GAMES FOR THE OPTIMAL DEPLOYMENT OF SECURITY FORCES

Corine Maartje Laan

GAMES FOR THE OPTIMAL DEPLOYMENT OF SECURITY FORCES

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the Rector Magnificus,
prof.dr. T.T.M. Palstra,
on account of the decision of the Doctorate Board,
to be publicly defended
on Friday the 25$^{\text{th}}$ of January 2019 at 14.45 hours

by

Corine Maartje Laan

born on the 11$^{\text{th}}$ of June 1990
in Amsterdam, the Netherlands

This dissertation has been approved by:

Supervisor:
prof.dr. R.J. Boucherie
Co-supervisors:
dr. A.I. Barros
prof.dr. H. Monsuur

Ph.D. thesis, University of Twente, Enschede, the Netherlands
Digital Society Institute (No. 19-002, ISSN 2589-7721)

This PhD research was sponsored by TNO under a grant of the Netherlands Ministry of Defense, in cooperation with the Netherlands Defense Academy and the University of Twente.

Cover design: P.J. de Vries, Den Helder, the Netherlands
Printed by: Ipskamp, Enschede, the Netherlands
ISBN 978-90-365-4700-0
DOI 10.3990/1.9789036547000

Dissertation committee

Chairman & secretary:	prof.dr. J.N. Kok *University of Twente, Enschede, the Netherlands*
Supervisor:	prof.dr. R.J. Boucherie *University of Twente, Enschede, the Netherlands*
Co-supervisors:	dr. A.I. Barros *TNO, The Hague, the Netherlands* prof.dr. H. Monsuur *Netherlands Defense Academy, Den Helder, the Netherlands*
Members :	dr.ir. F. Bolderheij *Netherlands Defense Academy, Den Helder, the Netherlands* prof.dr. H.J.M Hamers *Tilburg University, Tilburg, The Netherlands* prof.dr. M.I.A. Stoelinga *University of Twente, Enschede, the Netherlands* prof.dr. M. Tambe *University of Southern California, Los Angeles, United States* dr. J.B. Timmer *University of Twente, Enschede, the Netherlands* prof.dr. M.J. Uetz *University of Twente, Enschede, the Netherlands*

Contents

CHAPTER 1

Introduction

1.1 Motivation

Physical security is a pressing issue as activities of intelligent adversaries targeting infrastructures and high value assets yield harmful effects to today's society. For instance, natural reserves are often victim of illegal fishing or poaching [97, 106]; high value assets like airports and public markets [63] need to be protected against adversary attacks and national borders need to be monitored in order to regulate the movement of people and goods [5, 116]. Although these security problems are quite different, they all involve the need to use the often scarce security resources in the most efficient way taking into account intelligent adversaries. In the literature, such security problems have received increasing attention and some of them are tackled using mathematical modeling (e.g., [3, 34, 113, 125, 127]).

In many of these papers, game theory is used to model the interaction between the security forces and the adversary. Although several of these papers take uncertainty in the adversary type and strategies into account, most studies consider deterministic input parameters. We present new modeling approaches for the efficient use of the available security resources that combine two essential elements: adaptivity of adversary behavior and uncertainty.

When protecting a certain area such as the sea, intelligent adversaries observe the security forces to find out their strategies, for example, where they are patrolling. Moreover, the advances in communication enable adversaries more than ever to communicate with each other and obtain information about the strategy of the security forces. Thus, the adversary is able to predict and react on the security forces' actions, in a cat and mouse game. Therefore, it is important to take into account the adaptive behavior of the adversary when developing models. We tackle this adaptive behavior by incorporating game theory into the modeling.

In practice, the behavior of adversaries is often unknown: when and where they will attempt to attack. Additionally, there can be uncertainty about the performance of the own security forces. For instance, the sensors of the security forces may not always be able to detect or produce false detections. This results in uncertainty about the adversary's position and thus the location of a possible attack. The environment might be subject to uncertainty, for example when weather conditions and seasonal fluctuations yield potential changes on the preferable attacking location. Therefore, when modeling security problems, uncertainty needs to be taken into account explic-

itly to derive realistic models that better mimic reality. We will address uncertainty explicitly by using several stochastic models.

In this thesis, we address the optimal deployment of (scarce) security forces taking into account the adaptive behavior of the adversary and the uncertainty that arises in these problems by combining game theory and stochastic modeling.

1.2 Background

In this section, we provide the mathematical concepts used in this thesis. First we give an introduction of game theory and discuss different game types used in security. We also give a short introduction about other techniques used in this thesis.

1.2.1 Game theory in the security domain

Many real world problems can be modeled using game theory. In this thesis, we apply game theoretical models to various security problems, resulting in agent-intruder games. Game theory provides a framework to model situations in which two or more players have some interaction [96]. All players compete over the value of the game, which depends on the actions of all players and each player commits to a strategy in order to optimize his or her payoff. Game theoretical models are used to analyze optimal strategies and the most likely outcomes. We only consider non-cooperative games, which means that all players commit to a strategy individually without cooperating with other players. By modeling the security forces (agent) and the adversary (intruder) as separate players, the adaptive behavior of the intruder is taken into account. In the rest of this thesis, we will refer to these games as agent-intruder games. The agent represents the security forces such as the coast guard or airport patrols and the intruder represents the adversary such as terrorists, enemy submarines, illegal fishermen or smugglers.

In a security game, both the agent and intruder can choose different actions resulting in the game value. The challenge is to find optimal strategies for the agent. A strategy for a player is a plan that gives in each situation the actions that have to be played. A best response strategy is an optimal strategy given the strategies of all other players. In general, we are searching for an equilibrium solution, which is a combination of strategies for all players where none of the players have the incentive to change strategies. So, in an equilibrium, all players play a best response strategy. There are several concepts available to analyze equilibrium strategies. In this thesis, we mainly focus on finding Nash equilibria [104]. In a Nash equilibrium, all players commit to a strategy that cannot be improved by deviating from the equilibrium strategy, given that the other players do not deviate from their strategy. All players have multiple actions to choose from and in an optimal strategy, it is allowed to randomize over these actions. A strategy that randomizes over multiple actions is called a mixed strategy, while a strategy that only picks one actions is called a pure strategy.

Another equilibrium concept that is common for security games, originates from Stackelberg games. While for Nash equilibria, it is assumed that both players make a move at the same time, players move sequentially in Stackelberg games. In a Stackelberg game, the agent commits to a strategy and thereafter, the intruder decides on a best response to this strategy [104]. In contrast to a Nash equilibrium, the strategy of the intruder in a Stackelberg equilibrium is a pure strategy since the agent's strategy

is already known. Therefore, the game value for Stackelberg equilibria can be different from the game values for Nash equilibria. However, in this thesis, we mainly consider zero-sum games (in which the gain for one player equals the loss for the other player) and for these games, the game value and agent's strategy coincide [133].

In the following example, we give a basic security game to explain the different game elements.

Example 1.1 (Basic security game)**.** Consider a patrolling game on a part of the North Sea which can be divided into two areas A and B. Since this is a protected area, it is not allowed to fish in A and B. However, in both areas, there is a lot of fish available, so these are popular places for fishermen (intruder) to fish illegally. To prevent illegal fishing in both areas, the coast guard (agent) has one patrolling ship available. This ship is able to patrol and protect one area from illegal fishing each day. At the beginning of a day, the intruder chooses one area to fish. When the intruder fishes successfully, i.e., without being caught by the agent, he obtains a gain. Assume that fishing in B is better than in A: the gain of successfully fishing in area A equals 3 and for area B, the fisherman's gain equals 5. The gain for the intruder equals the loss for the agent if he is patrolling the area where the intruder is not fishing. However, when the agent is patrolling the area where the intruder is fishing, the intruder is caught and the gain for the agent equals 1 (which is also the loss for the intruder).

The game above can be described in a matrix displaying the actions and payoffs for all players. In this game, the matrix is given by:

		Intruder	
		fish in A	fish in B
Agent	patrol A	$1/-1$	$-5/5$
	patrol B	$-3/3$	$1/-1$

By analyzing this game, the optimal strategy for the agent (and the intruder) can be found. Intuitively, one could argue that patrolling area B will be better, since the loss for the agent is the highest in that area. However, when the agent always patrols B, the intruder will adapt his strategy to always fishing in A, guaranteeing a gain of 3 (and a loss of 3 for the agent). The optimal mixed strategy is to patrol area A with probability $\frac{2}{5}$ and area B with probability $\frac{3}{5}$. This results in an expected loss of $1\frac{2}{5}$ for the agent, which is better than a loss of 3. □

1.2.2 Different game types

In this thesis, we use different game theoretical models to analyze various security problems. We briefly introduce different types of games. Early work considering security problems, for example the protection of networks in [36, 129, 131] or searching for a moving target in [31], only approaches the problem from the agent's perspective. Thus, possible reactions of intruders to the agent's strategy are not taken into account. However, to model intelligent intruders who know of and react to the strategy of an agent, game theoretic models have been developed (e.g., [2, 4, 18, 118, 126]). For example, Washburn [126] introduces a two-person zero-sum interdiction game that explicitly models the interaction between agents and intruders.

An important class of security problems is network interdiction. Generally speaking, network interdiction involves two sets of players which compete over the value of

the network: the intruder and the agent. The intruder tries to optimize the value of the system, for example by (1) computing the shortest path between a source node and a sink node [14, 36, 59]; (2) maximizing the amount of flow through the network [18, 76, 113]; (3) maximizing the probability of completing a route [29, 92, 93, 101]. The agent attempts to intercept the intruder before the goal is achieved.

Another type of games are search or patrolling games in which the goal of the agent is to find a hidden intruder by patrolling an area. This area can be modeled as a graph: the intruder can attack one or multiple nodes in this graph while an agent is searching this graph to prevent the attack. An overview of models for this type of problems is given by Hohzaki [58]. It is possible that the intruder is hidden at a fixed node (e.g., [4, 78, 95]), or moves through the network (e.g., [55, 120]). Neuts [95] introduces a search game in which the intruder hides in one node, while the agent must search in a set of nodes.

The problem of protecting vulnerable targets from attackers using limited security resources manifests in many real world applications. To this end, Stackelberg security games are introduced (e.g., [9, 11, 69, 118, 132]). In a security game, an agent is usually protecting a set of targets that are threatened by one or multiple intruders. Many search, patrolling or interdiction games are also modeled as a Stackelberg security game.

1.2.3 Mathematical models

In this thesis, we address various security problems. To solve these problems, we use different mathematical models, which we briefly introduce in this section.

A matrix game as introduced in Example 1.1 can be solved using linear programming [104]. In this thesis, several variants and extensions of a linear programming formulation for a standard matrix games are used to solve the proposed models. We briefly describe this linear program.

Consider a zero-sum game between an agent and an intruder. The action set of the agent is A_A and the action set of the intruder is A_I. The payoff matrix of this game is M where m_{ij} is the payoff when actions i and j are played, $i \in A_A$, $j \in A_I$. A strategy for the agent is given by p, where p_i is the probability that the agent picks action i, and similarly, the strategy for the intruder is q. The optimal value of the game, where the agent is maximizing this payoff and the intruder is minimizing the payoff is found by:

$$
\begin{aligned}
V = \max_{p} \min_{q} \quad & p^T M q \\
\text{s.t.} \quad & \sum_{i \in A_A} p_i = 1, \\
& \sum_{j \in A_I} q_j = 1, \\
& p, q \geq 0,
\end{aligned}
$$

where the objective equals the game value and the constraints ensure that p and q follow a probability distribution. By using duality, this maxmin formulation can be rewritten as a single maximization problem which can be solved efficiently [104].

To model the uncertainty in the security environment, we use several stochastic models. We give a short description of the models used in this thesis.

Queueing theory Queueing theory is a mathematical framework to study the behavior of waiting lines [112]. Analyzing the arrival and service process of these models gives insight in the expected queue lengths, waiting times etc. In this thesis, we use a network of queues, where one queue represents a small area.

Approximate dynamic programming Approximate dynamic programming provides an algorithmic framework to solve large scale Markov decision processes (MDPs). Markov decision theory provides a framework for decision making [108]. A single player is sequentially making decisions in an environment where outcomes are uncertain but depend on these decisions. In an MDP, one reasons about optimal strategies depending on the possible actions, the state of the system and transition probability.

Stochastic game theory A stochastic game is an extension of an MDP concerning two or more players [96]. A stochastic game consists of the same elements as an MDP, but takes into account more players, and the outcomes of the game depend on the actions of all players. When developing an optimal strategy, each player does not only have to take into account the current state and the transition probabilities, but also the possible actions of the other players.

Next to these models, we use several concepts from probability theory, such as conditional expectations and Bayes's rule and probability constraints.

1.3 Thesis outline

In this thesis, we develop and analyze various models concerning the optimal deployment of security forces. To deal with the adaptive behavior of an intruder, we use game theoretical models to determine agent's optimal strategies. We apply different techniques of stochastic modeling, such as queueing theory, approximate dynamic programming and Bayesian beliefs, to take uncertainty into account and combine these techniques with game theory.

This thesis consists of three parts. In the first part, we discuss games which are played on a queueing network. By modeling the area on which the game takes place as a queueing network, stochastic arrivals and travel times can be taken into account. The second part of this thesis considers dynamic games where new information becomes available during the game. In practice, both the agent and intruder do not have complete information about the state of the system. However, if the game takes place over multiple time periods, new information becomes available. Therefore, we discuss such games and consider strategies which depend on this new information. In the third part of this thesis, we discuss games in which the agent's strategy space is restricted as in many situations, the agent's strategies have to satisfy extra conditions. For these situations, we introduce new models that model such conditions explicitly. A more detailed overview of all chapters is given below.

Part I

In **Chapter 2**, we introduce an interdiction game on a queueing network including multiple intruders and agents who have stochastic travel or service times. Game theory is used to model the interaction between the intruder and the agent. Queueing

theory models the dynamic flow and time-dependent interdictions in a stochastic environment. The strategies of the intruders and agents influence the queueing system. This approach enables the modeling of the flow of intruders and the timing of the actions of the agent. We show that there exists a unique optimal solution for these types of games. Moreover, we introduce analytical formulas and algorithmic approaches to find optimal solutions for special network structures.

In **Chapter 3**, we introduce a new game where multiple players route through a Jackson queueing network. Each player decides on an optimal routing strategy to optimize its own sojourn time. We consider two cases: the game with continuous strategy space where the players can distribute their arrival rate over a set of fixed routes and the game with discrete strategy space where each player is only allowed to pick a single route. We discuss the existence of a pure Nash equilibrium for several variants and describe an algorithmic approach to find such a Nash equilibrium.

Part II

In **Chapter 4**, we consider partially observable agent-intruder games (POAIGs). We deviate from the traditional stochastic game assumption that both players have full information about the position of the other. Instead, we consider the situation one encounters in reality, where players only have partial information about the others position. These problems, where both players do have partial observable information about the position of the intruder, can be modeled as a dynamic search game on a graph between security forces and an intruder. In this chapter, we prove the existence of ϵ-optimal strategies for POAIGs with infinite time horizon. We develop an approximation algorithm based on belief functions that can be used to find approximate solutions for these games. To prove existence of ϵ-optimal strategies for POAIGs with infinite time horizon, we use results obtained for POAIGs with finite time horizon. For POAIGs with finite time horizon we show that a solution framework, common to solve extensive form games, can also be used effectively. As security forces often are faced with partial information, (solving) POAIGs provides decision support for developing patrol strategies and determining optimal locations for scarce resources like sensors.

Chapter 5 describes anti-submarine warfare games where an enemy submarine (intruder) attempts to attack a high value unit and an agent is allocating frigates and helicopters to detect this intruder. We allow time dependent strategies for the agents in order to deal with moving high value units. We use two separate approaches for the anti-submarine operations. It is usually assumed that the location of the frigates is known to the intruder since they are easy to observe. We first model this as a game in which the frigate path for the complete time period is known to the intruder. In this case, both the agent and the intruder construct optimal strategies in advance, since no new information arrives. Second, we describe a model where the frigate's location becomes available during the game. So at the start of each time interval, the intruder gets new information about the frigate's position and can adjust his strategy to this information.

Part III

In **Chapter 6** and **Chapter 7**, we discuss security games with restrictions on the agent's strategy. The coast guard is responsible for patrolling the coastal waters. Patrolling strategies should be unpredictable, cover the entire region, and must satisfy

operational requirements for example on the frequency of visits to certain vulnerable parts of the region (cells). We develop a special security game dealing with the protection of a large area in which the agent's strategy set is restricted. This area consists of multiple cells that have to be protected during a fixed time period. The agent has to decide on a patrolling strategy, which is constrained by governmental requirements that establish a minimum number of visits for each cell. A static version of this model is discussed in **Chapter 6**, where a strategy for the complete time period is identified before the game starts. The requirements are modeled such that they are met with high probability by introducing a mathematical program with probability constraints. In **Chapter 7**, we consider a dynamic approach to the security game with restricted strategies in which the agent decides on his strategy for each day taking into account expected future rewards. This allows finding a more flexible strategy for the agent, where current payoffs and number of visits to each cell can be taken into account. By formulating this model as a stochastic game, the agent is able to adjust the strategy to the current situation and actions that already have been chosen in the past. We approximate optimal solutions of this game via an approximate dynamic programming approach adjusted to stochastic games.

In **Chapter 8**, we discuss Stackelberg security games with a large number of pure strategies for the agent. An optimal mixed strategy typically randomizes over a large number of these strategies resulting in strategies that are not practical to implement. We propose a framework to construct strategies that are operationalizable by allowing only a limited number of pure strategies in a mixed strategy. However, by restricting the strategy space and allowing only strategies with a small support size, the solution quality might decrease. To investigate the impact of this restriction, we introduce the price of usability, which measures the ratio between the optimal solution and the operationalizable solution. The concept of operationalizable strategies is applied for threat screening games, a special variant of a security game. For these games, we develop a heuristic approach to find operationalizable strategies efficiently and investigate the impact of these restrictions.

Finally, in **Chapter 9**, we give a general conclusion and provide some directions for futere research based on the findings of this thesis.

Part I

Queueing and game theory

CHAPTER 2

An interdiction game on a queueing network with multiple intruders

The results in this chapter were published in [75].

2.1 Introduction

Security forces are deployed to protect networks that are threatened by multiple intruders. To select the best deployment strategy, we analyze an interdiction game that considers multiple simultaneous threats. Intruders route through the network as regular customers, while agents arrive at specific nodes as negative customers.

In the field of network interdiction, a wide variety of models have been proposed. Wollmer [129] was one of the first authors to consider a network interdiction model on a network defined by a set of arcs and nodes. In this model, the agent can remove arcs from a network in order to minimize the maximum flow the intruder can obtain from a source node to a sink node. Several papers generalize this work by accounting for the agents resources [131], which they can use to remove arcs from the network. The resource cost for such an action depends on the arc itself. These problems are shown to be NP-complete by Wood [131], even when the costs are equal for all arcs.

Most of the literature focuses on deterministic network interdiction (e.g., [126, 129, 131]). However, many network properties, such as travel time or detection probability, are uncertain in practice. Cormican et al. [27] consider a max-flow interdiction model in which interdiction success is a random variable. Moreover, extensions are made in which arc capacities are also considered to be stochastic.

In this chapter, we introduce an interdiction game on a queueing network including multiple intruders and agents which have stochastic travel or service times. In literature, there is limited research combining queueing theory and game theory in the security domain. Wein and Atkinson [127] combine game theory, dynamic programming and queueing theory to intercept terrorists on their way to the city center. A game theoretic approach is used to determine the sensor configuration and to calculate the detecting probabilities. The outcome of the game then becomes input for the queueing model.

Our model is developed to find an optimal deployment strategy for the agents that inspect the network nodes, i.e. which nodes should be inspected more often than others. Intruders enter the network at a certain node modeled as a queue and, after

having received service, route through the network to their target node. The routing strategies of the intruders can be modeled in a fixed or probabilistic manner. In the case of fixed routing, upon arrival at the network, intruders select their complete route to the sink node. In the case of probabilistic routing, intruders decide their next step at each node according to a certain probability. At the same time, agents inspect nodes of the network to prevent the arrival of intruder at the target nodes. When an agent inspects a node in which an intruder is being served, the intruder is removed from the network. In this context, the value of the network can be represented by the throughput of the intruders. Multiple intruders and agents compete to maximize and minimize this value respectively.

To model the intruders and agents, we use the concept of negative customers, which is introduced by Gelenbe et al. [41]. These authors describe a network of single server queues that includes positive and negative customers. Positive customers join the queue with the intention of getting served and then leave the system. Upon arrival of a negative customer, a positive customer (if present) is removed from the queue. We construct a game on this network to find the optimal deployment strategy for the agents. These strategies are reduced to choosing arrival rates for inspecting the nodes of the network. The intruders are modeled as the positive customers of the network, and the agents as the negative customers.

Our approach of an interdiction game on a queueing network combines two areas of research: game theory and queueing theory. Game theory is used to model the interaction between the intruder and agent. Queueing theory models the dynamic flow and time-dependent interdictions in a stochastic environment. The strategies of the intruders and agents influence the queueing system. This approach enables the modeling of the flow of intruders and the timing of the actions of the agent. The network itself may represent a region that the intruder is required to traverse before it can reach its destination. The queues then have service times that correspond to the stochastic travel times. Alternatively, routes in the network may represent sequences of tasks an intruder must complete before it is able to reach its target node.

This chapter is organized as follows. In the next section, we introduce the problem for fixed routing and analyze the proposed interdiction game on a queueing network. In Section 2.3 we determine optimal strategies for this game and provide some examples. Next, in Section 2.4, we discuss the game with probabilistic routing and show that these games are closely related. Finally, in Section 2.5, we present conclusions and provide directions for future research.

