,a}) \leq e^{-\frac{k\epsilon^2}{2}}$$

$$P(\neg E_{1,a}) \leq e^{-\frac{k\epsilon^2}{2}}$$

$$P(\neg E_1) \leq 2e^{-\frac{k\epsilon^2}{2}}$$

Similarly:

$$P(\neg E_{2,a}) \leq e^{-\frac{k\epsilon^2}{2}}$$

$$P(\neg E^b_{2,i}) \leq e^{-\frac{k\epsilon^2}{2}}$$

$$P(\neg E_{2,i}) \leq 2e^{-\frac{k\epsilon^2}{2}}$$

So that finally:

$$\begin{aligned} P(\neg E_1 \cap_i^n E_{2,i}) &\leq 2e^{-\frac{k\epsilon^2}{2}} + 2ne^{-\frac{k\epsilon^2}{2}} \\ &\leq 2(1+n)e^{-\frac{k\epsilon^2}{2}} \end{aligned}$$

For $k > \frac{4log(1+n)}{\epsilon^2}$ we then have that $P(\neg(E_1 \cap_i^n E_{2,i})) < 1$ and $P(E_1 \cap_i^n E_{2,i}) > 0$ which concludes the proof.

□

We derive these bounds following the proof of [Lipton et al., 2003], which for our problem are a factor 3 tighter. By fixing $p = \frac{1}{k}$, $\mathscr{P}\mathrm{k}$ can be solved directly as an MILP without the creation of extra binary variables. Algorithm 6 outlines this process. To speed up computation we first solve the full relaxation $\mathscr{P}\mathrm{k}^{\,LP}$ to get marginal values $\tilde{y}$ and $\tilde{n}$ (line 2). We then round these to get integral values y^r and n^r (line 3) which we then use as a warm start to solve the MILP (line 5).

Algorithm 6: k-Uniform Strategies

1: **procedure** k-UNIFORM
2: $\tilde{y}, \tilde{n}, \tilde{o} \leftarrow \mathscr{P}\mathrm{k}^{\,LP}$
3: $y^r, n^r, o^r \leftarrow$ Round($\tilde{y}, \tilde{n}, \tilde{o}$)
4: $p \leftarrow p_i = \frac{1}{k},\ i = 1, ..., k$
5: $y, n, o \leftarrow$ WarmStart($\mathscr{P}\mathrm{k}\ |q_k, y^r, n^r, o^r$)
6: **return** y, n, o

For any choice of k, we can then compute an ϵ-equilibrium and show that in practice this approach performs well. Additionally, it provides a general framework from which we can build more sophisticated and scalable algorithms which we demonstrate in the next section.

6.5.1 Heuristic Approach

While the approach described in Section 6.5 provides guarantees for any choice of k, in practice the problem can still be slow to solve, as it requires solving an MILP. Thus we provide a heuristic approach which can be solved more efficiently and still yields high solution quality in practice.

The novelty in our approach comes from exploiting the hierarchical structure of the SORT variables, as well as an optimized rounding procedure to decompose marginal solutions into an operationalizable set of k integral strategies.

The tactical variables (n, o) are dependent on the strategic variables y and so, starting from marginal solution to the LP relaxation, we first impose the operationalizable constraints on the strategic variables, keeping the tactical variables unconstrained. This gives us a set of k strategies with integral y, from which we can compute the corresponding integral tactical variables n for each of the k strategies. Both of these steps use an *optimized rounding procedure*. Because our objective is a function of the expected value of n and o, it becomes important to optimize over how we do our rounding. Ideally we would like to be able to exactly reconstruct the marginal values obtained from the LP relaxation in order to maximize our objective. Arbitrarily rounding the marginal variables to generate k integral strategies does not take into account the value of the resulting marginal and may result in very suboptimal solutions. Instead we compute an optimal rounding to compute feasible solutions, which take into account the value of the resulting marginal with respect to our objective.

Algorithm 2 outlines the steps of this solution method. We start by solving the full relaxation $\mathscr{P}\mathrm{k}^{LP}$ (line 2) to obtain a marginal solution for the strategic variables $\tilde{y}$. We then decompose this marginal solution into a set of k integral pure strategies (line 3) using

an optimized rounding procedure (which we formalize in the later section) which computes the best k roundings of the marginal $\tilde{y}$ (keeping a marginal $\tilde{n}_i$ for each strategy i, $i = 1, ..., k$). We then compute the best integral assignment n_i and corresponding overflow o_i for each resource configuration y_i (line 4) using the same optimized rounding procedure on the marginals $\tilde{n}_i$, $i = 1, ..., k$.

Algorithm 7: Multiple Hierarchical Relaxations

1: **procedure** MHR
2: $\tilde{y}, \tilde{n}, \tilde{o} \leftarrow \mathscr{P}\mathrm{k}^{LP}$
3: $y_i, \tilde{n}_i, \tilde{o}_i \;\; i = 1, ..., k \leftarrow$ Strategic($\mathscr{P}\mathrm{k}^{LP_n} | q, \tilde{y}$)
4: $y_i, n_i, o_i \;\; i = 1, ..., k \leftarrow$ Tactic($\mathscr{P}\mathrm{k} \, | y, \tilde{n}$)
5: **return** y, n, o

Strategic Variables: Resource Configurations At this stage (line 3) we determine what the k optimal integral variables y_i are assuming no integrality constraints on the n_i variables, i.e. we solve the problem $\mathscr{P}\mathrm{k}^{LP_n}$ equivalent to letting $E_p[n^w_{t,c}] = \sum_{i=1}^{k} p_i \tilde{n}^{w,i}_{t,c}$ and $E_p[o^w_r] = \sum_{i=1}^{k} p_i \tilde{o}^{w,i}_r$, where $\tilde{n}^{w,i}_{t,c}, \tilde{o}^{w,i}_r$ are in the integer relaxation of $\mathcal{X}_{y_i}$. Unfortunately the following theorem shows that $\mathscr{P}\mathrm{k}^{LP_n}$ is still intractable to solve.

Theorem 11. *Problem $\mathscr{P}\mathrm{k}^{LP_n}$ is NP-hard to solve.*

Proof. We give a reduction from the knapsack problem with unbounded items, which is NP-hard. The knapsack problem with unbounded items is described as follows. Given a set of n types of items, where v_j is the value of each item and w_j is the weight of each item, find an allocation of items to the knapsack such that the minimum value over all knapsacks is maximized.

Given an instance of the multiple-knapsack problem with n types value v_i and weights w_j. This can be reduced to the following instance of $\mathscr{P}\mathrm{k}^{LP_n}$. Let $n + 1$ be the number of

resources. The number of time windows, passenger types and attack methods equal 1, so we omit the indices w, c and m. Also, we choose $k = 1$.

Construct for each resource a team consisting of that resource. The efficiency of each team, E_t, corresponding to resource t equals the value v_t, $t = 1, ..., n$. Additionally, we have one team with resource $n + 1$ and the efficiency of this team E_{n+1} equals 0. The capacity of each team, C_r, equals 1.

The number of people required to use one resource, A_r, is given by v_r, $r = 1, ..., n$ and A_{n+1} is chosen to be 0. The maximum number of resources, M_r, is chosen such that $M_r \geq \frac{P}{A_r}$.

We choose the number of passengers arriving, N, such that it can never occur that all passengers can be send to resources $r = 1, ..., n$. The remaining passengers can be send to resource $n+1$. Different choices of N are possible, we choose $N = \max_r M_r$. The utility of a successful attack, U^+, is chosen as N and the utility of an unsuccessful attack, U^-, is 0.

Finally, we choose the value of ϕ_r so high that the overflow variables are always chosen as 0. Note that this model always gives a feasible solution since all remaining passengers be be send to the team with resource $n + 1$

Solving $\mathscr{P}\mathrm{k}^{LP_n}$ with the parameters described above gives an optimal solution for the knapsack problem with n types of items, where the resources scheduled corresponds to the items assigned to the knapsack. The optimal value of $\mathscr{P}\mathrm{k}^{LP_n}$ equals the optimal value of the knapsack problem. This can be seen as follows. If a specific resource (item) r, $r = 1, ..., n$ is scheduled, then 1 out of N passengers will be send to that team. So, this resource adds $\frac{E_r}{N}$ $(= \frac{v_r}{N})$ to the total value of z. Since z is multiplied by U^+ $(= N)$, this resource add v_r to the total value of that time window. □

To approximate this problem, as in Algorithm 6, we assume a uniform distribution for the mixed strategy. Note that by Theorem 10 for any choice of k that $\mathscr{P}\mathrm{k}^{\,LP_n}|p$ is an ϵ approximation to $\mathscr{P}\mathrm{k}^{\,LP_n}$. Given $\mathscr{P}\mathrm{k}^{\,LP_n}|p$, we compute a multiset of k integral solutions y_i, $i = 1, ..., k$, from the marginal $\tilde{y}$ using the following optimized rounding procedure. We make the change of variables $y_i = \lfloor \tilde{y} \rfloor + \delta_i$ such that $\delta_i \in \mathbb{N}^+$, $i = 1, ..., k$. Solving $\mathscr{P}\mathrm{k}^{\,LP_n}|p$ with this change of variables computes the best k roundings of the marginal $\tilde{y}$ which we use as our k pure strategies. This subroutine is outlined in Algorithm 8.

Algorithm 8: Determine resource allocations y_s

1: **procedure** STRATEGIC($\mathscr{P}\mathrm{k}^{\,LP_n}|p$, $\tilde{y}$)
2: $\quad p \leftarrow p_i = \frac{1}{k},\ i = 1, ..., k$
3: $\quad y \leftarrow y_i = \lfloor \tilde{y} \rfloor + \delta_i,\ \delta_i \in \mathbb{N}^+,\ i = 1, ..., k$
4: $\quad$ **return** $\arg\max_{y,\tilde{n},\tilde{o}} \mathscr{P}\mathrm{k}^{\,LP_n}|p$

Tactic Variables: Passenger Allocations (line 4) We now have for each pure strategy i, a marginal $\tilde{n}_i$, $i = 1, ..., k$. In this step we again apply the same optimized rounding procedure to these variables to obtain integral values n_i. Additionally, here we relax the constraint $p_i = \frac{1}{k}$ and allow the program to optimize over the distribution over pure strategies.

Reintroducing the mixed strategy p as a variable reintroduces the bilinear terms (6.4) in $\mathscr{P}\mathrm{k}$. However, with our rounding procedure, we can efficiently linearize these terms without creating a very large number of binary variables (as with the full MILP). We let

$n_i = \lfloor \tilde{n}_i \rfloor + \gamma_i$ and are left with the bilinear terms $p_i(\gamma_i)$. To linearize these we make a change of variable $z_i = p_i(\gamma_i)$ and can express constraints (6.4) as:

$$E[n^w_{t,c}] = \sum_{i=1}^{k} \left(p_i \left\lfloor \tilde{n}^{w,i}_{t,c} \right\rfloor + z^{w,i}_{t,c} \right),$$

$$0 \leq z^{w,i}_{t,c} \leq p_i, \qquad \forall\, i = 1, ..., k.$$

$$z^{w,i}_{t,c} \geq p_i - (1 - \gamma^{w,i}_{t,c}), \qquad \forall\, i = 1, ..., k.$$

This subroutine is outlined in Algorithm 9. First, we make the change of variable for the rounding procedure, and linearize the bilinear terms. We then solve the resulting optimization problem for the fixed y and b solved in the previous stage of the algorithm and finally return n and p which gives us a complete (k, τ)-operationalizable solution.

Algorithm 9: Determing Passenger type allocation n_s

1: **procedure** TACTIC($\mathscr{P}\mathrm{k}\,|(y, b)$, $\tilde{n}$)

2: $n \leftarrow n_i = \lfloor \tilde{n}_i \rfloor + \delta_{n_i},\ \delta_{n_i} \in \mathbb{N}^+, i = 1, ...k$

3: LinearizeTerms($\mathscr{P}\mathrm{k}\,|(y, b)$)

4: **return** $\arg\max_{n,p} \mathscr{P}\mathrm{k}\,|(y, b)$

6.6 Evaluation

We evaluate our algorithms on several different sized instances of SORT-TSG. We use instances of three types: small, moderate and large instance with time windows, passenger types and resources (W=1, C=2, R=2), (W=5, C=10, R=5), (W=10, C=20, R=5) respectively. The large instances correspond to a 10 hour planning window for a single terminal at a

large airport. [1] Each experiment is averaged over 50 randomized instances of the remaining parameters.

The Price of Usability In this paper, we proposed to mitigate the *price of usability (PoU)* by computing (k, τ)-operationalizable strategies. We have defined the price of usability similarly to the price of anarchy, as the ratio between the optimal solution with no usability constraints and the operationalizable equilibrium, (i.e. $\mathscr{P}$ /$\mathscr{P}$k) so that when the operationalizable game $\mathscr{P}$k has the same optimal objective as $\mathscr{P}$ the $PoU = 1$. In order to

[1] LAX (Los Angeles airport) has an average of 20 unique flight types per terminal (185 destination locations spread over 9 terminals).