2.2 Game on a network with negative customers

This section introduces an interdiction game on a queueing network with negative customers and fixed intruder routing. Each node in the network represents a queueing system in which the intruders (positive customers) are served by a single server according to a first-in-first-out service discipline. Intruders enter the network at the source node and travel through the network to the sink node. After service completion at a node, the intruder follows its route to another node in the network. If the intruder is not interdicted at some intermediate node (neither the sink nor the source node), he successfully reaches the sink node. Agents (negative customers) arrive at the network nodes to search for intruders. If the agent arrives at an empty node, he leaves the network immediately. If an agent arrives at a node and finds an intruder being served,

then he removes the intruder and leaves the network. Because handling an intruder requires extra effort and time, we assume that only the intruder in service is removed.

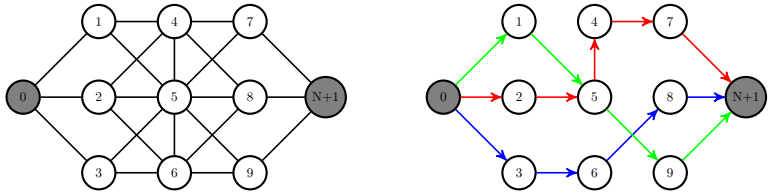
The players of the interdiction game, the intruders and the agents, are constrained by a budget. This limits the rates at which they arrive at the network: the agent has to determine arrival rates at nodes for inspecting the queueing systems and the intruder determines arrival rates at the routes. This repeated interplay results in probabilities of interdiction at nodes and ultimately yield intruder arrival rates at the sink node. The value of the game is therefore defined as the rate of intruders arriving at the sink.

In the following sections, we introduce a network with intruders and agents in which fixed routing of intruders is considered. After that, we give the game formulation and prove the existence of optimal strategies.

2.2.1 Network with fixed routing of intruders

Consider a queueing network with a source node 0, sink node $N+1$ and intermediate nodes $C = \{1, 2, ..., N\}$, on a connected and directed graph G. Intruders want to travel through the network undetected from source to sink, while agents try to intercept them at nodes in C. The source node 0 is linked to a non-empty set $C_S \subseteq C$ of start nodes, while there is a non-empty set of target nodes $C_T \subseteq C$ linked to the sink node $N+1$. There is no direct link between the source and sink, but it is possible that $C_S \cap C_T \neq \emptyset$. In addition, we assume that each node in C_S has just one incoming link (from the source); likewise, we assume that each node in C_T has just one outgoing link (to the sink). An example of such a network is shown in Figure 2.1a.

Given this queueing network, we consider the set of all routes from node 0 to node $N+1$, in which a route follows the links in the network. This set may be (countably) infinite, due to cycles in the network. We consider a finite subset K of the set of all routes without cycles. A route $k \in K$ (in which we do not take into account nodes 0 and $N+1$) is given by $r_k = [r(k,1), r(k,2), ..., r(k,N_k)]$, where $r(k,s)$ identifies the s-th node on route k and N_k is the length of route k. The set of nodes contained in route k is denoted by C_k. In Figure 2.1b an example network with three routes is given.



(a) Underlying graph. (b) Network with three routes.

Figure 2.1: Example graph G with $N = 9$.

Intruders arrive at the source of the network according to a Poisson process with rate Λ, and choose route k with probability p_k; i.e. the arrival rate of intruders following route k is given by $\lambda_k = p_k\Lambda$. Therefore, they enter at node $s \in C_S$ with arrival rate $\lambda_s = \sum_{k \in K, r(k,1)=s} \lambda_k$.

When intruders arrive at node i, they receive service or join the queue. The service time at node i is equal for all intruders and is exponential with rate $\mu_i > 0$. The service

time of each node is independent of the service time at other nodes.

Agents arrive at the network according to a Poisson process with rate Λ^- and select node i with probability p_i^-, such that they arrive at node $i \in C$ with rate $\lambda_i^- = p_i^- \Lambda^-$. Upon arrival of an agent, the intruder in service (if present) is removed from the node. If the agent arrives at an empty node, he immediately leaves the network.

Intruders routing through the network leave a node either because of service completion or because of interdiction while being served. Intruders are served at node i with exponential service rate μ_i and agents arrive independently according to a Poisson process with rate λ_i^-. This implies that intruders are interdicted with rate λ_i^-. Due to the memoryless property of the exponential distribution, the probability that an intruder leaves node i because of service completion corresponds to the probability that the service is completed before an agent arrives at node i:

$$\frac{\mu_i}{\mu_i + \lambda_i^-}, \tag{2.1}$$

and the probability that the intruder leaves node i (and is removed from the network) due to interdiction equals:

$$\frac{\lambda_i^-}{\mu_i + \lambda_i^-}.$$

These steady state probabilities are independent of the presence of other intruders in the network and of the time the intruders have spent in the queue. Route k is completed if an intruder completed service at each node of the route and reaches the sink node without being interdicted. Therefore, the probability that an intruder actually completes route k is given by:

$$\mathbb{P}(\text{intruder completes route } k) = \prod_{s=1}^{N_k} \frac{\mu_{r(k,s)}}{\mu_{r(k,s)} + \lambda_{r(k,s)}^-}. \tag{2.2}$$

2.2.2 Game description

To model the interaction between intruders and agents, we create an interdiction game on the queueing network described above. The intruders and agents compete over the value of this network, which is the arrival rate of intruders at the sink node, or equivalently, the sum of departure rates at nodes in C_T. This is a zero-sum game in which the intruders try to maximize their throughput by deciding on their routes, while the agents aim at minimizing this throughput by deciding on the inspection rates at nodes in C.

The intruders select their route by choosing λ_k for each route k, constrained by the total arrival rate Λ. Thus, the action set of the intruders given the set of routes K, is given by:

$$A_I = \left\{ \lambda \middle| \sum_{k \in K} \lambda_k = \Lambda, \lambda_k \geq 0,\ k \in K \right\}, \tag{2.3}$$

where $\lambda = (\lambda_k : k \in K)$.

The agents select the inspection rate, which is given by λ_i^- for all $i = 1, ..., N$, and the total rate is limited by a nonnegative interdiction budget Λ^-. So the action set of the agents is given by:

$$A_A = \left\{ \lambda^- \Big| \sum_{i=1}^{N} \lambda_i^- = \Lambda^-, \lambda_i^- \geq 0,\ i = 1, ..., N \right\}, \tag{2.4}$$

where $\lambda^- = (\lambda_1^-, ..., \lambda_N^-)$.

The payoff function of this game is the throughput (or arrival rate) of the intruders at the sink node, and is obtained by multiplying the arrival rate for each route k by the probability of completing the given route (see Equation (2.2)) and summing over all possible routes:

$$v(\lambda, \lambda^-) = \sum_{k \in K} \lambda_k \prod_{s=1}^{N_k} \frac{\mu_{r(k,s)}}{\mu_{r(k,s)} + \lambda_{r(k,s)}^-}. \tag{2.5}$$

2.2.3 Game analysis

In this section we analyze the interdiction game and prove the existence of pure optimal strategies.

Strategies for the intruders and agents are measures F and G defined for the sets A_I and A_A, such that $F(A_I) = 1$ and $G(A_A) = 1$. We define the expected payoff by:

$$\mathbb{E}(v(F, G)) = \int_{A_I \times A_A} v(\lambda, \lambda^-) d(F \times G).$$

A pure strategy for the intruder is a strategy F such that $F(\lambda) = 1$ for a particular $\lambda \in A_I$. This pure strategy then is denoted by λ, and is chosen with probability one. Likewise, pure strategies for the agent are represented by λ^-. The existence of pure strategies can be expressed by the following theorem:

Theorem 2.1. *Consider the interdiction game on a queueing network. The game has a saddle point λ^* and λ^{-*} in (optimal) pure strategies. Moreover, for the agent this strategy is unique. The value of the interdiction game is given by:*

$$v = \max_{\lambda} \min_{\lambda^-} v(\lambda, \lambda^-) = \min_{\lambda^-} \max_{\lambda} v(\lambda, \lambda^-).$$

Proof. Define the following two values:

$$v_I = \sup_F \inf_G \mathbb{E}(v(F, G)), \qquad v_{II} = \inf_G \sup_F \mathbb{E}(v(F, G)).$$

The payoff function $v(\lambda, \lambda^-)$ is continuous, and the action sets A_I and A_A are compact. Therefore, sup inf and inf sup may be replaced by max min and min max respectively, and $v_I = v_{II} = v$ and there exist optimal strategies (see Section IV.3 in [96]).

The existence of optimal pure strategies can be shown through the following function:

$$f(\lambda^-) = \prod_{i=1}^{N} \frac{\mu_i}{\mu_i + \lambda_i^-}.$$

The Hessian $\Delta^2 f(x)$ is positive definite, implying that $f(x)$ is strictly convex. The payoff function $v(\lambda, \lambda^-)$ is therefore strictly convex in λ^- for each λ. Moreover, $v(\lambda, \lambda^-)$ is a linear, and thus concave, function in λ for each λ^-. Thus, both the agent and the intruder have an optimal pure strategy and the value is given by v (see Section IV.4.1 in [96]). Because the payoff function is strictly convex in λ^-, the strategy for the agent is unique. ■

2.2.4 Optimization model

Given that optimal pure strategies exist, we formulate a minimization problem to find the optimal strategy of the agent. Let K be a fixed, finite set of routes from source to sink through the queueing network. The following optimization problem finds optimal strategies of the intruder and the agent:

$$v = \min_{\lambda^-} \max_{\lambda} \sum_{k \in K} \lambda_k \prod_{s=1}^{N_k} \frac{\mu_{r(k,s)}}{\mu_{r(k,s)} + \lambda^-_{r(k,s)}} \tag{2.6}$$

$$\text{s.t.} \sum_{j=1}^{N} \lambda_j^- = \Lambda^-, \tag{2.7}$$

$$\sum_{k=1}^{K} \lambda_k = \Lambda, \tag{2.8}$$

$$\lambda_i^-, \lambda_k \geq 0, \qquad i = 1, ..., N, k \in K. \tag{2.9}$$

Note that the value v is the arrival rate of intruders at the sink node $N+1$. In case $\Lambda = 1$, it also corresponds to the fraction of intruders that reach their destination, and thus the probability of reaching the sink node.

The optimal strategy of the agent can be found by solving the optimization problem as described in the next lemma.

Lemma 2.1. *For the interdiction game on a queueing network, the value of the game and the optimal strategy for the agent are found by solving the following convex minimization problem:*

$$v = \min_{\lambda^-} \quad w \tag{2.10}$$

$$s.t. \ \Lambda \prod_{s=1}^{N_k} \frac{\mu_{r(k,s)}}{\mu_{r(k,s)} + \lambda^-_{r(k,s)}} \leq w, \quad k \in K, \tag{2.11}$$

$$\sum_{j=1}^{N} \lambda_j^- = \Lambda^-, \tag{2.12}$$

$$\lambda_i^- \geq 0, \qquad i = 1, ..., N. \tag{2.13}$$

Proof. The probability of completing route k is given by Equation (2.2), so the throughput in the case where the intruder always chooses route k is $\Lambda \prod_{s=1}^{N_k} \frac{\mu_{r(k,s)}}{\mu_{r(k,s)} + \lambda^-_{r(k,s)}}$. Given any interdiction strategy λ^-, the worst case for the agent is when the intruder

chooses to assign his full budget Λ to the set of routes with maximal completion probability. The agent tries to minimize this worst case, which can be achieved by solving the non-linear program in (2.10)-(2.13). From the proof of Theorem 2.1, we know that Constraints (2.11) are convex in λ^-, so (2.10)-(2.13) yields a convex optimization problem. ■

Depending on the graph structure, the number of constraints in Lemma 2.1 can grow exponentially. This is certainly the case for a complete graph.

Note that w is the maximum payoff the intruders can obtain for any available route, given the choice of λ^- of the agents. In the following section, we solve this model for networks with special structures, such as networks with only parallel or only tandem nodes. These are the networks in which routes do not intersect. Because the payoff-function is continuous in λ^-, the probability of completing a specific route in these networks must be the same for each route. We also provide numerical results for networks with a general network structure.

2.3 Finding optimal strategies

In the previous section, we described an interdiction game in which intruders and agents compete over the throughput of the intruders. In this section, we derive analytical expressions and algorithms for finding optimal strategies, for three special cases. In these cases, we let K equal the set of all possible routes. Finally, we use the analytical expressions to speed up the solving process for general networks and provide numerical results.

2.3.1 Network of parallel nodes

Consider a network of parallel nodes as shown in Figure 2.2a. The length of each route k equals one. There are N possible routes such that $r_k = [k]$ for $k = 1, ..., N$. The payoff function of the game is given by:

$$v(\lambda, \lambda^-) = \sum_{k=1}^{N} \lambda_k \frac{\mu_k}{\mu_k + \lambda_k^-}.$$

The value and optimal strategies of this game are given in the following theorem.

Theorem 2.2. *Consider the interdiction game on a network of parallel nodes. For the agents, the unique optimal strategy λ^{-*} is given by:*

$$\lambda_i^{-*} = \frac{\mu_i}{\sum_{j=1}^{N} \mu_j} \Lambda^-, \quad \textit{for all } i = 1, ..., N. \tag{2.14}$$

The value of the game is:

$$v = \frac{\sum_{j=1}^{N} \mu_j}{\sum_{j=1}^{N} \mu_j + \Lambda^-} \Lambda. \tag{2.15}$$

Proof. According to Theorem 2.1, there exists an optimal pure strategy and the value is given by $v = \max_{\lambda} \min_{\lambda^-} v(\lambda, \lambda^-)$. Through Lemma 2.1, we know that optimal strategies for the agents can be found by solving:

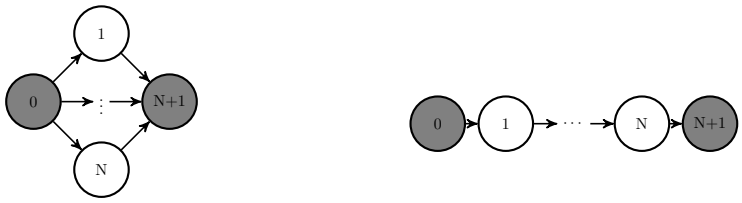
$$\begin{aligned} \min_{\lambda^-} \quad & w \\ \text{s.t. } & \Lambda \frac{\mu_i}{\mu_i + \lambda_i^-} \leq w, \quad i = 1, ..., N, \\ & \sum_{j=1}^{N} \lambda_j^- = \Lambda^-, \\ & \lambda_i^- \geq 0, \qquad\qquad i = 1, ..., N. \end{aligned} \tag{2.16}$$

Given this network of parallel nodes, the agent must ensure that the probability of completing a specific route will be the same for each route. Thus, for an optimal λ^{-*}:

$$v = \Lambda \frac{\mu_i}{\mu_i + \lambda_i^{-*}}, \quad i = 1, ..., N. \tag{2.17}$$

By combining (2.17) with the interdiction budget constraint $\sum_{j=1}^{N} \lambda_j^- = \Lambda^-$, we obtain the optimal strategy λ^{-*} and the value of the game. ■

Equation (2.14), shows that inspection rates increase with node service rates. Given Equation (2.15), it follows that the value of the game is dependent only upon the sum of the service rates μ_i and not upon how these rates are assigned to the nodes. Thus, from a game-theoretic point of view, a network of parallel nodes is equivalent to a single queue with service rate equal to the sum of service rates.



(a) Network of parallel nodes. (b) Network of tandem nodes.

Figure 2.2: Two networks for which an explicit value of the game can easily be derived.

2.3.2 Network of tandem nodes

Consider a network of tandem nodes as shown in Figure 2.2b. There is only one route with length N and rate Λ. Therefore, the value of the game only depends on the strategy of the agent. The payoff function of the game is given by:

$$v(\lambda^-) = \Lambda \prod_{i=1}^{N} \frac{\mu_i}{\mu_i + \lambda_i^-}. \tag{2.18}$$

For technical purposes, we introduce a relaxation of the optimization model described in Section 2.2.4. In this model, only the budget constraint (2.8) is taken into account, relaxing the non-negativity constraints (2.9). The value and optimal solutions of this relaxation model with the objective function (2.18) are given by the following lemma.

Lemma 2.2. *Consider the relaxation problem on a network of tandem nodes. The optimal solution λ^{-*} is given by:*

$$\lambda_i^{-*} = \frac{\Lambda^- + \sum_{j=1}^N \mu_j}{N} - \mu_i, \quad i = 1, ..., N, \tag{2.19}$$

and the value of this relaxation is:

$$v^r = \Lambda \prod_{i=1}^N \frac{N\mu_i}{\sum_{j=1}^N \mu_j + \Lambda^-}. \tag{2.20}$$

Moreover, if $\frac{\Lambda^- + \sum_{j=1}^N \mu_j}{N} \geq \max_j \mu_j$, the optimal solution and the value of the relaxation problem are equal to the optimal strategies and the value of the original interdiction game.

Proof. The value v^r of the relaxation can be found by solving the following optimization problem:

$$\begin{aligned} v^r = \min_{\lambda^-} \quad & \Lambda \prod_{i=1}^N \frac{\mu_i}{\mu_i + \lambda_i^-}, \\ \text{s.t.} \quad & \sum_{i=1}^N \lambda_i^- = \Lambda^-. \end{aligned}$$

In order to derive v^r, we use a Lagrangian approach. The Lagrangian of this problem is given by:

$$L(\lambda^-, \psi) = \Lambda \prod_{i=1}^N \frac{\mu_i}{\mu_i + \lambda_i^-} + \psi \left(\sum_{j=1}^N \lambda_j^- - \Lambda^- \right).$$

Taking the partial derivatives with respect to λ_i^- and ψ, and rewriting, enables the calculation of the optimal solution of the relaxation. If $\frac{\Lambda^- + \sum_{j=1}^N \mu_j}{N} \geq \max_j \mu_j$, then $\lambda_i^{-*} \geq 0$ for all $i = 1, ..., N$. In that case, it is also a feasible solution to the original game and v^r is an upper bound for the value v of the original game. Because there are fewer constraints in the relaxation model, v^r also gives a lower bound for v. Combining the lower and upper bound, gives $v^r = v$ and the resulting solution is also an optimal strategy for the original game. ■

Equation (2.19) shows that the inspection rate increases as the service rate decreases, contrary to the case for parallel nodes. This equation also suggests that if the service rate of a particular node i is very high, it is optimal to set $\lambda_i^- = 0$ before hand. To be more precise, suppose that $\frac{\Lambda^- + \sum_{j=1}^N \mu_j}{N} < \max_j \mu_j$. Then there is a node i such that $\lambda_i^- < 0$ and the value of the relaxation does not correspond to the value

of the original game. To find a feasible solution for the original interdiction game, we introduce an algorithm that, starting with the solution of the relaxation, sequentially removes nodes for which $\lambda_i^- < 0$. In every step of the algorithm, the state space is reduced by adjusting the value of λ_i^- that violates (2.9). By using this relaxation and iterative approach, we eventually find the optimal pure strategy for the agent for the original game.

Algorithm 1

Let C' be a subset of the set C, and $N' = |C'|$.

1: Set $C' = I$, and $N' = |C'|$.

2: Calculate for all $i \in C'$:

$$\lambda_i^- = \frac{\Lambda^- + \sum_{j \in C'} \mu_j}{N'} - \mu_i. \tag{2.21}$$

If $\lambda_i^- \geq 0$ for all $i \in C'$: STOP, λ^- is given by (2.21) and the value of the game is given by:

$$v = \Lambda \prod_{i \in C'} \frac{N' \mu_i}{\sum_{j \in C'} \mu_j + \Lambda^-}.$$

Else: Go to next step.

3: For all i such that $\lambda_i^- < 0$: Set $\lambda_i^- = 0$, remove i from C' and update N'. Return to step 2.

Theorem 2.3. *Algorithm 1 finds the optimal strategy for the agents and the value of the interdiction game on a network of tandem nodes.*

The proof of Theorem 2.3 can be found in Section 2.6.

2.3.3 Networks without intersecting routes

In this section, we consider networks in which the set of routes K is restricted to routes that do not intersect. An example of such a network with three routes is shown in Figure 2.3.

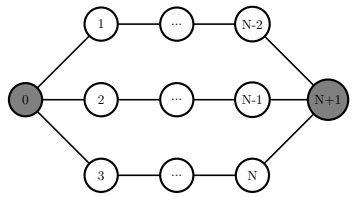


Figure 2.3: Network of parallel tandem nodes: the number of nodes per route may differ.

Consider a network of N nodes with routes K that do not intersect, in which route

k consists of N_k nodes. The value function of this game is given by:

$$v(\lambda, \lambda^-) = \sum_{k \in K} \lambda_k \prod_{s=1}^{N_k} \frac{\mu_{r(k,s)}}{\mu_{r(k,s)} + \lambda^-_{r(k,s)}}. \tag{2.22}$$

As before, we first consider the relaxation model such that $\lambda_i^- < 0$ is allowed. Λ_k^- is defined as the interdiction budget assigned by the agent to route k:

$$\begin{aligned} \Lambda_k^- &= \sum_{s=1}^{N_k} \lambda^-_{r(k,s)}, \quad k \in K, \\ \Lambda^- &= \sum_{k \in K} \Lambda_k^-. \end{aligned} \tag{2.23}$$

The optimal solution and the value of this relaxation are given by the following lemma.