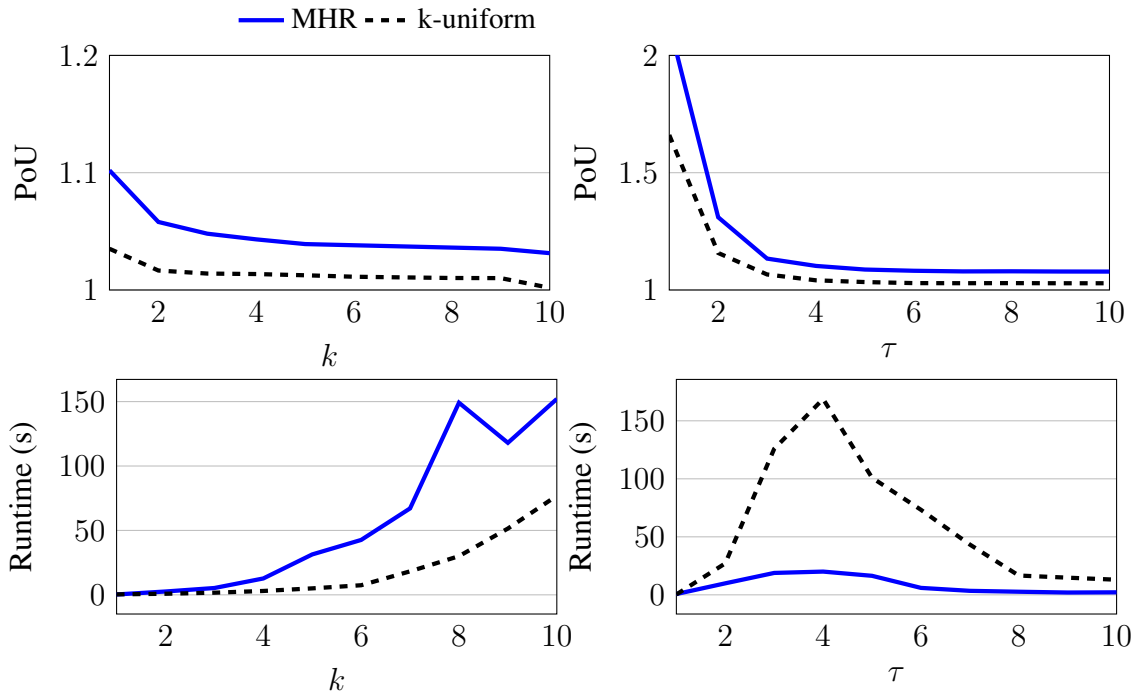


Figure 6.1: Here we show the empirical PoU, as well as the runtimes of both methods with increasing k and τ for both methods (left: $\tau = 10$, right: $k = 2$).

compare the operationalizable utility to that of $\mathscr{P}$, we use column generation to compute the optimal solution to the security game without usability constraints. We do this for moderately sized games, as the column generation method does not scale up to large instances. In Figure 6.1, we show that the PoU shrinks to almost 1 with increasing number of pure strategies k and team types τ. We note that the bump in runtime with increasing τ is due to a phenomenon in security games known as the deployment to saturation ratio [Jain et al., 2012b].

Solution quality We evaluate the solution quality of our algorithms by comparing to (1) two variations of a column generation heuristic, one which cuts off after I iterations and one which selects and re-optimizes over the top k strategies, and (2) the full MIP which optimally solves operationalizable security game $\mathscr{P}$k . Figure 6.2(a) shows the comparison of our methods with the first column generation (CG) baseline. When run to convergence, (CG) optimally solves $\mathscr{P}$, without operationalizable constraints (CG). We approximate $\mathscr{P}$k by cutting off (CG) after I iterations. We see that for small I, CG achieves very poor utilities compared to our algorithm, and that it takes up to 150 generated strategies (iterations) to match the solution quality of our methods. Additionally we investigate the support size of the mixed strategies computed by (CG) without operationalizable constraints. Figure 2(b) shows that number of strategies used grows as we increase the problem size (here, the number of flight types). We also compared to a second variation of the column generation method where we pick the top k pure strategies, and compute the optimal mixed strategy over these k strategies. This was done cutting column generation off after 10, 20, 50, 100 columns as well as after full convergence. The results are shown in Figure 6.3. We see on average a 30% loss in PoU when using this baseline compared to

our methods, and in the worst case up to 100% loss with $PoU \sim 2$ for the baseline when compared to our methods. This demonstrates that we can significantly reduce the support size and still obtain a $PoU \sim 1$

In Table 6.1, we compare utility of our algorithms with the utility obtained from solving the full MILP (which optimally solves $\mathscr{P}\mathrm{k}$). The full MILP can only be solved for small instances (maximum of $k = 3$). For these instances, we see that both our methods produce

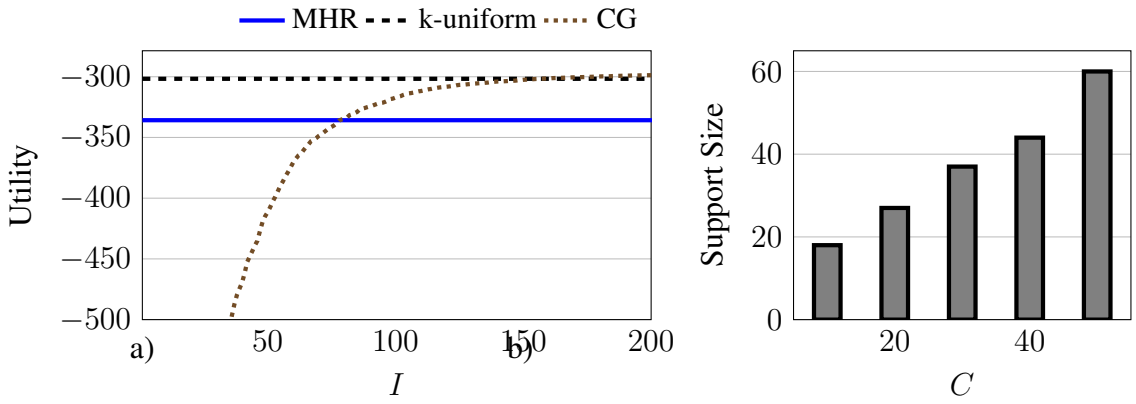


Figure 6.2: a) Comparison of our algorithms with CG which is cut off after I iterations ($k = 5$, $\tau = 10$). b) Support size of CG solutions for increasing problem size.

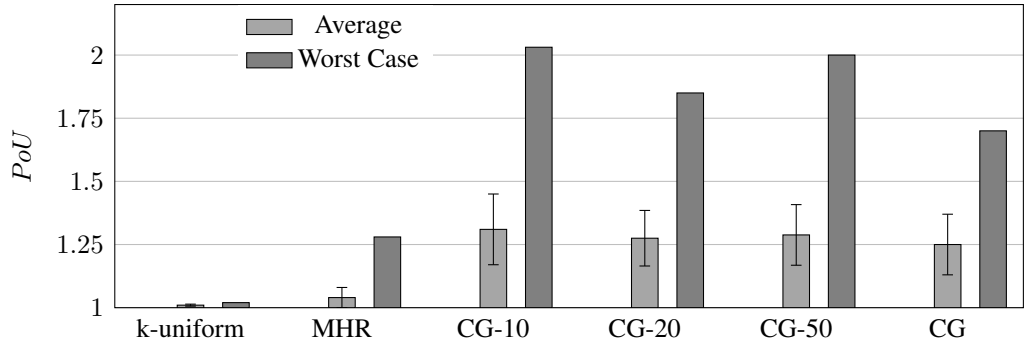


Figure 6.3: Average case price of usability and b) worst case price of usability, for our two methods (k-uniform and MHR) compared to a cutoff column generation baseline. Column generation (CG) was cutoff after 10, 20 and 50 columns and after convergence.

near-optimal solutions and can be executed significantly faster. For moderate and large sized instances, we see the k-uniform algorithm outperform MHR in terms of utility, but that MHR can solve large instances faster.

	Small		Moderate		Large	
	u*	rt(s)	u*	rt(s)	u*	rt(s)
k-uniform	-85.3	0.2	-543	48	-1258.8	219.4
MHR	-87.0	0.1	-661	20.1	-1315.8	91.2
MILP	-85.2	1154.3	-	-	-	-

Table 6.1: Runtime and utility u^* of the k-uniform and MHR algorithm compared with the solution of the full MIP (small: $k = 3$, moderate,large: $k = 4$).

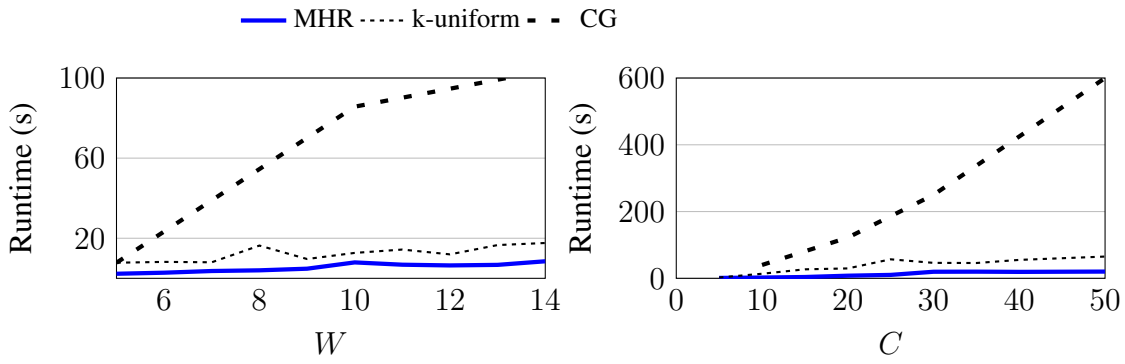


Figure 6.4: Runtime for different values of W and C ($k = 2$, $\tau = 5$, left: $C = 10$, right: $W = 5$).

Scalability To evaluate the scalability of our algorithms, we compare the running time for different time windows W and number of passenger categories C. Figure 6.4 shows the running time for different values of W and C where the rest of the parameters are fixed. This figure shows that the running time is only slightly increasing in W and that our algorithms can be scaled up to a very large number of passenger types.

Non-uniform distribution of q In the heuristic approach proposed in Section 6.5, we use an uniform distribution over q. For large k, many distributions can be reconstructed in this way. However, when we only consider a small number of allowed strategies k it might be better to optimize over different distributions of q. We have tested this on a moderate instance for $k = 2, 3, 4$, using the the MHR algorithm described in Section 6.5.1. Instead of using a uniform distribution in 8, we solve this step for different distributions of q. We only allow for 'simple' distributions, consisting of sixths, fifths, fourths, thirds and seconds.

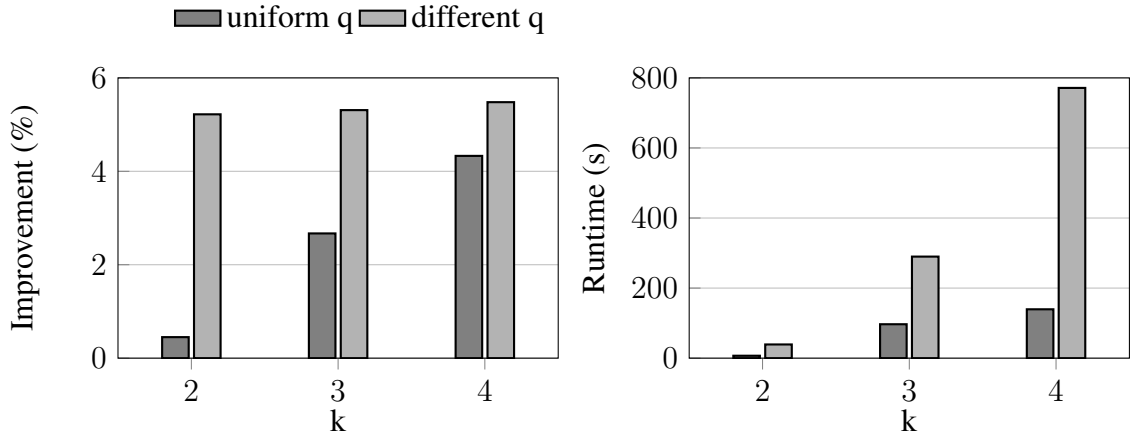


Figure 6.5: Improvement (compared to utility when $k = 1$) and runtime when varying over q ($\tau = 10$).

Figure 6.5 illustrates that varying over different distributions of q improves the solution quality a lot for small k. However, when k is increasing the benefit of different distributions decreases since more distributions can already be constructed. Also, the running time increases significantly because the same algorithm has to be run for different distributions.

6.7 Chapter Summary

This chapter introduces a fundamental new problem of operationalizable strategies in security games and provide a single framework which reasons about the three levels of planning: strategic, tactical and operational level decision problems. Motivated by the important problem of screening for threat we provide algorithmic solutions to overcome the computational challenges that arise when these planning problems are addressed for TSGs and which mitigate the Price of Usability.

Part V

Conclusion

Chapter 7

Contributions

Game theory for security has focused only the problem of tactical planning; namely how to best deploy a set of fixed security resources. Inspired by the practical uses of hierarchical planning, in this thesis I formalize the different levels of planning for security games, and provide analysis into both the complexity of these problems as well as when they are most impactful to consider. I introduce two new types of problem *strategic* and *operational* planning which look at higher level design problem in security games, such as choosing a set of resources subject to budgetary constraints as well as lower level implementability challenges, such as ensuring that not only do strategies adhere to usability constraints such as staff schedules but are computed in such a way that they optimize the ease of implementation.

First this thesis introduces the *strategic* planning problem, where I provide a formal model for this problem with respect to security games where it becomes necessary to optimize over both the strategic and tactical planning problems simultaneously. I provide hardness results on the corresponding problem, as well as an efficient algorithm for solving this problem, which uses hierarchical abstractions of the tactical problem in order to efficiently search through the space of possible design choices.