Lemma 2.3. *Consider the relaxation problem with objective function* (2.22) *on a network without intersecting routes. The value v^r of this model can then be found by solving:*

$$\Lambda^- + \sum_{i=1}^{N} \mu_i = \sum_{k \in K} N_k \sqrt[N_k]{\frac{\Lambda \prod_{s=1}^{N_k} \mu_{r(k,s)}}{v^r}}.$$

Moreover, the budget assigned to route k in the optimal solution, is given by:

$$\Lambda_k^{-*} = N_k \sqrt[N_k]{\frac{\Lambda \prod_{s=1}^{N_k} \mu_{r(k,s)}}{v^r}} - \sum_{t=1}^{N_k} \mu_{r(k,t)}.$$

Proof. If we knew the interdiction budget Λ_k^-, Lemma 2.2 could be used to obtain the value of the relaxation, and its optimal budget assignment to individual nodes on route k. The throughput of intruders over route k is:

$$v_k^r = \lambda_k \prod_{s=1}^{N_k} \frac{N_k \mu_{r(k,s)}}{\sum_{t=1}^{N_k} \mu_{r(k,t)} + \Lambda_k^-}. \tag{2.24}$$

Therefore, similar to the approach followed in Lemma 2.1, the optimal solution and value v^r in this relaxation can be found by solving:

$$\begin{aligned} \min_{\Lambda_k^-} \quad & w \\ \text{s.t. } & \Lambda \prod_{s=1}^{N_k} \frac{N_k \mu_{r(k,s)}}{\sum_{t=1}^{N_k} \mu_{r(k,t)} + \Lambda_k^-} \leq w, \quad k \in K, \\ & \sum_{k \in K} \Lambda_k^- = \Lambda^-. \end{aligned} \tag{2.25}$$

Solving (2.25) yields the optimal strategy Λ^{-*} for the relaxation. As routes do not intersect, for an optimal Λ_k^{-*}:

$$v^r = \Lambda \prod_{s=1}^{N_k} \frac{N_k \mu_{r(k,s)}}{\sum_{t=1}^{N_k} \mu_{r(k,t)} + \Lambda_k^{-*}}, \quad \text{for all } k \in K, \tag{2.26}$$

implying:

$$\Lambda_k^{-*} = N_k \sqrt[N_k]{\frac{\Lambda \prod_{s=1}^{N_k} \mu_{r(k,s)}}{v^r}} - \sum_{t=1}^{N_k} \mu_{r(k,t)}. \tag{2.27}$$

Combining (2.23) and (2.27) yields:

$$\Lambda^- + \sum_{i=1}^{N} \mu_i = \sum_{k \in K} N_k \sqrt[N_k]{\frac{\Lambda \prod_{s=1}^{N_k} \mu_{r(k,s)}}{v^r}}. \tag{2.28}$$

The value v^r can be found by solving Equation (2.28) iteratively. ■

The optimal strategy is one in which the probability of completing a particular route, is the same for each possible route. It may happen that for some route k, $\frac{\Lambda_k^- + \sum_{s=1}^{N_k} \mu_j}{N} < \max_{j \in r_k} \mu_j$, in which case the value of the relaxation model is not necessarily equal to the value of the original game with inequality constraints. Therefore, we introduce an algorithm to find a feasible solution. The core of this algorithm is similar to Algorithm 1: set λ_i^- to zero if it violates the inequality constraints and recalculate optimal strategies for the relaxation without these nodes.

Algorithm 2

Let C' be a subset of the set C, let $C_k = \{i \in C | i \in r_k\}$ and C'_k a subset of C_k. Moreover, let $N' = |C'|$ and $N'_k = |C'_k|$.

1: Set $C' = C$, $N' = |C'|$, and $C'_k = C_k$, $N'_k = |C'_k|$ for all $k \in K$.

2: Obtain v^r from:

$$\Lambda^- + \sum_{i \in C'} \mu_i = \sum_{k \in K} \sqrt[N'_k]{\frac{\prod_{i \in C'_k} N'_k \mu_i}{v^r}}.$$

3: For all $k \in K$, let:

$$\Lambda_k^- = N'_k \sqrt[N'_k]{\frac{\Lambda \prod_{i \in C'_k} \mu_i}{v^r}} - \sum_{i \in C'_k} \mu_i.$$

4: For all $k \in K$ and for all $i \in C'_k$, let:

$$\lambda_i^- = \frac{\Lambda_k^- + \sum_{j \in C'_k} \mu_j}{N'_k} - \mu_i. \tag{2.29}$$

If $\lambda_i^- > 0$ for all $k = 1, ..., K$ and for all $i \in C'_k$: STOP, λ^- is given by (2.29) and the value of the game is given by v^r.

5: Else: Go to the next step

6: For all $k \in K$ and for all $i \in C'_k$:

7: If $\lambda_i^- \leq 0$ and $\mu_i = \max_{j \in C'_k} \mu_j$ $(i \in C'_k)$: Set $\lambda_i^- = 0$ and remove i from C' and C'_k. Then, go back to Step 2.

Theorem 2.4. *Algorithm 2 finds the optimal strategy for the agents and the value of the interdiction game on a network of parallel tandem nodes without intersections.*

The proof of Theorem 2.4 can be found in Section 2.6.

Remark 2.1. The algorithm can be more efficient by replacing Step 5 of the algorithm with:

- For all $k \in K$ and for all $i \in C'_k$:
  If $\lambda_i^- < 0$: Set $\lambda_i^{-*} = 0$ and remove i from C' and C'_k. Then, go back to Step 2.

Due to its length, a proof that the adjusted algorithm also finds an optimal solution is omitted.

2.3.4 General network

In the previous sections, we obtained analytical expressions and algorithms to find optimal strategies for special networks, which do not contain intersecting routes. In this section, we discuss the general network case.

The optimal strategy for the general network case is obtained using Lemma 2.1. The previously introduced results can be used to speed up the process of solving general networks. In particular, utilizing Lemma 2.2 may decrease the number of general network variables with equal service rates in the following way. Each route can be split into a set of intersection nodes C_k^I (nodes that are also part of another route) and, between these intersection nodes, segments of tandem nodes $C_k^T = C_k \backslash C_k^I$. Constraints (2.11) in Lemma 2.1 can then be rewritten as follows:

$$\Lambda \prod_{i \in C_k^I} \frac{\mu_i}{\mu_i + \lambda_i^-} \prod_{i \in C_k^T} \frac{\mu_i}{\mu_i + \lambda_i^-} \leq w, \quad \text{for all } k \in K.$$

Given the interdiction rates λ^- and a route k, the order of the nodes in this route has no impact on the game value. Therefore, route k can be seen as a sequence of intersection nodes C_k^I and one separate tandem queue with nodes C_k^T. Let $\tilde{\Lambda}_k^-$ be the total budget that is assigned to the tandem nodes in route k, i.e., $\tilde{\Lambda}_k^- = \sum_{i \in C_k^T} \mu_i$. If $\tilde{\Lambda}_k^-$ is known, it is optimal to divide this budget over the nodes using Lemma 2.2, as this can be seen as a separate tandem queue. So, by Lemma 2.2, the constraints can be replaced with the following constraints:

$$\Lambda \prod_{i \in C_k^I} \frac{\mu_i}{\mu_i + \lambda_i^-} \prod_{i \in C_k^T} \frac{|C_k^T| \mu_i}{\sum_{j \in C_k^T} \mu_j + \tilde{\Lambda}_k^-} \leq w, \quad \text{for all } k \in K. \tag{2.30}$$

Remark 2.2. Lemma 2.2 gives a value for the relaxation, which equals the value of the original game only if no negative interdiction rate is assigned to one of the nodes. This is always the case if all nodes have an equal service rate because nodes with equal service rates always have the same interdiction rate. The constraints in (2.30) can also be used to solve networks with unequal service rates. Then, by analogy with Algorithm 2, the nodes with a negative interdiction rate can be removed from the network and the resulting non-linear program must be solved again.

2.3.5 Numerical examples

We have developed an interdiction game with intruders and agents and derived optimal strategies. In this section, we first consider the computational efforts of our algorithms. Then, we present two illustrative examples.

Computational efforts

This section explores the computational efforts required to obtain the optimal strategy. To this end, Table 2.1 below presents the running times for randomly generated networks both for a direct implementation of Lemma 2.1 and invoking the structural results of Section 2.3.4 based on Lemma 2.2. For the results of Table 2.1, we constructed random routes in a network whose underlying graph is complete, and all nodes have service rate one. For each case, we generated ten random instances and show the average values and 95%-confidence intervals in Table 2.1. The average length of the routes equals the square root of the number of nodes, and in general it holds that if the number of routes is small, the number of intersection nodes is also small.

To find optimal strategies, we used CVX 2.1, a package for solving convex programs [28, 46], in Matlab version R2014b [84] on an Intel(R) Core(TM) i7 CPU, 2.4GHz, 8 GB of RAM. To this end, we reformulated Constraints (2.11) such that they comply with the ruleset of Disciplined Convex Programming (DCP) [47] as follows:

$$\Lambda \prod_{s=1}^{N_k} \mu_{r(k,s)} \prod_{s=1}^{N_k} \left(\mu_{r(k,s)} + \lambda^-_{r(k,s)}\right)^{-1} \leq w,$$

which can be rewritten as:

$$C \prod_{s=1}^{N_k} v^{-1}_{r(k,s)} \leq w,$$

where $C = \Lambda \prod_{s=1}^{N_k} \mu_{r(k,s)}$ is a constant and $v_{r(k,s)} = \mu_{r(k,s)} + \lambda^-_{r(k,s)}$. Invoking the function *prod_inv* from the CVX library, the reformulation of the convex program of Lemma 2.1 meets the requirements of the DCP ruleset [47]. With this formulation, CVX finds the optimal solution of the problem.

From Table 2.1, we observe that the running time for networks of reasonable size remains acceptable for practical purposes. The network structure exploited in Lemma 2.2 considerably reduces the running time for networks containing a relatively low number of routes.

Networks of parallel and tandem nodes

First, we compare a network of parallel nodes with a network of tandem nodes. Both networks consist of ten nodes with service rate one. The results are shown in Figure 2.4a. For a network with tandem nodes, the throughput is much lower than for the network with parallel nodes. This is an intuitive result because the intruder must be served at all nodes within a tandem node network, while in the network with parallel nodes, intruders are only required to complete service at one node.

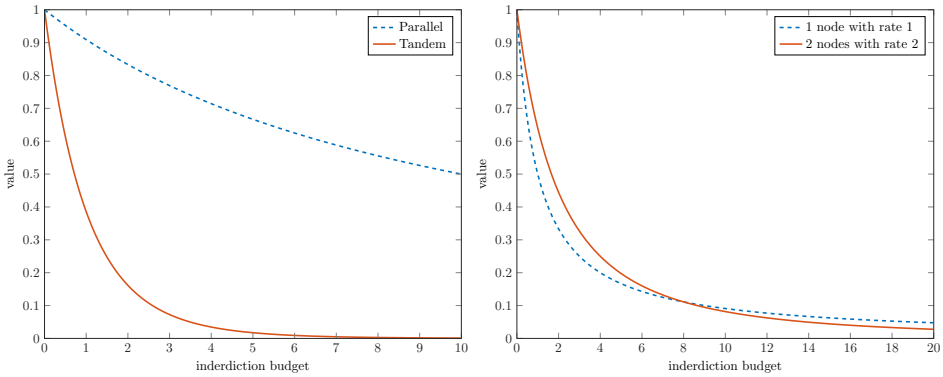
Second, we investigate whether it is better to design a network with one node or with multiple nodes, i.e. the optimal locations for protection against intruders. In

Table 2.1: Running times for solving Lemma 2.1 with and without implementation of Lemma 2.2.

Number of nodes	Number of routes	Λ	Running time with Lemma 2.2 (sec)	Running time without Lemma 2.2 (sec)	Game value
1000	10	5	2.44 ($\pm$ 0.13)	4.03 ($\pm$ 0.16)	0.38 ($\pm$ 0.02)
1000	50	5	13.66 ($\pm$ 2.99)	14.07 ($\pm$ 2.81)	0.69 ($\pm$ 0.02)
1000	100	5	27.39 ($\pm$ 1.21)	27.41 ($\pm$ 1.32)	0.78 ($\pm$ 0.01)
5000	10	10	4.17 ($\pm$ 0.46)	12.53 ($\pm$ 0.62)	0.20 ($\pm$ 0.03)
5000	50	10	25.13 ($\pm$ 1.10)	36.11 ($\pm$ 1.47)	0.56 ($\pm$ 0.02)
5000	100	10	66.95 ($\pm$ 5.10)	72.94 ($\pm$ 4.50)	0.68 ($\pm$ 0.01)
25000	10	20	8.81 ($\pm$ 1.79)	63.52 ($\pm$ 2.16)	0.06 ($\pm$ 0.02)
25000	50	20	55.66 ($\pm$ 3.97)	121.96 ($\pm$ 5.08)	0.39 ($\pm$ 0.02)
25000	100	20	273.56 ($\pm$ 19.42)	553.85 ($\pm$ 62.16)	0.54 ($\pm$ 0.01)

a network with parallel nodes, we see that the value of the game increases in the number of nodes because intruders can choose between multiple paths (see Theorem 2.2). Therefore, in order to obtain the same value in a network with multiple nodes, the service rate must be smaller in proportion to the number of nodes, e.g., the services rate must be halved if the number of nodes is doubled.

Now, consider a tandem network in which the intruders are required to complete service at all nodes. We compare one and two node cases. In the two node case the intruder is served twice as fast. Figure 2.4b shows that for a low interdiction budget, it is better to have one node, while for a high interdiction budget, most intruders are intercepted if multiple nodes are considered. These examples not only illustrate that our model can be used to determine optimal deployment strategies of the agents, but they may also help in the design of an effective network topology.



(a) Compare parallel and tandem nodes.

(b) Different network design.

Figure 2.4: Illustrative examples.

General network

Consider the network in Figure 2.5 with six intersecting routes $r_1, r_2, ..., r_6$. These routes have six intersection nodes $i_1, i_2, ..., i_6$ and 35 tandem nodes. For each node, the service rate equals one. We solved our model in Matlab for different values of Λ^-. The total arrival rate of the intruder Λ equals one. The value v and optimal strategies λ^- and $\tilde{\Lambda}^-$ for the agent are shown in Table 2.2. The rates for all intersection nodes are given by $\lambda^-_{i_1}, ..., \lambda^-_{i_6}$ and $\tilde{\Lambda}^-_{r_1}, ..., \tilde{\Lambda}^-_{r_6}$ are the rates for all tandem nodes of one route. The results are summarized in Table 2.2.

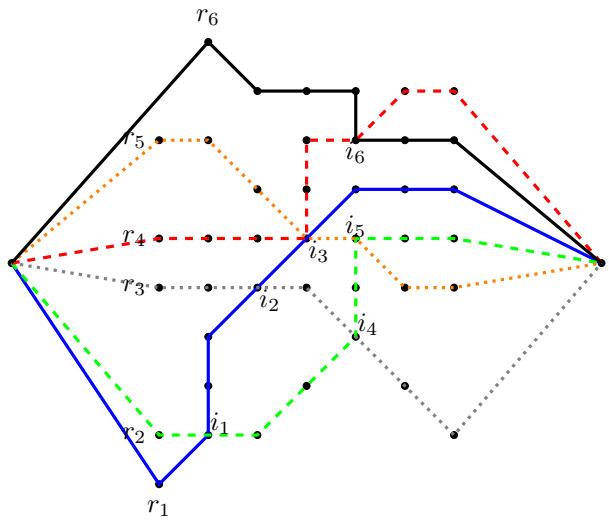


Figure 2.5: Example of a general network.

Table 2.2: Agent strategies for the general network of Figure 2.5.

	$\Lambda^- = 0.5$	$\Lambda^- = 1$	$\Lambda^- = 5$	$\Lambda^- = 10$	$\Lambda^- = 50$
v	0.8512	0.7321	0.2818	0.1162	0.0012
$\lambda^-_{i_1}$	-	-	0.16 (3.2%)	0.68 (6.8%)	2.18 (4.4%)
$\lambda^-_{i_2}$	0.07 (14.1%)	0.14 (14.1%)	0.50 (9.9%)	0.83 (8.3%)	2.78 (5.6%)
$\lambda^-_{i_3}$	0.10 (19.5%)	0.20 (19.7%)	1.05 (20.9%)	1.79 (17.9%)	4.36 (8.7%)
$\lambda^-_{i_4}$	0.10 (19.5%)	0.20 (19.7%)	0.78 (15.6%)	1.18 (11.8%)	2.84 (5.7%)
$\lambda^-_{i_5}$	0.07 (14.1%)	0.14 (14.1%)	0.72 (14.4%)	1.10 (11.0%)	2.73 (5.5%)
$\lambda^-_{i_6}$	0.07 (14.1%)	0.14 (14.1%)	0.73 (14.7%)	1.29 (12.9%)	3.11 (6.2%)
$\tilde{\Lambda}^-_{r_1}$	-	-	-	-	3.19 (6.4%)
$\tilde{\Lambda}^-_{r_2}$	-	-	-	0.11 (1.1%)	3.74 (7.5%)
$\tilde{\Lambda}^-_{r_3}$	-	-	0.30 (5.9%)	0.83 (8.3%)	6.24 (12.5%)
$\tilde{\Lambda}^-_{r_4}$	-	-	-	0.30 (3.0%)	4.77 (9.5%)
$\tilde{\Lambda}^-_{r_5}$	-	-	0.01 (0.1%)	0.40 (4.0%)	5.54 (11.1%)
$\tilde{\Lambda}^-_{r_6}$	0.09 (18.8%)	0.18 (18.2%)	0.76 (15.2%)	1.48 (14.8%)	8.54 (17.1%)

We would expect that the interdiction budget is evenly spread over the routes to make sure that the maximum completion probability is minimal. Table 2.2 shows

the expected spread of interdiction budget over the routes. For example in the last case ($\Lambda^- = 50$), all routes get around 24% of the total budget. From Lemma 2.2, we expect that nodes in shorter routes (routes 3, 5 and 6) would have higher interdiction rates than nodes along longer routes. This can also be seen in Table 2.2. Table 2.2 also shows that if the interdiction budget Λ^- is low, most budget is assigned to the intersection nodes because multiple routes can be protected simultaneously from these nodes. However, if the total interdiction budget increases, more budget remains for the tandem nodes. Moreover, more budget is assigned to intersection nodes where more routes intersect, such as i_3, because more routes can be protected from the same point. Also, routes with a small number of intersection nodes, such as r_6, have more budget allocated on the tandem nodes to ensure that these routes are sufficiently protected. In this example, the total route budget is almost the same for each route. This doesn't have to be the case if the lengths of all routes are very different or the service rates are unequal.

2.4 Probabilistic routing of intruders

In Section 2.2, we described an interdiction game on a network with fixed routing of intruders. In that game, intruders select their route upon arrival at the network by choosing from a fixed set of routes. We can also model probabilistic routing of the intruders. In this case, intruders decide their next step at each node according to a certain probability. In this section, we describe the game with probabilistic routing of intruders and show that the results coincide with those for fixed routing of intruders.

2.4.1 Network with probabilistic routing of intruders

Consider a network, similar to the network of Section 2.2.1, but now with probabilistic routing of the intruders. Intruders arrive at the network according to a Poisson process with rate Λ and route through the network using a probability matrix $P = (p_{i,j})$, $i, j \in \{0, 1, \ldots, N, N+1\}$ where $p_{i,j}$ is the probability of routing to node j after service completion at node i. This probability $p_{i,j}$ is only allowed to be positive if there is a link between node i and node j in the queueing network; the set of all possible links is given by E. Intruders arrive at node i, $i \in C_S$, with probability $p_{0,i}$, so the arrival rate at node i is given by $\lambda_i = p_{0,i}\Lambda$. If $i \notin C_S$, $\lambda_i = 0$. As $i \in C_T$ has just one outgoing arc (to $N+1$), the probability of leaving the network after service completion at node $i \in C_T$ is given by $p_{i,N+1} = 1$. Note that a P matrix may introduce routes with an arbitrary number of cycles.

Let R be the (possibly infinite) set of all possible finite routes through the network, in which $r(k, s)$ is the s-th node of route $k \in R$ and N_k is the length of route k (in which 0 and $N+1$ are not accounted for). We let $r(k, N_k + 1) = N + 1$. Then, given matrix P, the probability that route k is chosen by the intruder equals:

$$\mathbb{P}(\text{route } k \text{ is chosen}) = p_{0,r(k,1)} \prod_{s=1}^{N_k - 1} p_{r(k,s),r(k,s+1)}. \tag{2.31}$$

The probability that intruders leave node i because they finished service is given by Equation (2.1) and the probability that route k is actually completed without interdiction is given by Equation (2.2).

2.4.2 Game description

Consider the interdiction game with the probabilistic routing of intruders. Instead of intruders selecting arrival rates λ_k for route k, intruders select a routing matrix P. Therefore, the action set of the intruders, see Equation (2.3), is replaced by:

$$\bar{A}_I = \left\{ P \middle| \sum_{j=1}^{N+1} p_{i,j} = 1,\ i = 0, ..., N; p_{i,j} \geq 0, p_{i,j} = 0 \text{ if } (i,j) \notin E \right\}.$$

The agents action set remains the same as in the fixed routing scenario, see Equation (2.4). The payoff function is replaced by the corresponding payoff function, which defines the arrival rate of intruders at node $N+1$:

$$\bar{v}(P, \lambda^-) = \sum_{k \in R} \lambda_{r(k,1)} \prod_{s=1}^{N_k} p_{r(k,s),r(k,s+1)} \frac{\mu_{r(k,s)}}{\mu_{r(k,s)} + \lambda^-_{r(k,s)}}, \tag{2.32}$$

with $\lambda_{r(k,1)} = p_{0,r(k,1)}\Lambda$.

2.4.3 Relation between optimal strategies

In Section 2.3, we described methods to find optimal strategies for the interdiction game on a network with fixed routing of intruders. In this section, we discuss the relationship between the optimal strategies for a network with probabilistic routing of intruders. We first show that for each network with probabilistic routing, there exists a network with fixed routing of intruders such that the average arrival rates are equal and vice versa.

Lemma 2.4. *Take λ^- fixed. For every network with probabilistic routing of intruders and given λ, there exists a network with fixed routing of intruders, such that the average arrival rate at each node is the same in both networks. Furthermore, for every network with fixed routing of intruders and given λ, there exists a network with probabilistic routing of intruders, such that the average arrival rate at each node is the same in both networks.*

The proof of Lemma 2.4 can be found in Section 2.6.