Second I looks at the idea of resource heterogeneity in security games as the type of tactical planning problem which make strategic planning necessary. I focus in particular on a source of heterogeneity that can be found in all domains, resource effectiveness and I

show that solving games with this type of heterogeneity is hard. While imperfect resource effectiveness has been studied in the past, heterogenous in this area i.e. having resources with varying degrees of effectiveness does not frequently appear. In this thesis I look at one of the few games which models this type of heterogeneity and address a significant limitation which allows the defender to design more adaptive strategies for deploying these resources. I provide a novel framework based on linear decision rule approximations of the optimal strategy as well as robust reformulations and constraint randomization for efficiently computing solutions to this large scale problem. The solution approach not only provides very little loss in optimality but is also scalable, making it suitable for use in real world settings.

Third, motivated by the continued adoption of these security game models in the real world, I address usability concerns by looking at the *operational* planning problem. This thesis provides the first formal definition of usability for security games by introducing a new solution concept known as *operationalizable strategies* which are mixed strategies with small supports. This is a significant step towards ensuring the continued use of security games in the field as it provides a framework for address challenges in the day to day implementations of the mixed strategy solutions to these games. To this end, this thesis also provides a efficient algorithm for computing ϵ-optimal equilibria to these games as well as a scalable heuristic. This work also provides a cost benefit analysis into the trade-off in having operationalizable strategies and quantifies this by introducing the *Price of Usability* which measures the utility loss in operationalizable strategies compared to the full mixed strategy solutions.

Lastly, in the final chapter of this thesis I combine all three levels of planning into a single framework, looking at simultanously optimizing over strategic level planning decision

on how to schedule and form teams of heterogenous resources, how to tactically deploy these resources as well as ensuring that the mixed strategy solutions are operationalizable. I show that the *Price of Usability*, i.e., the loss in optimality to obtain a strategy that is easier to operationalize, is typically not high with the solution approaches provided in this work.

7.1 Future Directions

Overall this thesis opens up a large avenue for future work as it introduces two new types of planning problems to security games, that of strategic and operational planning. While this work addresses an important aspect of *strategic planning* namely, choosing among a set of heterogenous resources, there are other strategic design problems which can appear in security games. Problems where the defender has the ability to choose the payoff structure or alter the topology of the game are also strategic level planning problems that have yet to be considered in the literature.

Additionally, addressing uncertainty in the strategic planning problem will be an important direction for future work as it may be the case that there is uncertainty in the availability of different resources, their costs or in the budget available for investment. This is particularly relevant in the multistage setting where we need to make decisions now based uncertain information about how we may invest in the future. This is a challenging problem to address as now both the tactical and strategic plans need to be policies.

In the domain of tactical planning there are still many games which do not handle heterogenous resources and so an important direction for future work is developing models

which do not assume homogenous resources as well as more techniques for dealing with heterogeneity in resource types.

Lastly with respect to operational planning this thesis presents the idea of *k-operationalizable* mixed strategies which limit the support size of strategies to be of size k. This introduces a new design parameter k which allows the defender to tune the tradeoff between usability and utility. Indeed, an avenue for future work would be optimizing this choice of k in order to more efficiently tune this tradeoff and to ensure that the price of usability is not unnecessarily high.

Reference List

[AAAE, 2014] AAAE (2014). Transportation security policy. Technical report, American Association of Airport Executives.

[Agmon and Stone, 2012] Agmon, N. and Stone, P. (2012). Leading ad hoc agents in joint action settings with multiple teammates. In *AAMAS*, pages 341–348.

[Al-Mashari et al., 2003] Al-Mashari, M., Al-Mudimigh, A., and Zairi, M. (2003). Enterprise resource planning: A taxonomy of critical factors. *European Journal of Operational Research*, 146:352–364.

[Alpern and Gal, 2003] Alpern, S. and Gal, S. (2003). The theory of search games and rendezvous. *Springer*.

[Arieli and Babichenko, 2017] Arieli, I. and Babichenko, Y. (2017). Simple approximate equilibria in games with many players. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, pages 681–691. ACM.

[AVG,] AVG. https://www.facebook.com/Alliance-Voahary-Gasy-182047391868929/timeline/.

[Balcan et al., 2015] Balcan, M.-F., Blum, A., Haghtalab, N., and Procaccia, A. D. (2015). Commitment without regrets: Online learning in stackelberg security games. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 61–78. ACM.

[Ballou, 1973] Ballou, R. (1973). *Business Logistics Management*. Prentice-Hall international series in management. Prentice-Hall.

[Barrett, 2015] Barrett, D. (2015). U.s. suspects hackers in china breached about four (4) million people's records. *Wall Street Journal*.

[Basilico et al., 2009] Basilico, N., Gatti, N., and Amigoni, F. (2009). Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS*.

[Ben-Tal et al., 2004] Ben-Tal, A., Goryashko, A., Guslitzer, E., and Nemirovski, A. (2004). Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376.

[Biernat et al., 2003] Biernat, M., Kobrynowicz, D., and Weber, D. L. (2003). Stereotypes and shifting standards: Some paradoxical effects of cognitive load. *Journal of Applied Social Psychology*, 33(10):2060–2079.

[Bilge Bilgen, 2004] Bilge Bilgen, I. O. (2004). Strategic tactical and operational production-distribution models: a review. *International Journal of Technology Management*, 28:151–171.

[Birge and Louveaux, 1997] Birge, J. R. and Louveaux, F. (1997). *Introduction to stochastic programming*. Springer series in operations research. Springer, New York.

[Borders and Prakash, 2004] Borders, K. and Prakash, A. (2004). Web tap: detecting covert web traffic. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 110–120. ACM.

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

[Brown et al., 2014] Brown, M., Saisubramanian, S., Varakantham, P., and Tambe, M. (2014). Streets: Game-theoretic traffic patrolling with exploration and exploitation. In *AAAI*.

[Brown et al., 2016] Brown, M., Sinha, A., Schlenker, A., and Tambe, M. (2016). One size does not fit all: A game-theoretic approach for dynamically and effectively screening for threats. In *AAAI*, pages 425–431.

[Burns et al., 2015] Burns, W. J., Dillon-Merrill, R., and John, D. R. (2015). Dynamic aviation risk management system (darms): A proof of concept study examining the role of multi-attribute utility. Technical report, Department of Homeland Security.

[Calinescu et al., 2007] Calinescu, G., Chekuri, C., Pál, M., and Vondrák, J. (2007). Maximizing a submodular set function subject to a matroid constraint (extended abstract). In *Proceedings of the 12th International Conference on Integer Programming and Combinatorial Optimization*, IPCO '07, pages 182–196, Berlin, Heidelberg. Springer-Verlag.

[Campi and Garatti, 2008] Campi, M. C. and Garatti, S. (2008). The exact feasibility of randomized solutions of uncertain convex programs. *SIAM Journal on Optimization*, 19(3):1211–1230.

[Cho et al., 2013] Cho, J.-H., Chen, I.-R., Wang, Y., Chan, K. S., and Swami, A. (2013). Multi-objective optimization for trustworthy tactical networks: A survey and insights. Technical report, DTIC Document.

[Cooper and Sweller, 1987] Cooper, G. and Sweller, J. (1987). Effects of schema acquisition and rule automation on mathematical problem-solving transfer. *Journal of Educational Psychology*, pages 347–362.

[Dhital et al., 2015] Dhital, N., Vololomboahangy, R. R., and Khasa, D. P. (2015). Issues and challenges of forest governance in madagascar. *Canadian Journal of Development Studies / Revue canadienne d'tudes du dveloppement*, 36(1):38–56.

[Eppstein and Goodrich, 2008] Eppstein, D. and Goodrich, M. T. (2008). Studying (non-planar) road networks through an algorithmic lens. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, page 16. ACM.

[Fang et al., 2013a] Fang, F., Jiang, A. X., and Tambe, M. (2013a). Optimal patrol strategy for protecting moving targets with multiple mobile resources. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[Fang et al., 2013b] Fang, F., Jiang, A. X., and Tambe, M. (2013b). Optimal patrol strategy for protecting moving targets with multiple mobile resources. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 957–964. International Foundation for Autonomous Agents and Multiagent Systems.

[Fang et al., 2017] Fang, F., Nguyen, T. H., Pickles, R., Lam, W. Y., Clements, G. R., An, B., Singh, A., Schwedock, B. C., Tambe, M., and Lemieux, A. (2017). Paws-a deployed game-theoretic application to combat poaching. *AI Magazine*, 38(1):23.

[Fang et al., 2016] Fang, F., Nguyen, T. H., Pickles, R., Lam, W. Y., Clements, G. R., An, B., Singh, A., Tambe, M., and Lemieux, A. (2016). Deploying paws: Field optimization of the protection assistant for wildlife security. In *Proceedings of the Twenty-Eighth Innovative Applications of Artificial Intelligence Conference*.

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

[Farahanim Misni, 2017] Farahanim Misni, L. S. L. (2017). A review on strategic, tactical and operational decision planning in reverse logistics of green supply chain network design. *Journal of Computer and Communications*, 5:83–104.

[FarnHam, 2013] FarnHam, G. (2013). Detecting dns tunneling. Technical report, SANS Institute InfoSec Reading Room.

[Feige, 1998] Feige, U. (1998). A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652.

[Feng et al., 2017] Feng, X., Zheng, Z., Mohapatra, P., and Cansever, D. (2017). A stackelberg game and markov modeling of moving target defense. In *GameSec*.

[Fischetti and Monaci, 2012] Fischetti, M. and Monaci, M. (2012). Cutting plane versus compact formulations for uncertain (integer) linear programs. *Mathematical Programming Computation*, 4(3):239–273.

[Ford et al., 2014a] Ford, B., Kar, D., Delle Fave, F. M., Yang, R., and Tambe, M. (2014a). Paws: adaptive game-theoretic patrolling for wildlife protection. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1641–1642. International Foundation for Autonomous Agents and Multiagent Systems.

[Ford et al., 2014b] Ford, B., Kar, D., Fave, F. M. D., Yang, R., and Tambe, M. (2014b). Paws: Adaptive game-theoretic patrolling for wildlife protection (demonstration). In *Thirteenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[Gal, 1980] Gal, S. (1980). Search games. *Academic Press*.

[Gaston and desJardins, 2005] Gaston, M. E. and desJardins, M. (2005). Agent-organized networks for dynamic team formation. In *AAMAS*, pages 230–237. ACM.

[Gerkey and Mataric, 2003] Gerkey, B. P. and Mataric, M. J. (2003). Multi-robot task allocation: Analyzing the complexity and optimality of key architectures. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 3, pages 3862–3868. IEEE.

[Gooding, 2016] Gooding, D. (2016). Cluster of megabreaches compromises a whopping 642 million passwords. *Arstechnica*. http://arstechnica.com/security/2016/05/cluster-of-megabreaches-compromise-a-whopping-642-million-passwords/.

[Hart et al., 2011] Hart, M., Manadhata, P., and Johnson, R. (2011). Text classification for data loss prevention. In *Proceedings of the 11th International Conference on Privacy Enhancing Technologies*, PETS'11.

[Haskell et al., 2014a] Haskell, W., Kar, D., Fang, F., Tambe, M., Cheung, S., and Denicola, E. (2014a). Robust protection of fisheries with compass. In *IAAI*.

[Haskell et al., 2014b] Haskell, W. B., Kar, D., Fang, F., Tambe, M., Cheung, S., and Denicola, L. E. (2014b). Robust protection of fisheries with compass. In *IAAI*.

[Hogg, 2007] Hogg, N. M. (2007). Measuring cognitive load. In *Handbook of Research on Electronic Surveys and Measurements*, pages 188–194. Idea Group Inc.

[Hota et al., 2016] Hota, A. R., Clements, A. A., Sundaram, S., and Bagchi, S. (2016). Optimal and game-theoretic deployment of security investments in interdependent assets. In *International Conference on Decision and Game Theory for Security*, pages 101–113. Springer.

[Hunsberger and Grosz, 2000] Hunsberger, L. and Grosz, B. J. (2000). A combinatorial auction for collaborative planning. In *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*, pages 151–158. IEEE.

[Intel Security, 2015] Intel Security (2015). Grand theft data, data exfiltration study: Actors, tactics, and detection. Technical report. http://www.mcafee.com/us/resources/reports/rp-data-exfiltration.pdf.

[Isaacs, 1965] Isaacs, R. (1965). Differential games. *John Wiley and Sons*, pages 345–349.

[Jain et al., 2013] Jain, M., Conitzer, V., and Tambe, M. (2013). Security scheduling for real-world networks. In *AAMAS*, pages 215–222.

[Jain et al., 2010a] Jain, M., Kardeş, E., Kiekintveld, C., Tambe, M., and Ordóñez, F. (2010a). Security games with arbitrary schedules: a branch and price approach. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 792–797. AAAI Press.

[Jain et al., 2011] Jain, M., Korzhyk, D., Vaněk, O., Conitzer, V., Pěchouček, M., and Tambe, M. (2011). A double oracle algorithm for zero-sum security games on graphs. In *AAMAS*, pages 327–334.

[Jain et al., 2012a] Jain, M., Leyton-Brown, K., and Tambe, M. (2012a). The deployment-to-saturation ratio in security games. In *Conference on Artificial Intelligence (AAAI)*.

[Jain et al., 2012b] Jain, M., Leyton-Brown, K., and Tambe, M. (2012b). The deployment-to-saturation ratio in security games. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 1362–1370. AAAI Press.

[Jain et al., 2010b] Jain, M., Tsai, J., Pita, J., Kiekintveld, C., Rathi, S., Tambe, M., and Ordóñez, F. (2010b). Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces*, 40(4):267–290.

[Jiang et al., 2013] Jiang, A. X., Procaccia, A. D., Qian, Y., Shah, N., and Tambe, M. (2013). Defender (mis) coordination in security games. In *IJCAI*, pages 220–226.

[Jim Breithaupt, 2014] Jim Breithaupt, M. S. M. (2014). *Information Security Principles of Success*. Information Security: Principles and Practices, 2nd Edition.