We use Lemma 2.4 to prove that optimal strategies also exist in the case that intruders use probabilistic routing. Consider a network with N intermediate nodes, a source node and a sink node. Moreover, let F_{total} be the finite set of all possible fixed routes without cycles between the source node 0 and the sink node $N+1$. For that case, an optimal strategy for the intruders and agents can be calculated by the optimization model in Section 2.2.4. These strategies are given by λ^* and λ^{-*} and the optimal value is given by v. We show that the value of the game with probabilistic routing of intruders exists and is the same as the value of the game with fixed routing of intruders. Moreover, optimal strategies of the agent are the same for both games.

Theorem 2.5. *Consider the interdiction game on a queueing network with probabilistic routing of intruders. There exist optimal strategies P^* and λ^{-*} and the value of the game with probabilistic routing of intruders equals the value v of the game with fixed routing on F_{total}. Moreover, the strategy of the agent is also optimal for the game with fixed routing.*

Proof. Take an arbitrary routing matrix P that describes a strategy of intruders for a network with probabilistic routing of intruders. Suppose that the agent chooses the arrival rates according to the optimal strategy λ^{-*} of the game with fixed routing of intruders on F_{total}. By Lemma 2.4 and given λ^{-*}, we can construct a network with a set of fixed routes $\bar{F}$ and strategy $\bar{\lambda}$ such that the average arrival rate at each node is the same for the network with probabilistic routing and fixed routing of intruders. Because the payoff of both games, see Equations 2.5 and 2.32, is given by the arrival rate at the sink node, it follows that:

$$v(\bar{\lambda}, \lambda^{-*}) = \bar{v}(P, \lambda^{-*}). \tag{2.33}$$

The set of fixed routes $\bar{F}$, derived from probabilistic routing may be infinite. This is due to the fact that probabilistic routing may induce cyclic paths. We show that for our model with fixed routing, cyclic routes can be eliminated. To this end, suppose that the intruder assigns a positive arrival rate to a cyclic route k: $\lambda_k > 0$. By arbitrarily eliminating detours in the cyclic route, we obtain a non-cyclic route $\bar{k}$ such that $\mathbb{P}(\text{route } \bar{k} \text{ is completed}) \geq \mathbb{P}(\text{route } k \text{ is completed})$ (by Equation (2.2)). Transferring the rate λ_k to $\lambda_{\bar{k}}$ results in an improved strategy for the intruder.

So, let $\bar{F}'$ be the set of routes derived from P, with all cyclic routes eliminated and let $\bar{\lambda}'$ be the corresponding improved strategy for the intruder, so:

$$v(\bar{\lambda}, \lambda^{-*}) \leq v(\bar{\lambda}', \lambda^{-*}). \tag{2.34}$$

Also, because λ^* is the optimal strategy of the intruder for the case that all possible fixed routes without cycles are allowed, it follows that

$$v(\bar{\lambda}', \lambda^{-*}) \leq v(\lambda^*, \lambda^{-*}) = v. \tag{2.35}$$

Combining Equations (2.33), (2.34) and (2.35) yields:

$$\bar{v}(P, \lambda^{-*}) \leq v, \quad \text{for all } P. \tag{2.36}$$

We now complete the proof by showing that there exists a P^* such that $\bar{v}(P^*, \lambda^-) \geq v$, for all λ^-. Given optimal strategies λ^* and λ^{-*} from the game with fixed routing, a routing matrix P^* can be constructed according to Lemma 2.4. Because the average arrival rates are the same, the average arrival rates at the sink node are also equal and the values of the payoff functions of both the game with probabilistic routing and the game with fixed routing are equal. Therefore:

$$\bar{v}(P^*, \lambda^{-*}) = v(\lambda^*, \lambda^{-*}) = v. \tag{2.37}$$

Consider an arbitrary strategy λ^- for the agent. Using the same argument, we know that:

$$\bar{v}(P^*, \lambda^-) = v(\lambda^*, \lambda^-) \geq v, \tag{2.38}$$

as λ^* is optimal for the intruder.

Combining (2.36) and (2.38) proves that the value exists and is given by v. Moreover, the optimal strategy of the agent remains the same. ■

Remark 2.3. For a network with probabilistic routing, the construction of the network with fixed routing as in Lemma 2.4 also ensures that the probability of following a specific route is the same for both networks. This means that for every network with probabilistic routing of intruders, there exists an equivalent network with fixed routing of intruders.

However, the reverse is not necessarily true. In fact, consider a network with fixed routing of intruders; it is not always possible to construct an equivalent network with probabilistic routing of intruders as the next example shows. Consider a network with two routes and one intersection node. If two routes intersect, there may exist more routes in the network with probabilistic routing than in the original network with fixed routing, due to combinations of the original routes.

Although it is not always possible to create a probabilistic network that is equivalent to the network with fixed routing, one can introduce multiple intruders to ensure that the probabilistic network contains the same routes.

The question now becomes: how many intruder types are necessary to construct a network with probabilistic routing which is equivalent to the network with fixed routing? Below we describe how to find an upper bound for the number of types we need.

Consider a network with fixed routing and K possible routes. To find an upper bound for the number of customer types, we construct a graph G, that has K nodes. All nodes correspond to a route, and two nodes are considered to be connected if the corresponding routes intersect in the network with fixed routing. An upper bound for the number of types equals the chromatic number $\chi(G)$. The nodes that do have the same type in the vertex coloring do not intersect, so they are allowed to have the same intruder type in the probabilistic network. This upper bound cannot be improved for the general case. However, fewer types will be enough in many specific situations, such as when routes only intersect at their last node. □

Remark 2.4. Note that the network with probabilistic routing is known to have a product-form solution [40]. We do not need this to calculate the intruders' probability of completing a specific route because we can rely on the fact that intruders are only removable from the queue if they are in service and that agents arrive at the network according to an independent Poisson process. □

2.5 Concluding remarks

In this chapter, we have described an interdiction game on a network with intruders and agents. The interdiction game is played within a queueing network where intruders are the regular customers and the agents are the negative customers. For the case that the routing of the intruders is considered fixed, we designed a network game that has optimal pure strategies and we found analytical expressions for special cases, such as networks with only tandem or parallel queues. Also, for a network without intersecting nodes, we introduced an algorithm to find optimal strategies for the agents.

For general networks, we showed that the analytical results can be used to speed up finding optimal strategies, by dividing the network into a set of intersection nodes and separate tandem nodes. Moreover, if there is a subnetwork of the network, that only consists of parallel routes which do not intersect, then the optimal strategy of the agent is such that the completion probability is the same for each of these routes. Also,

if the network contains a part that only consists of tandem nodes without intersections, the nodes with a lower service rate must be inspected more often.

In this chapter, we also considered modeling the routing of intruders in a probabilistic manner. We showed that in this case, optimal strategies for agents and intruders also exist. Moreover, the value and optimal strategies of the agent of this game equal the value and optimal strategies of the agent of the corresponding game with fixed routing of intruders. So, the intruders cannot improve their strategy by deciding to use a probabilistic routing strategy.

There are several possible extension of our model. Instead of modeling agents that arrive at a specific node for inspection and then leave the network, it could be more realistic in some cases to model routing of agents. In this approach, agents not only inspect the nodes, but also route through the network. Another possible extension concerns each node as a single server queue with exponential service time. For further research, it would be interesting to study different types of queues, for example with multiple servers or a different service discipline.

2.6 Appendix

Proof of Theorem 2.3. Rewrite the optimization problem for the original game as:

$$\begin{aligned} \min_{\lambda^-} \quad & \Lambda \prod_{i=1}^{N} \frac{\mu_i}{\mu_i + \lambda_i^-} \\ \text{s.t.} & \sum_{i=1}^{N} \lambda_i^- - \Lambda^- = 0, \\ & -\lambda_i^- \leq 0, \qquad i = 1, ..., N. \end{aligned}$$

The KKT-conditions can be used to prove optimality of a solution in a non-linear program (see Section 4.3 in [13]). In order to prove optimality, we show that the KKT-conditions hold for the solution λ^{-*} obtained by Algorithm 1 . Thus, α and β must be found such that:

$$\frac{-1}{\mu_i + \lambda_i^{-*}} v(\lambda^{-*}) + \alpha - \beta_i = 0, \quad i = 1, ..., N, \tag{2.39}$$

$$\sum_{j=1}^{N} \lambda_j^{-*} - \Lambda^- = 0, \tag{2.40}$$

$$\beta_i \lambda_i^{-*} = 0, \quad i = 1, ..., N, \tag{2.41}$$

$$\beta_i \geq 0, \quad i = 1, ..., N. \tag{2.42}$$

Let $C' = \{i \in C | \lambda_i^{-*} > 0\}$ and let $N' = |C'|$. The equality condition (Equation (2.40)) holds by construction of the algorithm:

$$\sum_{i=1}^{N} \lambda_i^{-*} = \sum_{i \in C'} \left(\frac{\sum_{j \in C'} \mu_j + \Lambda^-}{N'} - \mu_i \right) = N' \frac{\sum_{j \in C'} \mu_j + \Lambda^-}{N'} - \sum_{i \in C'} \mu_i = \Lambda^-$$

Moreover, from (2.41), we know that $\beta_i = 0$, for all $i \in C'$, so (2.39) gives:

$$\frac{-1}{\mu_i + \lambda_i^{-*}} v(\lambda^{-*}) + \alpha = 0, \quad \text{for all } i \in C'.$$

Therefore, using (2.21):

$$\alpha = \frac{1}{\mu_i + \lambda_i^{-*}} v(\lambda^{-*}) = \frac{N'}{\sum_{j \in C'} \mu_j + \Lambda^-} v(\lambda^{-*}).$$

If $\lambda_i^{-*} = 0$ (for all $i \in C \setminus C'$), (2.39) gives:

$$\frac{-1}{\mu_i} v(\lambda^{-*}) + \frac{N'}{\sum_{j \in C'} \mu_j + \Lambda} v(\lambda^{-*}) - \beta_i = 0,$$
$$\beta_i = \left(\frac{N'}{\sum_{j \in C'} \mu_j + \Lambda^-} - \frac{1}{\mu_i} \right) v(\lambda^{-*}).$$

By proving that $\beta_i \geq 0$ for all i, the proof that the KKT-conditions hold is completed.

Note that the value of the function $v(\lambda^{-*})$ is positive by definition, Moreover, by construction of the algorithm, we know for any $i \in C \setminus C'$:

$$\frac{\sum_{j \in C' \cup \{i\}} \mu_j + \Lambda^-}{N' + 1} \leq \mu_i,$$
$$\frac{\sum_{j \in C'} \mu_j + \Lambda^-}{N' + 1} \leq \frac{N'}{N' + 1} \mu_i,$$
$$\frac{N'}{\sum_{j \in C'} \mu_j + \Lambda^-} - \frac{1}{\mu_i} \geq 0.$$

Therefore, $\beta_i \geq 0$, for all $i \in C$ and the KKT-conditions hold for the solution found by Algorithm 1. Furthermore, because of the convexity of the value function and linearity of the constraints, the KKT-conditions are sufficient, which completes the proof. ■

Proof of Theorem 2.4. The value of route k is defined as the payoff of the game if the intruders choose to use only route k, so $\lambda_k = \Lambda$. In an optimal solution, the value for each route should be equal. If this is not the case, the strategy of the agent can be improved by shifting the arrival rate λ^{-*}, such that more interdiction budget is assigned to the route of minimal detection probability. To prove optimality of the algorithm, (1) the value over each route must be equal, (2) the algorithm must find a feasible solution, and (3) the arrival rates that are set to zero by the algorithm, must also be zero in the optimal solution.

The last condition is necessary because of the following. If for a certain node the agent's arrival rate is set to zero in the optimal solution, then the probability of service completion at this node equals one. In essence, this node has no impact on the total throughput of the intruders. Therefore, ignoring these nodes in the optimization model, which is done for the relaxation in the last step of the algorithm, gives the same solution. If solving this relaxation without these nodes also yields a feasible solution, the solution of the relaxation also is a solution for the original game.

The first condition holds because of construction of the algorithm: the optimal strategy is calculated under this assumption (by Equation (2.26)). The second condition holds because the algorithm stops if all arrival rates are larger than or equal to zero. We will prove that the arrival rates that are set to zero by the algorithm, are also

zero in the optimal solution. Let C be the set of all nodes and let C_k be the set of nodes that are in route k. Moreover, let $C' = \{i \in C | \lambda_i^- > 0, \text{ by algorithm}\}$ and let $C^{OPT} = \{i \in C | \lambda_i^{-*} > 0, \text{ in optimal solution}\}$. We must therefore prove that $C' = C^{OPT}$. Let v' be the value found by the algorithm, let v^{OPT} be the value of the original game and let v^t be the value of the relaxation calculated during iteration t in step 2 of the algorithm.

We first prove that $C' \supseteq C^{OPT}$. Take $i \notin C'$ such that i is removed during the first iteration and $\mu_i = \max_{j \in C_k} \mu_j$ for some k. Because v^1 is the value for the relaxation without any inequality constraints, we know that $v^{OPT} \geq v^1$. Let Λ_k^{-1} be the budget assigned to route k during the first iteration and let Λ_k^{-OPT} be the budget assigned to route k in the optimal solution. Consider two cases:

1. $\Lambda_k^{-1} \geq \Lambda_k^{-OPT}$:
   Thus, in the optimal solution, route k receives a smaller or equal amount of budget. The arrival rate λ_i^- is obtained from Equation (2.29) and is increasing in Λ_k^-. Since it is optimal to use Algorithm 1 to assign the budget to the nodes of one tandem, the same formula must hold for λ_i^{-*} in the optimal solution. So if $\Lambda_k^{-1} \geq \Lambda_k^{-OPT}$, λ_i^{-*} would also be zero in the optimal solution and therefore, $i \notin C^{OPT}$.

2. $\Lambda_k^{-1} < \Lambda_k^{-OPT}$:
   Assume that $i \in C^{OPT}$, so $\lambda_i^{-*} > 0$. Because of maximality of μ_i, all $j \in C_k$ are in C^{OPT}. Because for all $j \in C_k$, $\lambda_i^{-*} > 0$ and the value of the game equals the value for each route (Lemma 2.1), v^{OPT} is given by (2.24) with $\Lambda_k^- = \Lambda_k^{-OPT}$ and $\lambda_k = \Lambda$. Moreover, during the first iteration, $C' = I$, so v^1 is given by (2.24) with $\Lambda_k^- = \Lambda_k^{-1}$. Equation (2.24) is decreasing in Λ_k^- and therefore, $v^{OPT} < v^1$, but this contradicts with $v^{OPT} \geq v^1$. So, our assumption is not correct and $i \notin C^{OPT}$.

Therefore, if $i \notin C'$ such that i is removed during the first iteration, it follows that $i \notin C^{OPT}$.

Now assume that for all $i \notin C'$ that are removed until iteration $t-1$, $i \notin C^{OPT}$. Take $j \notin C'$ such that j is removed during iteration t ($t > 1$). Let $\bar{C} = \{i \in C | i \text{ not removed before iteration } t\}$ and $\bar{C}^{OPT} = \{i \in \bar{C} | \lambda_i^{-*} > 0, \text{ in optimal solution}\}$. By induction, we know for all $i \in C \setminus \bar{C}$ that $i \notin C^{OPT}$. So, running the algorithm with $\bar{C}$ gives the same solution as running the algorithm with C. By this logic, the above argument can be used to prove that if $i \notin C'$ such that i is removed during the t-th iteration and $\mu_i = \max_{j \in C_k} \mu_j$ for some k, then $i \notin C^{OPT}$. In general, if $i \notin C'$ then $i \notin C^{OPT}$ and $C' \supseteq C^{OPT}$.

We now prove $C' \subseteq C^{OPT}$ by contradiction. Assume that $C' \not\subseteq C^{OPT}$. Because we already proved $C' \supseteq C^{OPT}$, it follows that $C' \supset C^{OPT}$. The values v' and v^{OPT} are the solutions of the optimization problem (Lemma 2.1) under the additional constraints $\lambda_i^- = 0$, for all $i \notin C'$ and $\lambda_i^- = 0$, for all $i \notin C^{OPT}$ respectively. If $C' \supset C^{OPT}$, there exists at least one $i' \in C'$ such that $i' \notin C^{OPT}$. Therefore, $\lambda_{i'}^- > 0$ for the first case ($\lambda_i^- > 0$, for all $i \notin C'$), but $\lambda_{i'}^- = 0$ for the second case ($\lambda_i^- = 0$, for all $i \notin C^{OPT}$). This results in a worse solution for the second case and $v' < v^{OPT}$. This also contradicts with optimality of v^{OPT}, thus the assumption was incorrect and $C' \subseteq C^{OPT}$.

Combining $C' \supseteq C^{OPT}$ and $C' \subseteq C^{OPT}$ gives $C' = C^{OPT}$, which completes the proof. ■

Proof of Lemma 2.4. Consider a network with probabilistic routing of intruders with arrival rates of intruders λ_i and agents λ_i^-, and service rates μ_i for all $i \in C$. R is the set of all possible routes induced by P (R may be infinite). Now, we construct a network with fixed routing of intruders with arrival rate of intruders $\bar{\lambda}_k$ for a route k. Define for each route $k \in R$ the arrival rate of intruders following fixed routing by:

$$\bar{\lambda}_k = \lambda_{r(k,1)} p_{r(k,1),r(k,2)} p_{r(k,2),r(k,3)} \cdots p_{r(k,N_k),N+1}. \tag{2.43}$$

This is the arrival rate at the first node multiplied by the probability of following this route, given P. For the network with probabilistic routing, the mean arrival rate α_i at node i is given by the traffic equations:

$$\alpha_i = \lambda_i + \sum_j \alpha_j p_{j,i} \frac{\mu_j}{\lambda_j^- + \mu_j}. \tag{2.44}$$

For the network with fixed routing, the mean arrival rates are defined by:

$$\bar{\alpha}_i = \sum_{k \in R} \sum_{s=1}^{N_k} a_i(k,s),$$

where

$$a_i(k,s) = \begin{cases} \bar{\lambda}_k \prod_{t=1}^{s-1} \frac{\mu_{r(k,t)}}{\mu_{r(k,t)} + \lambda_{r(k,t)}^-}, & \text{if } r(k,s) = i, \\ 0, & \text{otherwise.} \end{cases}$$

Substituting the definitions (2.43) and (2.44), and rearranging terms yields:

$$\alpha_i = \lambda_i + \sum_j \lambda_j p_{j,i} \frac{\mu_j}{\lambda_j^- + \mu_j} + \sum_j \sum_h \lambda_h p_{h,j} p_{j,i} \frac{\mu_j}{\lambda_j^- + \mu_j} \frac{\mu_h}{\lambda_h^- + \mu_h} + ...$$

The same expression can be found for $\bar{\alpha}_i$ by rewriting the traffic equations for the network with fixed routing and substituting the definition of $a_i(k,s)$. Thus, by construction, the average arrival rates at each node of the network with intruders following fixed routing equal the average arrival rates of the network with intruders following probabilistic routing.

Now consider a network with fixed routing of intruders with rates $\bar{\lambda}_k$ and λ^-. We can construct a network with probabilistic routing of intruders such that the average arrival rates are equal. For every route k, we have $r(k,1), r(k,2), ..., r(k,N_k)$ and arrival rate $\bar{\lambda}_k$ at node $r(k,1)$. The probability $p_{i,j}$ that an intruder is going to node j after completing service in node i can be calculated by dividing the flow from i to j by the total flow out of i:

$$p_{i,j} = \frac{\sum_{k=1}^{K} \sum_{s=1}^{N_k - 1} \bar{b}_{i,j}(k,s)}{\sum_{k=1}^{K} \sum_{s=1}^{N_k} \bar{a}_i(k,s)},$$

where:

$$\bar{a}_i(k,s) = \begin{cases} \bar{\lambda}_k, & \text{if } r(k,s) = i, \\ 0, & \text{otherwise,} \end{cases}$$

$$\bar{b}_{i,j}(k,s) = \begin{cases} \bar{\lambda}_k, & \text{if } r(k,s) = i \text{ and } r(k,s+1) = j, \\ 0, & \text{otherwise.} \end{cases}$$

Also, the arrival rates at node i for the network with probabilistic routing are given by:

$$\lambda_i = \sum_{k=1}^{K} \bar{a}_i(k,1).$$

Now, we can readily show that given λ^- the average arrival rates at each node of the network with fixed routing equal the average arrival rates of the network with probabilistic routing. ■

CHAPTER 3

Non-cooperative queueing games on a Jackson network

This chapter resulted in [74].

3.1 Introduction

In this chapter, we introduce and analyze a new type of queueing games: non-cooperative games on a Jackson network. The game takes place on a Jackson queueing network, where multiple players compete to minimize their own expected sojourn time. All players decide on a routing strategy through the network by distributing their arrival rate over a fixed set of routes. First, we analyze the game with continuous strategies, where players can split their total rate over several routes. Second, we consider the game where each player is only allowed to pick a single route from the source to sink node. This game is a special variant of a weighted congestion game [111]. In this case, there does not always exists a Nash equilibrium in pure strategies, but we discuss several cases for which there does exist one.

In the literature, there are several papers combining game theory and queueing models. For example in communication networks, game theory is used to determine optimal routing strategies (e.g.,[6]) and in security, queueing is used to model stochastic elements in interdiction games (e.g., [127] or Chapter 2). In [53], the authors give a broad overview of models of rational behavior in queueing systems. In [24], the authors discuss a queueing game where Braess' paradox also occurs on a queueing network.

In this chapter, we discuss a game on a Jackson network. The cooperative variant of Jackson games is introduced by Timmer and Scheinhardt in [121, 122]. This game is played on a network of M/M/1 queues and each server in this network is considered as a player. In [121], the authors analyze the game where the players are allowed to cooperate by redistributing their service rates over the nodes in order to minimize the lung run expected queue length. In [122], a similar game is considered where the players cooperate by redistributing the arrival rates of the customers. We introduce the non-cooperative variant of this sgame and analyze for several cases the existence of a Nash equilibrium.

This chapter is organized as follows. In Section 3.2 we give a general description of the non-cooperative game on a queueing network. In Section 3.3 we consider the game with continuous strategy space, where each player is allowed to distribute his total

arrival rate over the set of fixed routes. We prove the existence of a Nash equilibrium and discuss an algorithm to find optimal solutions. In Section 3.4, we discuss the game where players route their total arrival rate over a single route. We discuss for several variants whether pure Nash equilibria exist or not. Finally, we give some concluding remarks in Section 3.5.

3.2 Model

In this section, we introduce the game on a Jackson network. By adjusting the constraints on the strategy space, both the continuous and discrete case are given by the same model.

Consider an n-player game on a queueing network with N nodes. The set of all players is $\mathcal{N}$ and the set of all nodes is C. Each node i represents an M/M/1 queue with service rate μ_i, $i = 1, .., N$. The arrival rate, λ_i, of each node depends on the routes chosen by each player. The total arrival rate for player j is $\lambda^{(j)}$, which has to be divided over a given set of routes $R^{(j)}$, $j = 1, ..., n$. A route $r \in R^{(j)}$ is given by a set of nodes. Each player decides on the arrival rate to each route in order to minimize its own total expected sojourn time. The strategy of player j is given by the vector $p^{(j)}$, where $p_r^{(j)}$ is the probability that route $r \in R^{(j)}$ is selected. So the arrival rate for player j to route r is $p_r^{(j)}\lambda^{(j)}$. Let $p = \{p^{(1)}, ..., p^{(n)}\}$ be the strategy of all players and $p_{-j} = \{p^{(1)}, ..., p^{(j-1)}, p^{(j+1)}, ..., p^{(n)}\}$ the strategy without player j's strategy.

In this game, the objective of all players is to minimize their own expected sojourn time. The mean sojourn time of a single node i depends on the strategy of all players and is given by:

$$\frac{1}{\mu_i - \lambda_i},$$

where $\lambda_i = \sum_{j=1}^{n} \sum_{r:i\in r, r\in R^{(j)}} p_r^{(j)}\lambda^{(j)}$ is the total arrival rate for node i. The total weighted mean sojourn time for player j is:

$$f^{(j)}(p) = \sum_{i=1}^{N} \frac{\sum_{r:i\in r, r\in R^{(j)}} p_r^{(j)}}{\mu_i - \lambda_i}, \tag{3.1}$$

and the total weighted mean sojourn time of all players is:

$$f(p) = \sum_{j=1}^{n} f^{(j)}(p).$$

Consider the game with continuous strategy space and thus, each player is allowed to split the arrival rate over all possible routes. Each player only minimized their own expected sojourn time. Given the strategy p_{-j}, the optimal strategy for player j can be found by solving the following linear program:

$$\min_{p^{(j)}} \quad \sum_{i=1}^{N} \frac{\sum_{r:i\in r, r\in R^{(j)}} p_r^{(j)}}{\mu_i - \lambda_i} \tag{3.2}$$

$$\text{s.t. } \lambda_i = \sum_{j=1}^{n} \sum_{r:i\in r, r\in R^{(j)}} p_r^{(j)}\lambda^{(j)}, \quad i \in C, \tag{3.3}$$

$$\lambda_i < \mu_i, \qquad i \in C, \tag{3.4}$$

$$\sum_{r \in R^{(j)}} p_r^{(j)} = 1, \tag{3.5}$$

$$p^{(j)} \geq 0. \tag{3.6}$$

By solving (3.2)-(3.6) simultaneously for all players j gives the value of p in a Nash equilibrium. By replacing Constraint (3.6) by $p_k^{(j)} \in \{0, 1\}$, the game with discrete strategies where players can only select a single route is modeled.

In the next section, we analyze the game with continuous strategy space and thereafter in Section 3.4, we discuss the game with discrete strategies.

3.3 Game with continuous strategies

In the previous section, we described non-cooperative games on a Jackson network. In this section, we analyze the game where continuous strategies are considered, so where $p_r^{(j)} \in [0, 1]$, $j = 1, ..., n$, $r \in R^{(j)}$. This means that each player is allowed to split his arrival rate over the set of fixed routes. We prove that there exists a unique pure Nash equilibrium for this game and discuss approaches to find this Nash equilibrium.

3.3.1 Existence of pure Nash equilibrium

We first prove convexity of the payoff function for each player separately, such that the strategies of the other players are fixed. Let $f^{(j)}(p)$ (Equation (3.1)) be the payoff function of player j.

Lemma 3.1. *The payoff function $f^{(j)}(p)$ is strictly convex in $p^{(j)}$ for $p \in \mathcal{P}$, $p = (p^{(j)}, p_{-j})$, where:*

$$\mathcal{P} = \{p | \sum_{r: r \in R^{(j)}} p_r^{(j)} = 1, p^{(j)} \geq 0, j \in \mathcal{N}, \sum_{j=1}^{n} \sum_{r: i \in r, r \in R^{(j)}} p_r^{(j)} \lambda^{(j)} < \mu_i, \ i \in C\}. \tag{3.7}$$

Proof. First, we rearrange $f^{(j)}(p)$ such that it is a function of $p^{(j)}$. Since p_{-j} is fixed, we can subtract the total arrival rate at a node by all players but j from μ_i. Let $f(\tilde{p}) = \sum_{i \in C} \frac{\tilde{p}_i}{\tilde{\mu}_i - \tilde{p}_i \lambda^{(j)}}$, where $\tilde{p} = p^{(j)}$. Moreover, $\tilde{p} \in \tilde{\mathcal{P}}$ describes the probability that node i is selected by player j and $\tilde{\mu}_i = \mu_i - \sum_{k \in \mathcal{N} \setminus j} \sum_{r: i \in r, r \in R^{(k)}} p_r^{(k)} \lambda^{(k)}$, which is a constant number since p_{-j} is fixed. $\tilde{\mathcal{P}}$ is the subset of $\mathcal{P}$ where p_{-j} is given:

$$\tilde{\mathcal{P}} = \{\tilde{p} | \tilde{p}_i^{(j)} = \sum_{j=1}^{n} \sum_{r: i \in r, r \in R^{(j)}} p_r^{(j)}, i \in C, j \in \mathcal{N}, p \in \mathcal{P}\}.$$

We construct the Hessian of $f^{(j)}(\tilde{p})$, which is given by the following entries:

$$\frac{\partial f^{(j)}}{\partial \tilde{p}_i^{(j)}} = \frac{\tilde{\mu}_i}{(\tilde{\mu}_i - \tilde{p}_i^{(j)} \lambda^{(j)})^2}, \qquad i \in C,$$

$$\frac{\partial f^2}{\partial \tilde{p}_i^{(j)} \partial \tilde{p}_i^{(j)}} = \frac{2 \tilde{\mu}_i \lambda^{(j)}}{(\tilde{\mu}_i - \tilde{p}_i^{(j)} \lambda^{(j)})^3}, \qquad i \in C,$$

$$\frac{\partial f^2}{\partial \tilde{p}_i^{(j)} \partial \tilde{p}_k^{(j)}} = 0, \qquad i,k \in C.$$

Since only the diagonal elements have non-zero numbers which are non-negative for all $\tilde{p} \in \tilde{\mathcal{P}}$, it can easily be seen that the Hessian is positive and therefore, $f^{(j)}(p)$ is strictly convex in $p^{(j)}$. ■

Using Lemma 3.1, we can show that there exists a Nash equilibrium for the non-cooperative game on a Jackson network.

Theorem 3.1. *The non-cooperative game on a Jackson network has a unique pure Nash equilibrium if the set $\mathcal{P}$ is non empty, where $\mathcal{P}$ is given as in* (3.7).

Proof. A continuous game has a unique pure Nash equilibrium if the strategy space is bounded and convex and the payoff function is continuous and strictly convex for each player (see Theorem 1 in [110]). It can be easily seen that the strategy space is bounded since $p^{(j)} \geq 0$ and $\sum_{r:r\in R^{(j)}} p_r^{(j)} = 1$. Moreover, the set is convex since it is constructed with linear (in)equalities. From Lemma 3.1 it follows that the payoff function is convex for each player. Also, since $\frac{p}{\mu - p\lambda}$ is continuous for each p where $\mu - p\lambda > 0$, the payoff function is continuous on $\mathcal{P}$. Therefore, it follows that there exists a unique pure Nash equilibrium. ■

3.3.2 Finding pure Nash equilibria

In this section, we discuss methods that can be used to find optimal solutions.

To find an optimal Nash equilibrium, Equations (3.2)-(3.6) have to be solved for each player j. One method to find the exact solutions of these mathematical programs simultaneously is by using Lagrange multipliers. However, as we show in the following example, this results in solving a system of nonlinear equations.

Example 3.1. Consider a network with 3 nodes. There are two players with routes $R^{(1)} = \{\{1\},\{2\}\}$ and $R^{(2)} = \{\{2\},\{3\}\}$. The payoff functions for player 1 and 2 are given by:

$$f^{(1)}(p^{(1)}) = \frac{p^{(1)}}{\mu_1 - p^{(1)}\lambda^{(1)}} + \frac{1-p^{(1)}}{\mu_2 - (1-p^{(1)})\lambda^{(1)} - (1-p^{(2)})\lambda^{(2)}},$$
$$f^{(2)}(p^{(2)}) = \frac{p^{(2)}}{\mu_3 - p^{(2)}\lambda^{(2)}} + \frac{1-p^{(2)}}{\mu_2 - (1-p^{(1)})\lambda^{(1)} - (1-p^{(2)})\lambda^{(2)}},$$

where $p^{(1)}$ is the probability that player 1 selects node 1 and $1-p^{(1)}$ is the probability that he selects node 2. Similar, $p^{(2)}$ is the probability that player 2 selects node 3 and $1-p^{(2)}$ is the probability that he selects node 2.

We solve this network using Lagrange multipliers. We assume that $\mu_1 > \lambda^{(1)}$, $\mu_3 > \lambda^{(2)}$ and $\mu_2 > \lambda^{(1)} + \lambda^{(2)}$.

The Lagrangians for both players are:

$$L_1(p^{(1)}, \alpha_1, \alpha_2) = \frac{p^{(1)}}{\mu_1 - p^{(1)}\lambda^{(1)}} + \frac{1-p^{(1)}}{\mu_2 - (1-p^{(1)})\lambda^{(1)} - (1-p^{(2)})\lambda^{(2)}} + \alpha_1 p^{(1)} - \alpha_2(1-p^{(1)}),$$

$$L_2(p^{(2)}, \beta_1, \beta_2) = \frac{p^{(2)}}{\mu_3 - p^{(2)}\lambda^{(2)}} + \frac{1 - p^{(2)}}{\mu_2 - (1 - p^{(1)})\lambda^{(1)} - (1 - p^{(2)})\lambda^{(2)}} + \beta_1 p^{(2)} - \beta_2(1 - p^{(2)}).$$

An optimal solution for $p^{(1)}$ and $p^{(2)}$ can be found by solving the following (in)equalities:

$$\begin{aligned}
&\frac{\partial L_1}{\partial p^{(1)}} = \frac{\mu_1}{(\mu_1 - p^{(1)}\lambda^{(1)})^2} - \frac{\mu_2 - p^{(2)}\lambda^{(2)}}{(\mu_1 - p^{(1)}\lambda^{(1)} - p^{(2)}\lambda^{(2)})^2} - \alpha_1 + \alpha_2 = 0,\\
&\frac{\partial L_2}{\partial p^{(2)}} = \frac{\mu_3}{(\mu_3 - p^{(2)}\lambda^{(2)})^2} - \frac{\mu_2 - p^{(1)}\lambda^{(1)}}{(\mu_1 - p^{(1)}\lambda^{(1)} - p^{(2)}\lambda^{(2)})^2} - \beta_1 + \beta_2 = 0,\\
&\alpha_1 p^{(1)} = 0,\\
&\alpha_2(1 - p^{(1)}) = 0,\\
&\beta_1 p^{(2)} = 0,\\
&\beta_2(1 - p^{(2)}) = 0,\\
&0 \le p^{(1)} \le 1,\\
&0 \le p^{(2)} \le 1.
\end{aligned}$$

Solving this set of equations gives an optimal solution where $p^{(1)} = p^{(2)} = 0.39$ if $\lambda^{(1)} = \lambda^{(2)} = 1$, and $\mu_1 = \mu_3 = 3$ and $\mu_2 = 4$. □

The approach described above can be used to find a Nash equilibrium for the non-cooperative queueing game. However, already for such a small game, this results in (in)equalities that are difficult to solve. Therefore, we use an iterative approach to find a Nash equilibrium.

A common approach to find Nash equilibria is by a best-response approach where all players iteratively search for there best-response:

Algorithm 3 Best-response algorithm

1: Initialize: construct a feasible solution for $p^{(j)}$, $j \in \mathcal{N}$.
2: Select a player $j \in \mathcal{N}$ such that $p^{(j)}$ is not a best response to p_{-j}. Update $p^{(j)}$ to a best response of j to p_{-j}.
3: If p is a Nash equilibrium then stop. Else, go to Step 2 with p.

Following [100] and [114], the best response algorithm always converges to the Nash equilibrium under the same conditions as needed for the existence of a pure Nash equilibrium. Therefore, following the same arguments as in the proof of Theorem 3.1 it can be shown that this best response algorithm always converge to a Nash equilibrium.

Theorem 3.2. *Algorithm 3 converges to a pure Nash equilibrium for the non-cooperative games on a Jackson network with continuous strategies.*

3.4 Game with discrete strategies

In the previous section, we discussed the game with continuous strategies. This section considers the game where each player is only allowed to choose discrete strategies. That

is, each player can only select one route resulting in a finite set of strategies for each player. For this game, there always exists a Nash equilibrium in mixed strategies [94]. However, we are interested in the case where players are not allowed to randomize over the possible routes. We first explain how this game translates to a weighted congestion game and thereafter, we discuss the existence of pure Nash equilibria for several cases.

3.4.1 Weighted congestion games

Games on a Jackson network with discrete strategies are a special variant of congestion games. In congestion games, multiple resources are available and each player selects a subset of these resources minimizing their own cost. The cost of each resource depends on the number of players selecting that resource. In a traditional congestion game, all players are equivalent and have the same influence on the cost of a single resource. For these games, it can be proven that there exists a Nash equilibrium in pure strategies [111].

The game discussed in this section is a special variant of a weighted congestion game. In weighted congestion games each player has a weight and the cost for each resource depends on the weighted sum of the players that pick that resource. Here, the weight translates to the total arrival rate of each specific player. In general, weighted congestion games do not always possess a Nash equilibrium in pure strategies (for example in [50, 90]).

However, there are several cases for which Nash equilibria do exist. For example, for games in which the underlying matrix is a matroid, it can be proven that there exists a pure Nash equilibrium [1]. Also for for games with affine or exponential cost functions, pure Nash equilibria exist [50]. In [123], the authors prove that weighted congestion games where players can split (integers only) their total weight over the resources have a Nash equilibrium when the cost functions are convex and monotonically increasing. Also for Shapley network congestion games where the players split the cost of a shared edge (e.g., [7, 23, 68]), pure Nash equilibria exist when at most two players can share an edge or when all players have the same source and sink node.

Motivated by [50], we illustrate by means of an example that for the games on a Jackson network discussed in this chapter, there does not necessarily exist a pure Nash equilibrium.

Example 3.2 (Pure Nash equilibrium need not exist)**.** Consider a Jackson game with two players with arrival rates $\lambda^{(1)} = 1$ and $\lambda^{(2)} = 2$. The game takes place on a network with 6 nodes and both player 1 (blue) and player 2 (red) have two possible strategies (Figure 3.1). The strategies of player 1 are $r_1^{(1)} = \{1, 2, 3\}$ and $r_2^{(1)} = \{4, 5, 6\}$, and for player 2 the strategies are $r_1^{(2)} = \{1, 2, 4\}$ and $r_2^{(2)} = \{3, 5, 6\}$. Moreover the service rates are given by $\mu_1 = \mu_2 = \mu_5 = \mu_6 = 6$ and $\mu_3 = \mu_4 = 4.95$. A matrix representing the expected sojourn time of both players for this game is:

		Player 2	
		$r_1^{(2)}$	$r_2^{(2)}$
Player 1	$r_1^{(1)}$	0.920/1.006	0.913/1.013
	$r_2^{(1)}$	0.913/1.013	0.920/1.006

As can be easily seen in this matrix representation, this game does not have a pure

Nash equilibrium since at each entry, at least one of the players has the incentive to switch to another strategy (row/column).

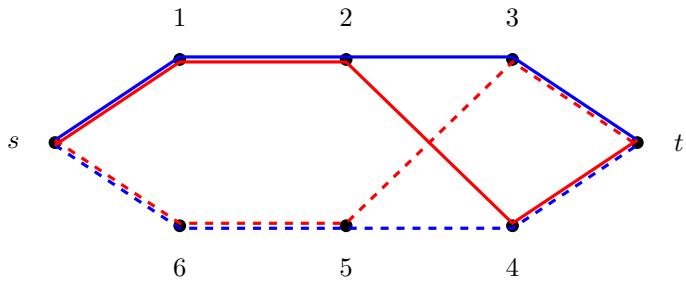


Figure 3.1: Counter example.

□

3.4.2 Existence of pure Nash equilibria

There are several subclasses of games that allow for a pure Nash equilibrium. If the arrival rates of all players are identical, this game translates to a traditional congestion game that has a pure Nash equilibrium.

Theorem 3.3. *The n-player non-cooperative game on a Jackson network with equal arrival rates $\lambda^{(j)} = \lambda$ for all players j, $j \in \mathcal{N}$, has a pure Nash equilibrium.*

Proof. We will show that the n-player non-cooperative game on a Jackson network with equal arrival rates $\lambda^{(j)} = \lambda$ for all players j is a congestion game. The result then follows from [111], since a congestion game has a pure Nash equilibrium.

Consider a strategy profile p for all players. Let $a_i^{(j)}(p)$ equal 1 if node i is used by player j in profile p, and 0 otherwise. Then $x_i(p) := \sum_{j=1}^{n} a_i^{(j)}(p) = \#(\text{players using node } i)$. Under strategy profile p, the sojourn time (3.1) of player j is

$$f^{(j)}(p) = \sum_{i=1}^{N} \frac{\sum_{\{r:i\in r, r\in R^{(j)}\}} p_r^{(j)}}{\mu_i - \lambda_i} = \sum_{i=1}^{N} \frac{a_i^{(j)}(p)}{\mu_i - \lambda_i},$$

and

$$\lambda_i = \sum_{j=1}^{n} \sum_{\{r:i\in r, r\in R^{(j)}\}} p_r^{(j)} \lambda^{(j)} = \sum_{j=1}^{n} a_i^{(j)}(p)\lambda^{(j)} = x_i(p)\lambda.$$

Let the delay function $d_i : \mathbb{N} \to \mathbb{R}$ be defined as

$$d_i(x) = \frac{1}{\mu_i - x\lambda}.$$

Then

$$f^{(j)}(p) = \sum_{i=1}^{N} a_i^{(j)}(p) d_i(x_i(p)).$$

In its domain $\{x|x < \mu_i/\lambda\}$, $d_i(x)$ is positive and monotone increasing. Note that $x_i(p) < \mu_i/\lambda$ is required for strategy profile p to be feasible. This implies that the n-player non-cooperative game on a Jackson network with equal arrival rates is a congestion game. ■

If the service rates at all nodes are identical in a 2-player non-cooperative game on a Jackson network, then the game has a pure Nash equilibrium.

Theorem 3.4. *For a two-player game with μ_i equal for each node i, $i \in C$, there always exists a pure Nash equilibrium.*

Proof. In [50], Theorem 3.12, the authors prove that there exists a pure Nash equilibrium for any two-player game where the cost function for each resource (node) can be written as $am(x) + b$, where $m(x)$ is a monotonic function and $a, b \in \mathbb{R}$. Since $\mu_i = \mu$ for each $i \in C$, we can choose $m(x) = \frac{1}{\mu - x}$, which is a monotonic increasing function for $\mu > x$. Then for every node, the cost function equals $m(\lambda_i)$ and can therefore be written as a $am(x) + b$. So, this game has a pure Nash equilibrium. ■

From Theorems 3.3 and 3.4 it follows that two-player games with equal λ or μ have a Nash equilibrium in pure strategies. However, this is not the case in general as Example 3.2 shows. The following theorem discusses 2-player games on a Jackson network on an $A \times B$ grid. The network has AB nodes that may be represented by their coordinates (x, y), $x = 1, \ldots, A,\ y = 1, \ldots, B$. Customers may only route to the neighboring nodes $(x-1, y)$, $(x+1, y)$, $(x, y-1)$ and $(x, y+1)$ provided these nodes are part of the grid. Let μ_i be the service rate of node i, $i = (x, y)$, and let $\lambda^{(j)}$ be the arrival rate of customers for player j, $j = 1, 2$. All routes of the players start at a node at the side of the grid and end at a node on the side of the grid. Furthermore, we only allow routes of minimal length, i.e., containing the minimum number of nodes required to move from source node to sink node so that the length of the possible routes for player j is fixed, say $N^{(j)}$ for player j, $j = 1, 2$. Once again, each player's goal is to minimize its expected sojourn time.