[Jung and Tambe, 2003] Jung, H. and Tambe, M. (2003). Performance models for large scale multiagent systems: using distributed pomdp building blocks. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 297–304. ACM.

[Korzhyk et al., 2010a] Korzhyk, D., Conitzer, V., and Parr, R. (2010a). Complexity of computing optimal Stackelberg strategies in security resource allocation games. In *Proceedings of the 24th AAAI conference on Artificial Intelligence (AAAI)*, pages 805–810.

[Korzhyk et al., 2010b] Korzhyk, D., Conitzer, V., and Parr, R. (2010b). Complexity of computing optimal stackelberg strategies in security resource allocation games.

[Kryo, 2014] Kryo (2014). Iodine.

[Leitmann, 1978] Leitmann, G. (1978). On generalized stackelberg strategies. 26:637–643.

[Lelarge and Bolot, 2008] Lelarge, M. and Bolot, J. (2008). A local mean field analysis of security investments in networks. In *Proceedings of the 3rd ACM International Workshop on Economics of networked systems*, pages 25–30.

[Liemhetcharat and Veloso, 2012] Liemhetcharat, S. and Veloso, M. (2012). Modeling and learning synergy for team formation with heterogeneous agents. In *AAMAS*, AAMAS '12, pages 365–374, Richland, SC.

[Lipton et al., 2003] Lipton, R. J., Markakis, E., and Mehta, A. (2003). Playing large games using simple strategies. In *Proceedings of the 4th ACM conference on Electronic commerce*, pages 36–41. ACM.

[Madani et al., 1999] Madani, O., Hanks, S., and Condon, A. (1999). On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, AAAI '99/IAAI '99, pages 541–548, Menlo Park, CA, USA. American Association for Artificial Intelligence.

[Marcolino et al., 2014] Marcolino, L. S., Xu, H., Jiang, A. X., Tambe, M., and Bowring, E. (2014). Give a hard problem to a diverse team: Exploring large action spaces. In *AAAI*.

[Matthews et al., 2012] Matthews, T., Ramchurn, S. D., and Chalkiadakis, G. (2012). Competing with humans at Fantasy Football: Team formation in large partially-observable domains. In *Proceedings of the 26th Conference of the Associations for the Advancement for Artificial Intelligence*, pages 1394–1400.

[Mc Carthy et al., 2016] Mc Carthy, S., Tambe, M., Kiekintveld, C., Gore, M. L., and Killion, A. (2016). Preventing illegal logging: Simultaneous optimization of resource teams and tactics for security. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 3880–3886. AAAI Press.

[McAfee,] McAfee. Data loss prevention. http://www.mcafee.com/us/products/total-protection-for-data-loss-prevention.aspx.

[McCarthy et al., 2016a] McCarthy, S., Sinha, A., Tambe, M., and Manadhata., P. (2016a). Data exfiltration detection and prevention: Virtually distributed pomdps for practically safer networks. In *Conference on Decision and Game Theory for Security (GameSec)*.

[McCarthy et al., 2016b] McCarthy, S., Tambe, M., Kiekintveld, C., Gore, M. L., and Killion, A. (2016b). Preventing illegal logging: Simultaneous optimization of resource teams and tactics for security. In *AAAI conference on Artificial Intelligence (AAAI)*.

[McCarthy et al., 2017] McCarthy, S. M., Vayanos, P., and Tambe, M. (2017). Staying ahead of the game: Adaptive robust optimization for dynamic allocation of threat screening resources.

[McMahan et al., 2003] McMahan, H. B., Gordon, G. J., and Blum, A. (2003). Planning in the presence of cost functions controlled by an adversary. In *In Proceedings of the Twentieth International Conference on Machine Learning*.

[Miller, 2016] Miller, T. (2016). Supply chain frameworks: A constant in the midst of change in supply chain management and logistics: Innovative strategies and practical solutions.

[Morton et al., 2007] Morton, D. P., Pan, F., and Saeger, K. J. (2007). Models for nuclear smuggling interdiction. *IIE Transactions*, 39(1):3–14.

[Nair and Tambe, 2005] Nair, R. and Tambe, M. (2005). Hybrid BDI-POMDP framework for multiagent teaming. *Journal of Artificial Intelligence Research*, 23(1):367–420.

[Nguyen et al., 2016a] Nguyen, T. H., Sinha, A., Gholami, S., Plumptre, A., Joppa, L., Tambe, M., Driciru, M., Wanyama, F., Rwetsiba, A., Critchlow, R., and Beale, C. (2016a). Capture: A new predictive anti-poaching tool for wildlife protection. In *15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[Nguyen et al., 2016b] Nguyen, T. H., Sinha, A., Gholami, S., Plumptre, A., Joppa, L., Tambe, M., Driciru, M., Wanyama, F., Rwetsiba, A., Critchlow, R., et al. (2016b). Capture: A new predictive anti-poaching tool for wildlife protection. pages 767–775. AAMAS.

[Obata et al., 2011] Obata, T., Sugiyama, T., Hoki, K., and Ito, T. (2011). Consultation algorithm for Computer Shogi: Move decisions by majority. In *Computer and Games'10*, volume 6515 of *Lecture Notes in Computer Science*, pages 156–165. Springer.

[Okamoto et al., 2012] Okamoto, S., Hazon, N., and Sycara, K. (2012). Solving non-zero sum multiagent network flow security games with attack costs. In *AAMAS*, pages 879–888.

[Omic et al., 2009] Omic, J., Orda, A., and Van Mieghem, P. (2009). Protecting against network infections: A game theoretic perspective. In *IEEE INFOCOM*, pages 1485–1493.

[Paas, 1992] Paas, F. (1992). Training strategies for attaining transfer of problem-solving skill in statistics: A cognitive-load approach. *Journal of Educational Psychology*, pages 429–434.

[Papadimitriou and Tsitsiklis, 1987] Papadimitriou, C. H. and Tsitsiklis, J. N. (1987). The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450.

[Paruchuri et al., 2007] Paruchuri, P., Pearce, J. P., Tambe, M., Ordonez, F., and Kraus, S. (2007). An efficient heuristic approach for security against multiple adversaries. In *Proceedings of the 6th conference on Autonomous Agents and Multiagent Systems*. ACM.

[Paxson et al., 2013] Paxson, V., Christodorescu, M., Javed, M., Rao, J., Sailer, R., Schales, D., Stoecklin, M. P., Thomas, K., Venema, W., and Weaver, N. (2013). Practical comprehensive bounds on surreptitious communication over dns. In *Proceedings of the 22Nd USENIX Conference on Security*, SEC'13, pages 17–32, Berkeley, CA, USA. USENIX Association.

[Pita et al., 2008a] Pita, J., Jain, M., Marecki, J., Ordóñez, F., Portway, C., Tambe, M., Western, C., Paruchuri, P., and Kraus, S. (2008a). Deployed armor protection: The application of a game theoretic model for security at the los angeles international airport. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track*, AAMAS '08, pages 125–132, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

[Pita et al., 2008b] Pita, J., Jain, M., Ordóñez, F., Portway, C., Tambe, M., Western, C., Paruchuri, P., and Kraus, S. (2008b). Armor security for los angeles international airport. In *Proceedings of the 23rd national conference on Artificial intelligence-Volume 3*, pages 1884–1885. AAAI Press.

[P.M. Ghare and Turner, 1971] P.M. Ghare, D. M. and Turner, W. (1971). Optimal interdiction policy for a flow network. *Naval Research Logistics Quarterly*.

[Preciado et al., 2014] Preciado, V. M., Zargham, M., Enyioha, C., Jadbabaie, A., and Pappas, G. J. (2014). Optimal resource allocation for network protection against spreading processes. *IEEE Transactions on Control of Network Systems*, 1(1):99–108.

[Rascagneres, 2016] Rascagneres, P. (2016). New frameworkpos variant exfiltrates data via dns requests. https://blog.gdatasoftware.com/2014/10/23942-new-frameworkpos-variant-exfiltrates-data-via-dns-requests.

[Roy et al., 2012] Roy, S., Xue, M., and Das, S. (2012). Security and discoverability of spread dynamics in cyber-physical networks. *IEEE Transactions on Parallel and Distributed Systems*, 23(9):1694–1707.

[Schlenker et al., 2016] Schlenker, A., Brown, M., Sinha, A., and Tambe, M. (2016). Get me to my gate on time: Efficiently solving general-sum bayesian threat screening games. In *European Conference on AI (ECAI)*.

[Schlenker et al.,] Schlenker, A., Xu, H., Guirguis, M., Kiekintveld, C., Sinha, A., Tambe, M., Sonya, S., Balderas, D., and Dunstatter, N. Don't bury your head in warnings: A game-theoretic approach for intelligent allocation of cyber-security alerts. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*.

[Seth Bromberger, 2011] Seth Bromberger, Co-Principal Investigator, N. (2011). Dns as a covert channel within protected networks. Technical Report WP2011-01-01, National Electric Sector Cyber Security Organization.

[Shieh et al., 2012a] Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., and Meyer, G. (2012a). Protect: A deployed game theoretic system to protect the ports of the united states. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pages 13–20, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

[Shieh et al., 2012b] Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., and Meyer, G. (2012b). Protect: A deployed game theoretic system to protect the ports of the united states. In *AAMAS*.

[Shieh et al., 2013] Shieh, E., Jain, M., Jiang, A. X., and Tambe, M. (2013). Efficiently solving joint activity based security games. In *IJCAI*, pages 346–352. AAAI Press.

[Shieh et al., 2014] Shieh, E. A., Jiang, A. X., Yadav, A., Varakantham, P., and Tambe, M. (2014). Unleashing dec-mdps in security games: Enabling effective defender teamwork. In *ECAI*.

[Sidel, 2014] Sidel, R. (2014). Home depot's 56 million card breach bigger than target's. *The Wall Street Journal*. http://www.wsj.com/articles/home-depot-breach-bigger-than-targets-1411073571.

[Smith, 2007] Smith, T. (2007). *Probabilistic Planning for Robotic Exploration*. PhD thesis, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

[Sommer and Paxson, 2010] Sommer, R. and Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 305–316. IEEE.

[Stengel and Zamir, 2004] Stengel, B. V. and Zamir, S. (2004). Leadership with commitment to mixed strategies. Technical report.

[Symantec, 2017] Symantec (2017). Data Loss Prevention & Protection. https://www.symantec.com/products/information-protection/data-loss-prevention.

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

[Thomas F. Allnutt, 2013] Thomas F. Allnutt, Gregory P. Asner, C. D. G. G. V. N. P. (2013). Mapping recent deforestation and forest disturbance in northeastern madagascar. *Tropical Conservation Science*, 6(1):1–15.

[Trend Labs, 2013] Trend Labs (2013). Data exfiltration: How do threat actors steal your data? *Trend Micro Incorporated. A TrendLabs Security in Context Paper.* http://about-threats.trendmicro.com/cloud-content/us/ent-primers/pdf/how_do_threat_actors_steal_your_data.pdf.

[Tsai et al., 2009] Tsai, J., Rathi, S., Kiekintveld, C., Ordonez, F., and Tambe, M. (2009). Iris - a tool for strategic security allocation in transportation networks. In *The Eighth International Conference on Autonomous Agents and Multiagent Systems - Industry Track*.

[UNFCCC, 2016] UNFCCC (2016). Inclusion of the populations of madagascar in appendix ii, and limited to logs, sawn wood and veneer sheets by an annotation. In *Convention on International Trade in Endangered Species of Wild Fauna and Flora COP16 Prop 58*.

[Vaněk et al., 2012] Vaněk, O., Yin, Z., Jain, M., Bošanskỳ, B., Tambe, M., and Pěchouček, M. (2012). Game-theoretic resource allocation for malicious packet detection in computer networks. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 905–912. International Foundation for Autonomous Agents and Multiagent Systems.

[Varakantham et al., 2009] Varakantham, P., young Kwak, J., Taylor, M., Marecki, J., Scerri, P., and Tambe, M. (2009). Exploiting coordination locales in distributed pomdps via social model shaping.

[Vayanos et al., 2012] Vayanos, P., Kuhn, D., and Rustem, B. (2012). A constraint sampling approach for multi-stage robust optimization. *Automatica*, 48(3):459–471.

[Velagapudi et al., 2011] Velagapudi, P., Varakantham, P., Sycara, K., and Scerri, P. (2011). Distributed model shaping for scaling to decentralized pomdps with hundreds of agents. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 955–962. International Foundation for Autonomous Agents and Multiagent Systems.

[Wood, 1993] Wood, R. (1993). Deterministic network interdiction. *Mathematical and Computer Modelling*.

[WWF, 2015] WWF (2015). http://wwf.panda.org/about_our_earth/deforestation/deforestation_causes/illegal_logging.

[Xu, 2016] Xu, H. (2016). The mysteries of security games: Equilibrium computation becomes combinatorial algorithm design. *CoRR*, abs/1603.02377.

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

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

[Yin et al., 2015] Yin, Y., Xu, H., Gain, J., An, B., and Jiang, A. X. (2015). Computing optimal mixed strategies for security games with dynamic payoffs. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 681–687. AAAI Press.

[Yin et al., 2012] Yin, Z., Jiang, A. X., Johnson, M. P., Tambe, M., Kiekintveld, C., Leyton-Brown, K., Sandholm, T., and Sullivan, J. P. (2012). Trusts: scheduling randomized patrols for fare inspection in transit systems. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 2348–2355. AAAI Press.

[Zhu and Basar, 2015] Zhu, Q. and Basar, T. (2015). Game-theoretic methods for robustness, security, and resilience of cyberphysical control systems: Games-in-games principle for optimal cross-layer resilient control systems. *Control Systems, IEEE*, 35(1):46–65.