If the source and sink nodes of the players are such that their routes $r^{(j)}$ for player j, $j = 1, 2$, intersect in at least one node, i.e., $r^{(1)} \cap r^{(2)} \notin \emptyset$, then a strategy profile p is feasible if $\lambda^{(1)} + \lambda^{(2)} < \mu$. The mean sojourn times of the customers of the players are

$$
\begin{aligned}
f^{(1)}(p) &= \sum_{\{i \in r^{(1)} \setminus r^{(2)}\}} \frac{1}{\mu_i - \lambda^{(1)}} + \sum_{\{i \in r^{(1)} \cap r^{(2)}\}} \frac{1}{\mu_i - (\lambda^{(1)} + \lambda^{(2)})}, \\
f^{(2)}(p) &= \sum_{\{i \in r^{(2)} \setminus r^{(1)}\}} \frac{1}{\mu_i - \lambda^{(2)}} + \sum_{\{i \in r^{(1)} \cap r^{(2)}\}} \frac{1}{\mu_i - (\lambda^{(1)} + \lambda^{(2)})}.
\end{aligned}
$$

For the case $\mu_i = \mu$, $i \in \mathcal{C}$, if $\lambda^{(1)} + \lambda^{(2)} < \mu$ according to Theorem 3.4, the game has a pure Nash equilibrium. For this Nash equilibrium the intersection of the routes will be in a single node: $r^{(1)} \cap r^{(2)} = \{j\}$, since sharing multiple nodes clearly increases the mean sojourn time of the customers of both players. The following theorem shows that the game has a pure-strategy Nash equilibrium when we vary the service rates $\mu_i \in \{\mu, k\mu\}$ for k sufficiently small to guarantee that routes will intersect in a single point.

Theorem 3.5. *The 2-player game on a Jackson network on a grid with $\mu_i \in \{\mu, k\mu\}$, for $k > 1$, $i \in C$, and $\lambda^{(1)} = m\lambda$, $\lambda^{(2)} = \lambda$, $m < 1$, such that $(m+1)\lambda < \mu$ and $k\mu < \mu + m\lambda$, has a pure Nash equilibrium.*

Proof. The condition $k\mu < \mu + m\lambda$ guarantees that routes will intersect in at most 1 node, because then

$$\frac{1}{k\mu - (m+1)\lambda} > \frac{1}{\mu - \lambda} > \frac{1}{\mu - m\lambda}.$$

Let player 1 select a route $r^{(1)}$ that minimizes the sojourn time of its customers, and let player 2 select a route $r^{(2)}$ that is a best response to the route of player 1. If these routes do not intersect, then the strategy profile selecting these routes is a pure Nash equilibrium.

Now assume these routes intersect in node $i \in r^{(1)} \cap r^{(2)}$. There are two possible values for the service rate at the intersection of the routes: (1) $\mu_i = \mu$ and (2) $\mu_i = k\mu$. For each case we will consider player 1's best-response $r'^{(1)}$ to the route of player 2, which will also intersects with $r^{(2)}$ in one node. Let that be node $i' \in r'^{(1)} \cap r^{(2)}$.

Case (1): $\mu_i = \mu$. If $i = i'$ then the original route profile $(r^{(1)}, r^{(2)})$ is a pure Nash equilibrium. If $i \neq i'$ assume first that $\mu_{i'} = \mu$. Note that player 2's payoff is unaffected. Because the original route selected by player 1 was optimal without considering the influence of customers of player 2, the part of the route of player 1 that does not intersect with the route of player 2 cannot have a smaller sojourn time. Hence, the payoff of player 1 using route $r'^{(1)}$ cannot be below that of using $r^{(1)}$ and it will also not be larger because it is a best-response. Hence, the profile $(r'^{(1)}, r^{(2)})$ is a pure Nash equilibrium.

Second, assume $\mu_{i'} = k\mu$. This contradicts the optimality of $r^{(1)}$: if such a node i' is part of player 1's best response, then it should also have been part of the original route. So, this cannot happen.

Case (2): $\mu_i = k\mu$. We will show that under the conditions of the theorem these routes yield a pure Nash equilibrium.

If $i = i'$ then the original route profile $(r^{(1)}, r^{(2)})$ is a pure Nash equilibrium. If $i \neq i'$ then assume first that $\mu_{i'} = k\mu$. This will not affect the sojourn time of player 2. Following similar arguments as for case (1), the part of the route of player 1 that does not intersect with the route of player 2 cannot have a smaller sojourn time. Hence, the route profile $(r'^{(1)}, r^{(2)})$ is a pure Nash equilibrium.

Second, assume $\mu_{i'} = \mu$. For player 1, the new route $r'^{(1)}$ only results in a reduced sojourn time if the other nodes compensate for the increased sojourn time in node i'. Observe that the original route of player 1 was optimal. Therefore, the new route for player 1, that intersects in node i' with service rate $\mu_{i'} = \mu$, cannot contain more nodes with high service rate $k\mu$ than the original route. If the new route has one node with low service rate more than the original route, than the sojourn time of player 1 should have increased. This contradicts our assumption of $r'^{(1)}$ being a best-response. If the new route has the same number of nodes with high service rate, and therefore $r'^{(1)} \setminus r^{(2)}$ contains one extra high service rate node, the difference in sojourn times for player 1 $(r'^{(1)} - r^{(1)})$ is

$$\Delta f^{(1)} = \left(\frac{1}{\mu - (m+1)\lambda} + \frac{1}{k\mu - m\lambda}\right) - \left(\frac{1}{k\mu - (m+1)\lambda} + \frac{1}{\mu - m\lambda}\right),$$

which can readily be seen to be positive for $k > 1$, and $(m+1)\lambda < \mu$. But a positive difference means an increase in sojourn time. This contradicts the assumption that the new route is a best response to player 2's route. Hence, this situation cannot occur. ■

Observe that the assumption $\lambda^{(1)} < \lambda^{(2)} = \lambda$ is without loss of generality. The restriction $\mu \in \{\mu, k\mu\}$ in Theorem 3.5 is a sufficient condition to guarantee the existence of a pure Nash equilibrium. However, experimental results show that pure Nash equilibrium exists in most of the random networks we constructed.

3.5 Concluding remarks

In this chapter we have considered a new type of games: non-cooperative games on a Jackson network, where multiple players route through a network while they are all minimizing there own sojourn time. We have considered two cases: the game with continuous strategy space and the game with discrete strategy space.

In the continuous case, each player is allowed to distribute his arrival rate over multiple fixed routes. This results in a convex game with continuous strategy space for which it can be proven that a pure Nash equilibrium exists. This Nash equilibrium can be found by using a best response algorithm.

In the discrete case, all players are only allowed to select one single route resulting in a game with finite strategy space. This game is a special variant of a weighted congestion game. When the arrival rates of all players are equal or, for two-player games, when the service rates are equal, we can prove that a Nash equilibrium always exists in pure strategies. However, for the general case, we give a counter example to demonstrate that a pure Nash equilibrium does not always exist. We show that one can construct instances on a grid with unequal arrival and service rates for which pure Nash equilibria do exist.

Part II

Dynamic information security games

CHAPTER 4

Solving partially observable agent-intruder games with an application to border security problems

This chapter resulted in [73].

4.1 Introduction

Security forces, like coast guards, have to protect areas or high value assets against intruders that may have malicious intentions. Unfortunately, drug smuggling or illegal fishing units often have the means to prevent detection, and patrols or border controls of security forces will affect their course of action. In turn, security forces may use sensors or other information sources to locate intruders and to adjust their patrols, resulting in a chain of reactions and counter reactions. For both players, operating in an optimal way taking into account possible reactions of the other player is of paramount importance. For the security forces this means that patrol strategies or border control strategies at possible entry locations have to be found, in order to intercept the intruder or to prevent the illegal crossing of borders.

Several game-theoretic models have been developed to provide solutions for these kinds of problems (e.g., [62, 81, 78, 99]). In this paper we deviate from the traditional stochastic game assumption that both players have full information about the position of the other player. Instead, we consider the realistic situation where players only have partial information about the other player's position. These problems, where each player has partially observable information about the position of the opponent, can be modeled as dynamic search games on a graph between security forces and an intruder, where the possible border entry points, the area or high value assets that require protection are represented. The intruder is trying to either cross the border or to attack the high value assets (target nodes in the graph) while the security forces (agents) are aiming to prevent this. By modeling this as a partially observable agent-intruder game (POAIG), strategies can be identified to efficiently deploy the scarce security systems and personnel, while taking into account the intruder's behavior.

Game theoretical models with incomplete information have been developed to model intelligent adversaries and uncertainty about states or actions. For example in [48], the authors describe a hider-seeker game with incomplete information about

the starting position of the hider. This game is modeled as a Bayesian game where both players choose a path at the start of the game. Bayesian games are often used when one has incomplete information about the other player's type (e.g., [82, 101]). In contrast, we consider the game in a dynamic setting where decisions have to be made during multiple time steps.

The POAIG shows similarities with a partially observable stochastic game (POSG). A POSG is an extension of a partially observable Markov decision process (POMDP), which is a generalization of a Markov decision process (MDP) where the agent does not have full information about the state of the system [117]. In [31], for example, the author uses a POMDP to model an agent searching for a moving intruder, whose movements follow a known Markov process. In a POMDP, the agent does not observe the state, but makes observations after each action. These observations give some information about the state of the system and using this information, the agent can construct a belief about what the true state of the system is. By transforming a POMDP with a finite state space to an MDP with a continuous state space, where the state space is the set of all probability distributions over the original state space, it can be shown that there exists an optimal value for the POMDP and that optimal strategies exist [30].

In the literature, different formulations of POSGs can be found. In [44], the authors introduced a POSG where both players observe the state. All actions and observations in previous steps are common knowledge for both players. By analogy with the proof of [30], the authors show that a Nash equilibrium exists for this game by transforming the POSG into a stochastic game with complete information and a continuous state space. Since the state space of the constructed stochastic game is continuous, finding a Nash equilibrium is not possible using standard methods for stochastic games, since these methods enumerate over all states.

The POAIG considered in this paper is different from the POSG described in [44] as they assumed that both players have complete information about the observations and actions of all players in the previous steps. As such, each player knows the actions and observations of the other players in contrast to the POAIG where the intruder's position and actions are not known by the security forces. In [12], the authors also assume asymmetric information of both players, but the players do not play the game simultaneously, while they do in our game.

In [32, 49, 83], several solution methods for POSGs are introduced. In [49], the authors propose a dynamic program to solve the finite time POSGs using the normal form representation. However, the size of the strategy set is double exponential in the time and, even after deleting dominated strategies, only games with a small state space or a short time horizon can be considered. In [32]. the authors propose an approximation algorithm to solve POSGs in which both players solve a Bayesian game during every time step, assuming complete information in the next step. In that paper, the authors only discuss POSGs with common payoffs, which can also be seen as decentralized POMDPs (dec-POMDP) ([15]). A difference between a dec-POMDP and the POAIG that we describe in this paper, is that we consider games in which both players have opposite goals while in dec-POMDPs the players have a common goal. As a consequence, it might be beneficial for the players in a POAIG not to be observed by their opponent while this is not the case for dec-POMDPs. In [83], the authors suggest a transformation from POSGs to a traditional stochastic game and suggest to solve these by discretizing the state space. In this paper, we will describe

a solution approach using the sequence form representation of games in order to solve POAIGs efficiently.

In [54], the authors introduce a partial-information game on a Markov chain similar to the POSG in which the players have their own observations but without information about the previous actions of their opponent. In [54], both players only do observations, but the observations can be modeled such that the player's own position is contained in the observations. The authors prove the existence of a Nash equilibrium for games with a finite time horizon and give a necessary and sufficient condition for a pair of strategies to be a Nash equilibrium. In [16], a similar model for multiple players is considered and the existence of a Nash equilibrium for a finite time horizon is proven. The POAIG described in this paper is closely related to this game in the sense that the players know their own position and observe the position of the other player.

The main contribution of this paper is proving the existence of ϵ-optimal strategies for POAIGs with an infinite time horizon. We develop an approximation algorithm based on belief functions that can be used to find approximate solutions for these games. To prove the existence of ϵ-optimal strategies for POAIGs with an infinite time horizon, we use results obtained for POAIGs with a finite time horizon. For POAIGs with a finite time horizon we show that a solution framework, common to solving extensive form games, may also be used effectively. To this end, the game is reformulated as a sequence form game [67] and the column generation approach is applied to speed up the solution process [17, 48]. As security forces often are faced with partial information, POAIGs provide decision support for developing patrol strategies and determining optimal locations for scarce resources like sensors. A set of illustrative examples underpins the potential of our approach.

The paper is organized as follows. In the next section, we give a model description of agent-intruder games with complete information and introduce the POAIG. Section 4.3 describes how to find Nash equilibria for POAIGs with a finite time horizon. In Section 4.4, we consider POAIGs with an infinite time horizon. Next, in Section 4.5, we consider several applications of POAIGs and show numerical and computational results. Finally, in Section 4.6, we present our conclusions.

4.2 Model description

This section gives a description of the partially observable agent-intruder game (POAIG). This game originates from the traditional stochastic game in which the state of the system is known to both players. The agent-intruder game (AIG) with complete information is presented in Section 4.2.1. Section 4.2.2 extends this game to the POAIG.

4.2.1 Agent-intruder game

The AIG is modeled as a traditional stochastic game ([96], Chapter 5) with complete information.

Definition 4.1. An agent-intruder game AIG $= (S, A, P, R, s_1)$ is a stochastic game defined as follows:

- The state space is $S = S^{(1)} \times S^{(2)}$, where $S^{(i)}$ is the finite state space for player i, $i = 1, 2$. In state $s = (s^{(1)}, s^{(2)}) \in S$, $s^{(i)}$ records the position of player i,

$i = 1, 2$.

- The action set is $A(s) = A^{(1)}(s^{(1)}) \times A^{(2)}(s^{(2)})$, $s = (s^{(1)}, s^{(2)}) \in S$, where $A^{(i)}(s^{(i)})$ is the finite action set for player i in position $s^{(i)}$, $i = 1, 2$. In action $a = (a^{(1)}, a^{(2)})$, $a^{(i)}$ is the action of player i, $i = 1, 2$.

- The transition probabilities are $P(\bar{s}|s, a) = P^{(1)}(\bar{s}^{(1)}|s^{(1)}, a^{(1)})$ $\cdot P^{(2)}(\bar{s}^{(2)}|s^{(2)}, a^{(2)})$, where $P^{(i)}(\bar{s}^{(i)}|s^{(i)}, a^{(i)})$ is the probability of a transition from position $s^{(i)}$ to position $\bar{s}^{(i)}$ given action $a^{(i)}$ for player i, $i = 1, 2$.

- $R^{(i)}(s, a)$ is the finite reward for player i in state $s = (s^{(1)}, s^{(2)})$ given action $a = (a^{(1)}, a^{(2)})$, $i = 1, 2$.

- $s_1 = (s_1^{(1)}, s_1^{(2)})$ is the initial state.

The AIG defines a stochastic process: Let s_t and a_t be the random variables describing the state and action respectively at time t, $t \geq 1$. □

Remark 4.1. For the AIG, optimal strategies exist and can be found using an iterative approach (see Section 5.3 in [96]). A stochastic game has a finite value if the game ends within finite time with probability one [96]. This is achieved by introducing an absorbing state or by stopping the game after a finite time T:

- Adding an absorbing state s_A to the state space means that the game ends when the absorbing state is reached. In state s, when taking action a, the game makes a transition to the absorbing state and ends with probability $P(s_A|s, a)$. If $P(s_A|s, a) > 0$, for all $s \in S$, $a \in A(s)$, the game ends after a finite number of steps with probability one. When introducing the absorbing state, the transition probabilities can be changed in two different ways: ensure that $P^{(1)}$ or $P^{(2)}$ are defective or introduce a stopping probability q. In the first case, $P(s_A|s, a) = 1 - \sum_{\bar{s} \in S \setminus s_A} P(\bar{s}|s, a)$. In the second case, $P(s_A|s, a) = q$ and $P(\bar{s}|s, a) = (1 - q)\tilde{P}(\bar{s}|s, a)$, where $\tilde{P}$ is the transition matrix without the absorbing state.

- If we abort the game at finite time T, we introduce a termination reward $R_T^{(i)}$. At time $t < T$, the reward of player i is $R^{(i)}(s, a)$ and at time T the reward of player i, $i = 1, 2$, is $R_T^{(i)}(s, a)$ in state $s = (s^{(1)}, s^{(2)})$ given action $a = (a^{(1)}, a^{(2)})$. □

4.2.2 Partially observable agent-intruder game

In a POAIG, each player has complete information about his own state, but partial information about the state of the other player. To this end, we introduce the observation set $O^{(i)}$ and the observation function $Q^{(i)}(o^{(i)}|s)$, $o^{(i)} \in O^{(i)}$, $i = 1, 2$, that describes the information that player i has on the state of the other player. The POAIG with a finite time horizon is a special case of the game described by [54]. In our game, the players also know a part of the state, namely their own position.

Definition 4.2. A partially observable agent-intruder game POAIG $= (S, A, O, P, Q, R, s_1, \mu)$ is an AIG with the additional elements O, Q and μ defined as follows:

- The observation set is $O = O^{(1)} \times O^{(2)}$, where $O^{(i)}$ is the finite set of all possible observations of player i, $i = 1, 2$. In observation $o = (o^{(1)}, o^{(2)})$, $o^{(i)} \in O^{(i)}$ records the observation of player i, $i = 1, 2$.
- The observation function is $Q^{(i)}$, where $Q^{(i)}(o^{(i)}|s)$ is the probability that player i observes $o^{(i)}$ in state $s \in S$, $i = 1, 2$.
- $\mu = (\mu^{(1)}, \mu^{(2)})$ is the initial distribution of states, where $\mu^{(i)}$ is a probability distribution over all possible positions of the other player $j(\neq i)$, $i = 1, 2$.

The POAIG defines a stochastic process. Let s_t, o_t and a_t be the random variables describing the state, observation and action respectively at time t, $t \geq 1$. □

Remark 4.2. We assume that both players have complete information about the observation set and the observation functions of the other players. □

At the end of this section, we give an example of a finite POAIG to illustrate the various ingredients. For more examples on infinite time POAIGs, we refer to Section 4.5.

The goal for both players is to find strategies that optimize their rewards. In the POAIG, the strategies depend on all the previous observations and actions of the player. Let $\mathcal{F}_t^{(i)} = \sigma(s_n^{(i)}, o_n^{(i)}, a_m^{(i)}, 1 \leq n \leq t, 1 \leq m < t)$, $i = 1, 2$, be the σ-field containing all information about all possible states, actions and observations. $\mathcal{F}_t^{(i)}$ contains all possible history sequences at time t and $\mathcal{F}^{(i)}$ denotes the set of all possible $\mathcal{F}_t^{(i)}, t \in \{1, ..., T\}$. $F_t^{(i)} = \{s_1^{(i)}, o_1^{(i)}, a_1^{(i)}, ..., a_{t-1}^{(i)}, s_t^{(i)}, o_t^{(i)}\} \in \mathcal{F}_t^{(i)}$ is the realized history sequence at time t. The strategies and payoffs are defined as follows:

- The strategy of player i is $\pi^{(i)} \in \Pi^{(i)}$, where $\Pi^{(i)}$ is the set of all possible strategies of player i, $i = 1, 2$, and $\pi^{(i)}(a^{(i)}|\mu, F_t^{(i)})$ is the probability of action $a^{(i)}$ given the initial distribution μ and the history $F_t^{(i)} \in \mathcal{F}^{(i)}$, $i = 1, 2$, $t = 1, ..., T$, $F_t^{(i)} \in \mathcal{F}^{(i)}$.
- The payoff of the POAIG, given the initial distribution μ and strategies $\pi^{(i)}$, $i = 1, 2$, is:

$$V^{(i)}_{\pi^{(1)}, \pi^{(2)}}(\mu) = \mathbb{E}^{\mu}_{\pi^{(1)}, \pi^{(2)}} \sum_{t=1}^{T} R^{(i)}(s_t, a_t), \tag{4.1}$$

  where T might be infinite.

A pair of strategies $\pi^{*(1)}$, $\pi^{*(2)}$ is said to be optimal if:

$$V^{(1)}_{\pi^{*(1)}, \pi^{*(2)}}(\mu) \geq V^{(1)}_{\pi^{(1)}, \pi^{*(2)}}(\mu), \quad \text{for all } \pi^{(1)} \in \Pi^{(1)},$$

and similarly for player 2. The value of the game for player i starting with the initial distribution μ is $V^{(i)}_{\pi^{*(1)}, \pi^{*(2)}}(\mu)$, $i = 1, 2$.

For a POAIG with a finite time horizon T, it can be shown that optimal strategies exist [54]. The optimal strategies are mixed strategies that can be found by enumerating over all possible combinations of actions and observations. We discuss this in Section 4.3.

Example 4.1 (Border security)**.** Consider the game on the graph depicted in Figure 4.1. Player 1 is the agent and player 2 the intruder. The intruder starts at node 1 and aims to cross the border (nodes 12, 13, 14, 15 and 16). The agent starts at the border and tries to prevent the intruder from crossing the border. The game stops if the intruder reaches the border at $T = 4$. The intruder is caught if the intruder and agent are at the same node. The POAIG is described as follows:

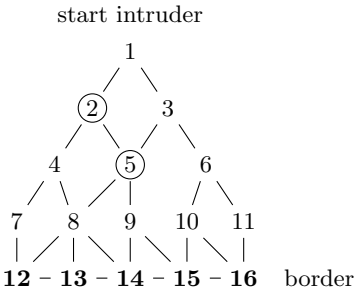


Figure 4.1: Example border security with sensors at nodes 2 and 5.

- The state space of the agent is $S^{(1)} = \{12, 13, ..., 16\}$ and the state space of the intruder is $S^{(2)} = \{1, 2, ..., 16\}$.
- At each step, each player moves to an adjacent node while the agent is also allowed to stay at its current node. The intruder only moves to the border and the agent moves along the border. The action set of each player depends on the current state and the actions uniquely determine the next state. For example, $A^{(1)}(13) = \{12, 13, 14\}$ and $A^{(2)}(5) = \{8, 9\}$, where actions correspond to states that can be reached in one step.
- The transition probabilities are:

$$P^{(i)}(\bar{s}^{(i)}|s^{(i)}, a^{(i)}) = \begin{cases} 1, & \text{if } \bar{s}^{(i)} = a^{(i)}, \\ 0, & \text{otherwise,} \end{cases} \quad i = 1, 2.$$

- The reward for the agent equals 1 if the agent catches the intruder and 0 if the intruder reaches the border without being caught.

$$R^{(1)}(s, a) = \begin{cases} 1, & \text{if } s^{(1)} = s^{(2)}, a \in A^{(1)}(s^{(1)}), \\ 0, & \text{otherwise.} \end{cases}$$

- The agent can observe the intruder at one of the sensor nodes, 2 and 5, or observes nothing, which is denoted by $\emptyset$, so $O^{(1)} = \{\emptyset, 2, 5\}$. The intruder can observe the agent at every node with a small probability, for example by information given by other intruders or by using a camera, or observes nothing, so $O^{(2)} = \{\emptyset, 12, 13, ..., 16\}$.
- We assume that each of the agent's sensors detects the intruder with probability 0.9, while the intruder can see the agent with probability 0.1, resulting in the

following observation functions:

$$Q^{(1)}(o^{(1)}|s) = \begin{cases} 0.9, & \text{if } o^{(1)} = s^{(2)}, s^{(2)} \in \{2,5\}, \\ 0.1, & \text{if } o^{(1)} = \emptyset, s^{(2)} \in \{2,5\}, \\ 1, & \text{if } o^{(1)} = \emptyset, s^{(2)} \notin \{2,5\}, \\ 0, & \text{otherwise,} \end{cases} \tag{4.2}$$

$$Q^{(2)}(o^{(2)}|s) = \begin{cases} 0.1, & \text{if } o^{(2)} = s^{(1)}, \\ 0.9, & \text{if } o^{(2)} = \emptyset, \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

- The initial state distribution is:

$$\mu^{(1)}(s) = \mu^{(2)}(s) = \begin{cases} 1, & \text{if } s = (14,1), \\ 0, & \text{otherwise.} \end{cases}$$

□

4.3 Nash equilibrium for finite POAIGs

In this section, we present different formulations to find optimal strategies for the POAIG with a finite time horizon. From now on, we only consider zero-sum games where the agent is maximizing and the intruder is minimizing the payoff, so $R(s_t, a_t) = R^{(1)}(s_t, a_t) = -R^{(2)}(s_t, a_t)$, and the game value is $V_{\pi^{(1)},\pi^{(2)}}(\mu)$. Without loss of generality, we assume that the rewards are always positive. Following [54], a Nash equilibrium can be found by constructing the normal form of this game. In Section 4.3.1, we describe how to construct this normal form by enumerating over all possible pure strategies. Optimal strategies can then be found using linear programming (LP). Since the size of the resulting matrix grows exponentially fast in the number of states, observations and actions, we use a column generation method. Finally, in Section 4.3.2, we address the sequence form representation, which provides a different formulation with smaller game matrix. In Section 4.5, we combine both formulations with column generation to speed up the solution process.

4.3.1 Nash equilibria by using the normal form representation

Solving POAIGs with a finite time horizon is computationally intractable as discussed in [49]. To find the optimal value of a finite POAIG, a straightforward approach is to enumerate over all possible combinations of observations and states. We construct the normal form of this game with all possible pure strategies for the intruder represented in the rows and all possible pure strategies for the agent represented in the columns. A mixed strategy can be found by solving the resulting matrix game. Since the state space, action sets and time horizon are finite, our game boils down to a finite matrix game and thus there always exists a mixed strategy Nash equilibrium (see [96], Chapter 2). Moreover, in [54] it is shown that the mixed strategies of this constructed game coincide with the strategies of the finite time horizon POAIG.

The normal form

We describe how the normal form is constructed. A pure strategy $\pi^{(i)} \in \Pi^{(i)}$ is a strategy such that, for every $F_t^{(i)} \in \mathcal{F}^{(i)}$, $t = 1, ..., T$, and given initial distribution μ, $\pi^{(i)}(a^{(i)}|\mu, F_t^{(i)}) \in \{0, 1\}$. Thus, in every pure strategy a unique action is specified for each possible history. Let $\tilde{\Pi}^{(i)} \subset \Pi^{(i)}$ be the set of all possible pure strategies for player i, $i = 1, 2$. $\tilde{M} = [\tilde{m}_{ij}]$ is a matrix of size $|\tilde{\Pi}^{(1)}| \times |\tilde{\Pi}^{(2)}|$ with all pure strategies of the agent in the rows and all pure strategies of the intruder in the columns. The initial distribution of the states is μ. Since we consider a finite POAIG, the number of pure strategies, and therefore $\tilde{M}$, is also finite. The entries of $\tilde{M}$ give the expected payoff for the policies $\pi^{(1)} \in \tilde{\Pi}^{(1)}$ and $\pi^{(2)} \in \tilde{\Pi}^{(2)}$ of the agent and intruder respectively, so $m_{\pi^{(1)}\pi^{(2)}} = V_{\pi^{(1)},\pi^{(2)}}(\mu)$, by Equation (4.1).

Example 4.2 (Normal form strategy)**.** Reconsider the border security problem described in Example 4.1. The size of a single pure strategy is illustrated by the following part of a pure strategy of the agent: start at node 14; during $t = 1$, move to node 13 (nothing is observed); during $t = 2$, move to node 12 if the intruder is observed at node 2 and stay at node 13 if nothing is observed; during $t = 3$, move to node 14 if the intruder is observed in node 2 and 5, and the agent stayed at node 13 if $t = 2$. The complete strategy is rather lengthy but can be completed in a similar way. □

The number of pure strategies can be very large. If for player i the number of states, actions and observations in each state are given by $N_s^{(i)}$, $N_a^{(i)}$ and $N_o^{(i)}$, $i = 1, 2$, respectively, the number of elements in $\mathcal{F}^{(i)}$ is:

$$\sum_{t=1}^{T} (N_s^{(i)})^t (N_o^{(i)})^t (N_a^{(i)})^{t-1}.$$

For given time horizon T, the number of pure strategies is:

$$|\tilde{\Pi}^{(i)}| = (N_a^{(i)})^{\sum_{t=1}^{T} (N_s^{(i)})^t (N_o^{(i)})^t (N_a^{(i)})^{t-1}}.$$

Thus, the matrix grows exponentially in the number of states, observations and actions. Therefore, in Section 4.3.2, we describe the sequence form to decrease the matrix size.

4.3.2 Nash equilibrium using the sequence form representation

In the previous section, we described how the normal form of the game, with all possible pure strategies, may be constructed. In this subsection, we describe a different way of representing the strategies in order to decrease the matrix size. This is called the sequence form representation and was introduced to find a Nash equilibrium in extensive form games by [67] and [124]. In a sequence form game, we do not consider all possible pure strategies, but instead consider all possible sequences of positions, actions and observations of each player. To find optimal strategies for the original game, we consider the realization probability of occurrence of each sequence. In Section 4.5 we discuss the advantage of the sequence form representation over the normal form approach.

The sequence form

Recall that the initial state distribution is μ. Let $\Sigma^{(i)}$ be the set of all possible sequences for player i, $i = 1, 2$. A sequence $\sigma^{(i)} \in \Sigma^{(i)}$ is defined as an ordered list of positions, observations and actions of player i, $i = 1, 2$. A sequence $\sigma^{(i)} \in \Sigma^{(i)}$ always ends with an action. The length of a sequence is defined by its number of actions, thus a sequence of length t is:

$$\sigma_t^{(i)} = \{s_1^{(i)}, o_1^{(i)}, a_1^{(i)}, ..., s_t^{(i)}, o_t^{(i)}, a_t^{(i)}\},$$

and $\emptyset$ denotes the empty sequence. For notational convenience, we omit the superscript (i) for the states, actions and observations since they always have the same index as $\sigma^{(i)}$. $\Sigma_t^{(i)}$ is the set of all possible sequences for player i, $i = 1, 2$, with length t. Let $S^{(i)}(\sigma^{(i)})$ and $O^{(i)}(\sigma^{(i)})$ be the sets of all possible positions and observations that can follow after sequence $\sigma^{(i)}$. Let $\sigma_{so}^{(i)}$ be the sequence $\sigma^{(i)}$ with additional position $s^{(i)}$ and observation $o^{(i)}$ and let $\sigma_{so}^{(i)}a$ be the sequence $\sigma_{so}^{(i)}$ with additional action $a^{(i)}$.

Example 4.3 (Sequence form representation)**.** Reconsider the border security problem described in Example 4.2. The agent starts at node 14 and the intruder at node 1, so the state at $t = 1$ is given by $s_1^{(1)} = 14$ and $s_2^{(1)} = 1$. The agent observes nothing in the first step and suppose that the agent decides to move to node 13. The corresponding sequence $\sigma^{(1)}$ is then $\{14, \emptyset, 13\}$. Now suppose the agent observes the intruder in node 2 and decides to move to node 12. Then, the corresponding sequence is $\{14, \emptyset, 13, 13, 2, 12\}$. Compared to the normal form representation (see Example 4.2), the strategies in the sequence form can be expressed more compactly. In this example, the present state equals the action taken in the previous time step. That need not be true in general. □

To find optimal strategies, realization probabilities are introduced. The realization probability $r^{(i)}(\sigma^{(i)})$ is the probability that player i plays the sequence of actions in $\sigma^{(i)}$ under the assumption that the observations and positions are as given in $\sigma^{(i)}$. Only sequences that are compatible with the actions and the observations of the other player are added in the sequence form game. The realization probabilities need to satisfy the following constraints:

- The realization probability of the empty sequence equals one:

$$r^{(i)}(\emptyset) = 1. \tag{4.4}$$

- For each combination of positions and observations following sequence $\sigma^{(i)}$, the sum of realization probabilities $r^{(i)}(\sigma_{so}^{(i)}a)$ for all $a^{(i)} \in A^{(i)}(s)$ equals $r^{(i)}(\sigma^{(i)})$:

$$r^{(i)}(\sigma^{(i)}) - \sum_{a \in A^{(i)}(s^{(i)})} r^{(i)}(\sigma_{so}^{(i)}a) = 0, \tag{4.5}$$

  where $\sigma^{(i)} \in \Sigma^{(i)}$, $s^{(i)} \in S^{(i)}(\sigma^{(i)})$, $o^{(i)} \in O^{(i)}(\sigma^{(i)})$.

Given a set of realization probabilities $r^{(i)}$ that satisfy (4.4) and (4.5), a strategy $\pi^{(i)}$ can be constructed in the following way:

$$\pi^{(i)}(a^{(i)}|\mu, F^{(i)}) = \frac{r^{(i)}(\sigma_{so}^{(i)}a)}{r^{(i)}(\sigma^{(i)})}, \qquad r^{(i)}(\sigma^{(i)}) > 0. \tag{4.6}$$

where $F^{(i)} = \sigma_{so}^{(i)}$. If $r^{(i)}(\sigma^{(i)}) = 0$, $\pi^{(i)}(a^{(i)}|\mu, F^{(i)})$ equals 1 if $a^{(i)}$ is the first element of $A^{(i)}$ and 0 otherwise. The strategy $\pi^{(i)}$ follows from the constraints on $r^{(i)}$. Note that (4.6) gives a one-to-one correspondence between strategies and realization probabilities.

Example 4.4 (Realization probabilities)**.** Reconsider the border security problem described in Example 4.3. The agent observes nothing in the first step and suppose that the agent's strategy is to move to node 13 with probability 0.5 and to stay at node 14 with probability 0.5 each. Then the realization probabilities of the sequences $\{14, \emptyset, 13\}$ and $\{14, \emptyset, 14\}$ are 0.5. Suppose that the agent has moved to node 13 and observed the intruder in node 2 at $t = 2$. If his strategy is moving to node 14 with probability 0.3 and moving to node 12 with probability 0.7 then the realization probabilities of the sequences $\{14, \emptyset, 13, 13, 2, 14\}$ and $\{14, \emptyset, 13, 13, 2, 12\}$ are 0.15 and 0.35 respectively. □

In a sequence form game, the payoff function g is the expected payoff for each combination of sequences. The payoff function $g : \Sigma^{(1)} \times \Sigma^{(2)} \to \mathbb{R}$ is defined such that $g(\sigma^{(1)}, \sigma^{(2)})$ is the expected payoff of $\sigma^{(1)} \in \Sigma^{(1)}$ and $\sigma^{(2)} \in \Sigma^{(2)}$. The expected payoff of two sequences $\sigma_t^{(1)}$ and $\sigma_t^{(2)}$ of the same length equals:

$$g(\sigma_t^{(1)}, \sigma_t^{(2)}) = R((s_l^{(1)}(\sigma_t^{(1)}), s_l^{(2)}(\sigma_t^{(2)})), (a_l^{(1)}(\sigma_t^{(1)}), a_l^{(2)}(\sigma_t^{(2)}))) \cdot p_{\sigma_t^{(1)}\sigma_t^{(2)}}, \quad (4.7)$$

where $s_l^{(i)}(\sigma_t^{(i)})$ and $a_l^{(i)}(\sigma_t^{(i)})$ are the last position and action in sequence $\sigma_t^{(i)}$ and $p_{\sigma_t^{(1)}\sigma_t^{(2)}}$ is the probability that the observations and states are as given in $\sigma_t^{(1)}$ and $\sigma_t^{(2)}$:

$$p_{\sigma_t^{(1)}\sigma_t^{(2)}} = \mu(s_1) \prod_{t'=2}^{t} P(s_{t'}|s_{t'-1}, a_{t'-1}) \prod_{t'=1}^{t} Q^{(1)}(o_{t'}^{(1)}|s_{t'}^{(1)}) Q^{(2)}(o_{t'}^{(2)}|s_{t'}^{(2)}), \quad (4.8)$$

where $s_{t'}$, $o_{t'}$ and $a_{t'}$, $t' = 1, ..., t$, are determined in the sequences $\sigma_t^{(1)}$ and $\sigma_t^{(2)}$. The matrix M of size $|\Sigma^{(1)}| \times |\Sigma^{(2)}|$ contains the expected payoffs for all combinations of sequences with all sequences of the agent and the intruder represented in the rows and columns respectively.

Example 4.5 (Payoff functions)**.** Consider the border security problem described in Example 4.4. The reward of two sequences is positive only when the length of the sequence is T. For all other sequences, it is not possible that the agent and the intruder are at the same node. If in such sequences $a_T^{(1)} = a_T^{(2)}$ then $g(\sigma_T^{(1)}, \sigma_T^{(2)}) = 1$ and otherwise $g(\sigma_T^{(1)}, \sigma_T^{(2)}) = 0$. □

Given the (feasible) realization plans of both players, the expected payoff of the game is given by $(r^{(1)})^\top M r^{(2)}$. Applying linear programming, as discussed in [67], a Nash equilibrium for the sequence form game can be found. Moreover, using the next lemma, it follows that the value of the sequence form game coincides with the value of the POAIG described in Section 4.2.

Lemma 4.1. *A Nash equilibrium for the sequence form game corresponds to a Nash equilibrium for the POAIG with a finite time horizon, with the same value.*

All proofs can be found in Section 4.7.1.

4.4 Approximate solutions for infinite POAIGs

In the previous section, we considered POAIGs with a finite time horizon. In this section, we consider infinite time horizons. First, we prove the existence of ϵ-optimal strategies in Section 4.4.1. Since there are no algorithms to find exact solutions of POAIGs with an infinite time horizon, we introduce in Section 4.4.2 a new approach to approximate optimal strategies.

4.4.1 Existence of ϵ-optimal strategies

In [44], it is shown that there exists a Nash equilibrium for POSGs. That proof is by analogy with the proof of the existence of a value for POMDPs. However, POAIGs differ from POSGs because, contrary to POSGs, the belief functions are not common knowledge for POAIGs. In this section, we prove that there exist ϵ-optimal strategies for the POAIG with an infinite time horizon.

To ensure that the game stops after a finite number of steps with probability one, we introduce an absorbing state s_A. Let M_t, $t \leq T$, be the matrix with the expected payoff for all sequences with length t. M_t has size $|\Sigma_t^{(1)}| \times |\Sigma_t^{(2)}|$. A finite POAIG can be formulated as a sequence form game as described in Section 4.3.2:

$$M = \begin{array}{c} \\ t=1 \\ t=2 \\ \vdots \\ t=T \end{array} \begin{array}{c} \begin{array}{cccc} t=1 & t=2 & \cdots & t=T \end{array} \\ \begin{pmatrix} M_1 & 0 & \dots & 0 \\ 0 & M_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & M_T \end{pmatrix} \end{array}.$$

Since M is a finite matrix, this game always has a Nash equilibrium, which can be found by solving a linear program. Moreover, if $r^{(1)}$ and $r^{(2)}$ are realization probabilities corresponding to M, then $r_t^{(1)}$ and $r_t^{(2)}$ are the parts of the realization probabilities that correspond to M_t. The following lemma gives an upper bound for $(r_t^{(1)})^\top M_t r_t^{(2)}$. The proof is provided in Section 4.7.1.

Lemma 4.2. *Let M_t be the expected payoff matrix for sequences of length t and let $r_t^{(1)}$ and $r_t^{(2)}$ be the corresponding realization probabilities. The expected payoff obtained at time t is bounded as follows:*

$$|(r_t^{(1)})^\top M_t r_t^{(2)}| \leq (1 - p_{s_A})^{t-1} \max_{a \in A, s \in S} R(s, a), \tag{4.9}$$

where $p_{s_A} = \min_{s,a} P(s_A|s, a)$.

Lemma 4.2 can be applied to prove the existence of ϵ-optimal strategies for the POAIG with an infinite time horizon. Let M_∞ be the operator defined by:

$$M_\infty = \begin{array}{c} \\ t=1 \\ t=2 \\ \vdots \\ t=T \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccc} t=1 & t=2 & \cdots & t=T & \cdots \end{array} \\ \begin{pmatrix} M_1 & 0 & \dots & 0 & \cdots \\ 0 & M_2 & \dots & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & M_T & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \end{array}.$$

Since the general existence theorem of Nash equilibria in matrix games only applies to finite matrices, games with an infinite matrix representation do not necessarily have a Nash equilibrium. We prove that there exists a ϵ-equilibrium for this matrix game. In an ϵ-equilibrium, none of the players can improve their payoff more than ϵ by deviating from there current strategy. The proof can be found in Section 4.7.1.

Theorem 4.1. *There exist ϵ-optimal strategies for the POAIG with an infinite time horizon.*

Note that it is more intuitive to prove the existence of a Nash equilibrium in analogy with the proof for POMDPs in [30] and POSGs in [44]. However, in [30] and [44], the authors transform the model to a complete information model with a continuous state space. The new state is then the belief over the original state and in the POSG the belief state is the same for all players. In our model, both players have a different belief state and they do not know each other's belief. Therefore, we cannot follow the proof of [30] and [44].

4.4.2 Calculate optimal strategies

As discussed in the previous section, finding optimal strategies by enumerating over all possible pure strategies is computationally intractable. Moreover, this method is restricted to POAIGs with a finite time horizon. In this section we introduce an algorithmic approach to approximate optimal solutions in each step of the game using the belief of the players. The belief of a player is a probability distribution over the state space S. The idea of a belief state was introduced to reformulate the POMDP as an MDP with a continuous state space [117]. We first provide a description of the belief functions and then introduce the approximation algorithm.

Belief functions

In the POAIG, neither player has complete information about the state of the system, since they only know their own state and not the state of the other player. As such we will introduce in this section the player's belief which provides for each player a probability distribution over all possible states of the other player. As the name suggests, the player's belief models the player's conviction of what the true state is, based on their own position, the observations they have made so far and the initial distribution. Let $\mu_t^{(i)}$ be the belief of player i at time t. The belief at time 1 is the initial distribution $\mu = (\mu^{(1)}, \mu^{(2)})$, which is common knowledge. The belief $\mu_t^{(i)}(s)$ of player i is the probability $P(s_t | o_k^{(1)}, k = 1, ..., t, a_k, k = 1, ..., t-1)$ that the state is s given the previous observations $o_k^{(i)}$ and actions a_k, $k = 1, ..., t-1$. After each action and new observation, the belief of player i is updated according to the last observation of player i, $i = 1, 2$. The belief at time t relates to the believe at time $t-1$ in the following way:

$$\begin{aligned}
&\mu_t^{(1)}(s_t | o_k^{(1)}, k = 1, ..., t, a_k, k = 1, ..., t-1) \\
&= \frac{P(o_t^{(1)} | s_t) \sum_{s_{t-1}} P(s_t | s_{t-1}, a_{t-1}) \mu_{t-1}^{(1)}(s_{t-1} | o_k^{(1)}, k \leq t-1, a_k, k \leq t-2)}{\sum_{s'_t} P(o_t^{(1)} | s'_t) \sum_{s_{t-1}} P(s'_t | s_{t-1}, a_{t-1}) \mu_{t-1}^{(1)}(s_{t-1} | o_k^{(1)}, k \leq t-1, a_k, k \leq t-2)},
\end{aligned}$$

where $P(s_t|s_{t-1}, a_{t-1}) = P(s_t^{(1)}|s_{t-1}^{(1)}, a_{t-1}^{(1)})P(s_t^{(2)}|s_{t-1}^{(2)}, a_{t-1}^{(2)})$ and $P(o_t^{(1)}|s_t) = Q^{(1)}(o_t^{(1)}|s_t)$. This belief function is derived using Bayes' rule. The complete derivation is given in Section 4.7.2.

In games with complete information, players also take the behavior of other players into account while optimizing their own payoff. As the belief of the other player is not known, we are also interested in the belief of the other player to predict his strategy. Since the observations of the opponent are unknown, each player makes a guess about the observation of the other player and uses this to make a guess about the other player's belief, which we call the counter belief. The counter belief, $\tilde{\mu}_t^{(i)}(\hat{s})$, is what player i believes that player j's beliefs about state $\hat{s}$ are. The counter belief at time 1 is the initial distribution and it follows that at that point the belief of one player equals the counter belief of the other player. The set of possible observations and the observation functions are common knowledge; these sets are used to construct and update the counter belief. After $t = 1$, in each round the belief and counter belief are updated by each player by using their own observations and actions. For the actions, they make a guess based on their belief and counter belief as described in Section 4.4.2. The counter belief of player 1 is defined as:

$$\begin{aligned}
&\tilde{\mu}_t^{(1)}(\hat{s}_t|o_k^{(1)}, k \leq t, a_k, k \leq t-1) \\
&= \sum_{o_t^{(2)}} \sum_{s'_t} P(o_t^{(2)}|s'_t)\mu_t^{(1)}(s'_t|o_k^{(1)}, k \leq t, a_k, k \leq t-1) \\
&\times \frac{P(o_t^{(2)}|\hat{s}_t)\sum_{\hat{s}_{t-1}} P(\hat{s}_t|\hat{s}_{t-1}, a_{t-1})\tilde{\mu}_{t-1}^{(1)}(\hat{s}_{t-1}|o_k^{(1)}, k \leq t-1, a_k, k \leq t-2)}{\sum_{\bar{s}} P(o_t^{(2)}|s'_t)\sum_{\hat{s}_{t-1}} P(s'_t|\hat{s}_{t-1}, a_{t-1})\tilde{\mu}_{t-1}^{(1)}(\hat{s}_{t-1}|o_k^{(1)}, k \leq t-1, a_k, k \leq t-2)},
\end{aligned}$$

and similarly for player 2.

Note that we assumed for the definition of the belief and counter belief functions that the actions are known. However, when updating the belief and counter belief functions the players do not know the exact actions of their opponent. What is required is a strategy which gives a probability distribution over the actions; the approximation algorithm described in the next section provides this strategy. This can be taken into account by replacing the actions $a_{t-1}^{(i)}$, $i = 1, 2$, in the belief and counter belief functions by strategy $\pi_{t-1}^{(i)}$ and $P^{(i)}(s^{(i)}|\bar{s}^{(i)}, a_{t-1}^{(i)})$ by:

$$P^{(i)}(s^{(i)}|\bar{s}^{(i)}, \pi_{t-1}^{(i)}) = \sum_{a_{t-1} \in A} P^{(i)}(s^{(i)}|\bar{s}^{(i)}, a_{t-1}^{(i)})\pi_{t-1}^{(i)}(a_{t-1}),$$

where $\pi_{t-1}^{(i)}$, $i = 1, 2$, is the strategy of player i at time $t-1$. In the next section, we discuss how the belief and counter belief can be used to find these strategies.

Approximation algorithm

In this section, we introduce an algorithm to approximate optimal strategies for infinite time horizon POAIGs. This algorithm constructs during each time step a Bayesian game based on the belief functions and payoffs obtained from the complete information game. This algorithm is an extension of the algorithm proposed by [65] as we explicitly consider actions when updating the belief functions.

Algorithm 4 Approximation for games with an infinite time horizon by using belief functions

1: Calculate the values for the agent-intruder game, using an iterative algorithm as explained in [96]. Let $A^{(s,r)}$ be the payoff matrix if the intruder is in state s and the agent in state r using the values from the game with complete information.
2: Set $t = 0$, $\mu_t^{(i)} = \tilde{\mu}_t^{(i)} = \mu$.
3: Calculate the optimal strategies for the agent for the Bayesian game with payoff matrix A and probability distributions $\mu_t^{(i)}, \tilde{\mu}_t^{(i)}$ for the agent and the intruder respectively by solving the following LP:

$$\begin{aligned} \max_x \quad & \sum_s \mu_t^{(2)}(s) v_s^{(2)} \\ \text{s.t.} \quad & \sum_{i,r} a_{ij}^{(r,s)} \tilde{\mu}_t^{(2)}(r) x_i^{(2)}(r) \geq v_s^{(2)}, \quad \forall j, s, \\ & \sum_i x_i^{(2)}(r) = 1, \qquad \forall r, \\ & x_i^{(2)}(r) \geq 0, \qquad \forall i, r, \end{aligned}$$

where $x_i^{(2)}(r)$ is the probability of choosing action i if the agent is in state r and $v_s^{(2)}$ is the maximal payoff for the agent if the intruder is in state s. Calculate the optimal strategy of the intruder by analogy to that for the agent.
4: Update the belief and counter belief using the functions from the previous section. Stop if an absorbing state is reached, otherwise return to Step 3

The idea of the Algorithm 4 is based on the approach in [32]. At each step, a one-step Bayesian game is solved. In a Bayesian game, there are multiple types and the players do not know against which type they are playing. They do have a probability distribution over all types of the other players. In our case, the types are all possible positions of the other player and the probability distribution over these types are the beliefs and counter beliefs which we have introduced in Section 4.4.2.

The payoff matrices for each step at the incomplete information game are unknown. Therefore, we approximate the payoff matrix for each combination of types by the payoff matrix of the game with complete information. Using this payoff matrix assumes that after one step, there is complete information. Since the payoff of the complete information game is used instead of the real payoff matrix, Algorithm 4 provides an approximation of the optimal strategy. In Section 4.5, we discuss the quality of this approximation. The Bayesian game can be solved at each step by standard techniques via construction of an extensive form game [51] and solving the corresponding matrix game using the LP approach of [105].

In the next section, we discuss the quality of this algorithm by comparing it with the POAIG with a finite time horizon for which the exact optimal strategies are known.

4.5 Applications and computational results

In the previous sections, we introduced the POAIG and methods to find a Nash equilibrium for POAIGs with a finite time horizon and approximate ϵ-optimal strategies

for the POAIG with an infinite time horizon. In this section, we provide numerical and computational results of different applications for finite and infinite time horizon POAIGs.

4.5.1 Finite time horizon POAIGs

Consider the border security problem described in Example 4.1. We first give computational results for the solution concepts introduced in Section 4.3 and describe how this approach can be used to determine optimal sensor locations for the agent.

Benefits of the sequence form game

In this section, we show the benefits of the sequence form representation compared to the normal form representation as described in Section 4.3. The results in this section are implemented in Matlab version R2016b [85] on an Intel(R) Core(TM) i7 CPU, 2.4GHz, 8 GB of RAM. Table 4.1 shows the running times for the network given in Figure 4.1, with different sensor locations. Since the intruder always moves towards the border, this game stops after a finite number of steps ($T = 4$). The first set of columns gives the sensor locations of the agent and the intruder, the second set of columns gives the computation time in seconds and the matrix size when considering the normal form and the third set of columns gives the computation time and the matrix size when considering the sequence form game. The last column gives the game value. Except for the locations of the sensors, the game elements are as described in Example 4.1. Some instances cannot be solved, which is given with a dash. Table 4.1 shows that the matrix size and computation time decrease significantly when using the sequence form game. Also, larger instances can be solved after implementation of the sequence form.

Table 4.1: Normal form versus sequence form.

Sensor locations		Normal form		Sequence form		Game
Agent	Intruder	Time	Size	Time	Size	Value
		0.199	69x17	0.324	107x32	0.200
2,5		621.821	101481x17	0.148	398x32	0.244
8,9,10		4.106	1635x17	0.065	314x32	0.322
	14	2206.080	69x109997	0.200	107x179	0.200
8,9,10	12,...,16	-	1635x109997	0.361	314x1421	0.291
2,...,11	12,...,16	-	-	2.748	2765x1421	0.314

This example also shows how the intruder can gain from the observations. In Table 4.1, only observing node 14 by the intruder does not change the game value (also see the fifth row), but when all nodes at the border can be observed, the game value decreases slightly from 0.322 to 0.291. The intruder observes the agent in an early stage which may enable the intruder to move to a node that is not reachable by the agent.

Applying column generation

In this section, we use a column generation approach to deal with the exponentially large number of pure strategies. In a Nash equilibrium, often only a limited number of pure strategies are used by the agent and the intruder. Therefore, we do not have to take all strategies into account. We first construct a restricted game with only a limited number of strategies and solve this game using linear programming. Thereafter, we calculate for both players the pure best response strategy against the other player's strategy computed in the restricted game and add these pure strategies to the restricted game. This is repeated until the outcome of both players cannot be improved by adding extra strategies. A proof of the optimality of the column generation algorithm for games can be found in [87].

In the sequence form game, complete strategies as used in the normal form are not taken into account. Therefore, the column generation approach is slightly adjusted. This method is developed by [17] to find optimal strategies for extensive form games with imperfect information and a similar method is also used to solve large zero-sum security games by [60]. In order to apply this method developed by [17] to POAIGs, we have made some adjustments described below

For sequence form games, it is not possible to create a restricted set by randomly selecting sequences, since the removal of a sequence also influences other sequences. If sequences are picked at random, it may happen that the outcome of the restricted game does not yield a strategy for each information set. Moreover, it may happen that a sequence in the restricted game is not feasible since the observations in a sequence of one player also depend on the positions in the sequences of the other player. The first problem is solved by using default strategies, such that the first action is chosen for information sets that are not contained in the restricted game.

In [17], the authors use a game tree representation for extensive form games where temporary utility values are assigned to nodes that are not end nodes (an end node is a node where the game ends). We do not use this tree representation in POAIGs and therefore, we do not use temporary utility values to solve the problem of infeasible sequences. Instead, we construct the restricted set such that only feasible sequences are included. This is done by choosing the initial restricted set and by choosing the realization probabilities such that they comply with the default strategy (recall (4.6)).

We compare the computation times with and without the column generation approach for networks with different time horizons as depicted in Figure 4.2. For example, for $T = 6$ the border consists of nodes 32 through 38. The first three columns in Table 4.2 contain the time horizon, the number of nodes in the network and the matrix size of the sequence form game respectively. The following columns contain the sensor locations, the computation time for the sequence form game with and without the implementation of column generation and the value of the game. The rest of the game elements are as given in Example 4.1. Table 4.2 shows that the computation time decreases significantly due to the implementation of the column generation approach.

Using POAIG to find optimal sensor locations

Consider the border security problem in Figure 4.2 with time horizon $T = 5$. The agent starts at node 27 and moves along the border, and the intruder starts at node 1. Except for the sensor locations, the game elements are the same as in the example

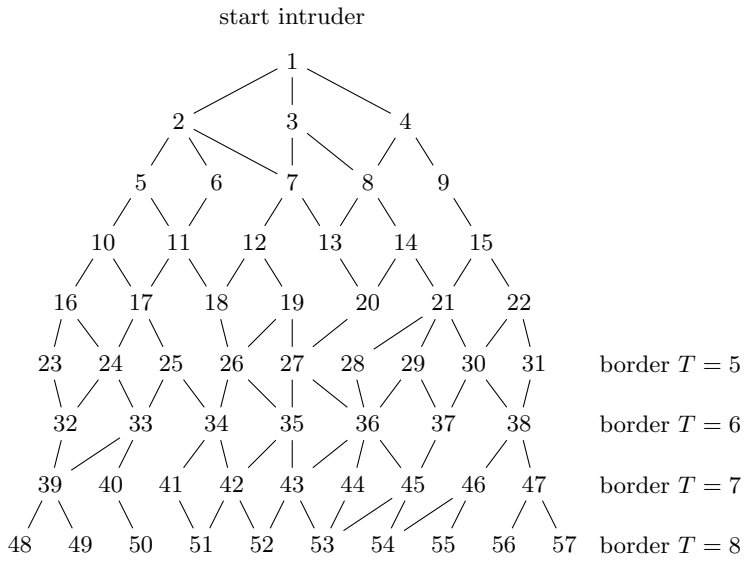


Figure 4.2: Example networks for $T = 5, ..., 8$.

Table 4.2: With and without column generation.

T	Nodes	Size	Sensors		Comp. time (sec)		Game
			Agent	Intr.	With	Without	Value
6	38	59596	2 3 5 6 8 11 12 14	35 37	29	384	0.25
		x5545	16 19 20 25 28 30				
7	47	206684	2 3 5 6 8 14 19 20	43 44	430	17207	0.20
		x108565	25 28 30 34 37				
8	57	642299	2 3 5 6 8 19 20 28	54 57	1018	-	0.20
		x62290	30 34 37 43 44 46				

described above. By modeling this game as a POAIG, we can decide how a given number of sensors have to be placed to optimize the game value for the agent. Table 4.3 gives optimal sensor locations for the agent for $0, 1, 2, 3, 4$ or 5 sensors. There are multiple optimal sensor configurations resulting in the same value: Table 4.3 shows one of these configurations. The results also show, for this game, that considering more than 4 sensors does not have added value, as 0.333 is the maximum game value that can be achieved. This can easily be explained by the fact that from node 21 three edges can be used to reach the border.

Note that the solution of the sequence form game also gives the strategy for the agent and the intruder as a probability distribution for each possible history. For example, in the game with one sensor node an optimal strategy for the agent is to move randomly between nodes 23 to 27 if the intruder is observed in node 2 and between nodes 26 and 31 otherwise. Full display of the optimal strategy is lengthy due to the size of the history space.

Table 4.3: Optimal sensor locations.

# sensors	Sensors Agent	Game Value
0	-	0.111
1	2	0.167
2	2 12	0.200
3	2 7 21	0.250
4	2 7 10 21	0.333
5	2 4 7 16 21	0.333

4.5.2 Infinite time horizon POAIGs

In this section, we consider two applications for infinite time horizon POAIGs. First, we consider a game on a grid in which an intruder is moving towards a target and the agent has sensors at specific nodes. Second, we discuss a similar model without sensors in which the agent is able to observe the nodes in his environment.

Search game on a grid

In this section, we consider a search game on a grid with fixed sensors for the agent. We use this game to test the quality of the approximation algorithm and describe how optimal sensor locations for the agent can be found.

Consider a dynamic search game on a grid between an agent and an intruder, in which the intruder aims to reach a target node from the set of target nodes while the agent aims to prevent and/or delay the intruder's attempt to reach one of the target nodes. At each step, both players may stay at their node or move to an adjacent node and the game stops if either the intruder reaches the target node or the agent catches the intruder, which occurs if they are at the same node at the same moment. In this case the agent receives a payoff of 1. The intruder is subject to breakdown, for example due to a limited amount of fuel or a malfunctioning system. A breakdown may occur with probability 0.05 at each step. If a breakdown occurs, the intruder reveals its position and is caught by the agent, resulting in a payoff of 1 for the agent. As a consequence, delaying the intruder increases the likelihood of breakdown and is beneficial for the agent, which is modelled via a payoff of 0.1 for the agent for each time step the intruder is delayed.

The POAIG on a grid of size 4×4 is described as follows. The state spaces for the agent and the intruder are $S^{(i)} = \{1, 2, ..., 16, s_A\}$, $i = 1, 2$, where s_A is an absorbing state which is reached when the agent and intruder are at the same node, when the intruder is at one of the target nodes or when the intruder breaks down. The action set for each player depends on the current state, for example, $A^{(i)}(1) = \{1, 2, 5\}$, $i = 1, 2$. The transition probabilities are:

$$P(\bar{s}|s, a) = \begin{cases} 0.95, & \text{if } \bar{s}^{(i)} = a^{(i)}, i = 1, 2, \\ 0.05, & \text{if } \bar{s}^{(i)} = s_A, i = 1, 2, \\ 0, & \text{otherwise.} \end{cases}$$

The reward for the agent equals 1 if the agent catches the intruder, -1 if the intruder reaches one of the target nodes, 4 and 13, and 0.1 otherwise. The rewards of the agent

and the intruder sum to 0. The reward for the agent is:

$$R^{(1)}(s,a) = \begin{cases} 1, & \text{if } s^{(1)} = s^{(2)}, a \in A^{(1)}(s^{(1)}), \text{ or } s^{(i)} = s_A, i = 1,2, \\ -1, & \text{if } s^{(2)} \in \{4,13\}, s^{(1)} \neq s^{(2)}, a \in A^{(1)}(s^{(1)}), \\ 0.1, & \text{otherwise.} \end{cases}$$

The agent can observe the intruder at one of the sensor nodes 2 and 5, so $O^{(1)} = \{\emptyset, 2, 5\}$. The intruder does not observe the agent, so $O^{(2)} = \{\emptyset\}$. The observation function is given by (4.2) in Example 4.1.

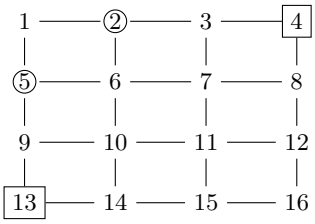


Figure 4.3: Example searching an intruder with sensors at nodes 2 and 5 and targets at 4 and 13.

Approximation algorithm for infinite time horizon POAIGs We cannot compute the exact solution of the infinite time horizon game, therefore we will compare the performance of approximation Algorithm 4 with the exact solution of the corresponding finite time sequence form game. For both the exact solution and the algorithmic approximations, we use a finite number of time steps, even if the intruder is not caught or did not reach the target node. This comparison provides insight into the quality of our approximation algorithm. To calculate the expected average game value when using the approximation algorithm, we generated 1000 sample paths using discrete event simulation, following the steps of Algorithm 4. The results for different initial distributions of the states are given in Table 4.4. The results in the table show that the approximation algorithm gives a good approximation if the initial position is known (Case 1), if there is a large probability that the initial position of the intruder is known (Case 2), or if the initial position is narrowed down to a small area (Cases 4 and 5). If the initial states may be far apart (Case 3) the approximation algorithm results in an error in the game value.

Table 4.4: Comparison of approximation with exact solution ($T = 5$).

case	Initial distribution	Game Value	
		approximation	exact
1	$\mu((6,1)) = 1$	0.1976	0.2233
2	$\mu((6,1)) = 0.9, \mu((6,16)) = 0.1$	0.109	0.110
3	$\mu((6,1)) = 0.5, \mu((6,16)) = 0.5$	-0.170	0.067
4	$\mu((6,1)) = 0.9, \mu((5,1)) = 0.1$	0.226	0.255
5	$\mu((6,1)) = 0.5, \mu((5,1)) = 0.5$	0.269	0.313

Using POAIG to find the agent's optimal start position with fixed sensors
Now assume that there are sensor nodes for the agent at nodes 2, 7, 9 and 15. Using approximation Algorithm 4, we can determine an optimal strategy for the agent. The game values are given in Table 4.5.

Table 4.5: Optimal start position of the agent.

Initial distribution	Targets	Start$_A$	Value
$\mu^{(2)}(1) = 0.2, \mu^{(2)}(2) = 0.3, \mu^{(2)}(6) = 0.3, \mu^{(2)}(10) = 0.2$	4 16	3	0.183
$\mu^{(2)}(5) = 0.1, \mu^{(2)}(6) = 0.3, \mu^{(2)}(13) = 0.5, \mu^{(2)}(14) = 0.1$	4 16	12	0.431
$\mu^{(2)}(5) = 0.1, \mu^{(2)}(6) = 0.3, \mu^{(2)}(13) = 0.5, \mu^{(2)}(14) = 0.1$	3 8	6	0.913

The game value and optimal strategy depend on the starting position of the agent. In Table 4.5 the agent's start position that gives the highest game value is displayed, given different initial distributions and targets for the intruder.

Since the strategies of both the agent and the intruder depend on all previous observations and states, we cannot display the full strategies of the players. However, Table 4.5 shows that it is good to start close to (one of) the target nodes or between the target nodes and the expected position of the intruder. When nothing is observed, the agent moves in the area around the targets and when the intruder is observed, the agent will move in the direction of this observation. Also, when the possible start locations of the agent and the targets of the intruder are close to each other, as in the last row of Table 4.5, the game value for the agent is higher.

Optimal agent's strategy with moving sensors

As a final example of the infinite time POAIG, we consider a game with moving sensors. We will use two different observation functions for the agent to show the effect on the agent's strategy.

Consider a game on a grid of size 5×8, similar to Figure 4.3, with nodes labeled from 1 in the upper left corner to 40 in the lower right corner. The agent cannot place sensors at nodes, but is only able to observe the nodes that are reachable by the agent in one or two time steps, so the (sensor) observations depend on the agent's location, which is natural for sensors (e.g., helicopters) based on a ship. The state space, action space, transition probabilities and reward functions are constructed similar to those in Section 4.5.2. The target node of the intruder is node 38. The intruder can take one step and the agent is allowed to take one or two steps each time unit.

The intruder always observes the agent, which yields $O^{(2)} = \{1, ..., 40\}$. The agent can observe the intruder if the intruder is in one of the adjacent nodes, which also yields $O^{(1)} = \{\emptyset, 1, ..., 40\}$. However, the agent observes the intruder in the adjacent nodes with probability 0.75, and in the nodes two steps away with probability 0.25, so the observation functions are:

$$Q^{(1)}(o^{(1)}|s) = \begin{cases} 0.75, & \text{if } o^{(1)} = s^{(2)}, s^{(2)} \in D_1(s^{(1)}), \\ 0.25, & \text{if } o^{(1)} = s^{(2)}, s^{(2)} \in D_2(s^{(1)}), \\ 0.25, & \text{if } o^{(1)} = \emptyset, s^{(2)} \in D_1(s^{(1)}), \\ 0.75, & \text{if } o^{(1)} = \emptyset, s^{(2)} \in D_2(s^{(1)}), \\ 0, & \text{otherwise,} \end{cases}$$

$$Q^{(2)}(o^{(2)}|s) = \begin{cases} 1, & \text{if } o^{(2)} = s^{(1)}, \\ 0, & \text{otherwise,} \end{cases}$$

where $D_1(s^{(1)})$ and $D_2(s^{(1)})$ are the nodes with distance 1 and 2 from node $s^{(1)}$, respectively. The intruder starts at node 2, 3 or 4 with equal probability and the agent starts at node 33, close to the target node.

Due to the large number of possible states and actions, it is infeasible to present the strategy in a table. Therefore, we provide a description of the optimal strategies for the agent and the intruder. From Algorithm 4 we find that the agent's optimal strategy is to move a few steps towards the starting positions of the intruder and then search in that area until the intruder is observed. If the intruder is observed, the agent moves in the direction of this observation. The intruder's strategy is to move in the direction of the target and wait if the agent is close by. If the intruder manages to pass the agent, we find that the intruder reaches the target node with high probability. We generated 1000 sample paths using discrete event simulation, following the steps of Algorithm 4. The intruder is caught by the agent with probability 0.65.

In a second experiment, we reduced the observations of the agent by only allowing observations at nodes with distance 1, but now the intruder is observed with probability 1. In this case, the agent's strategy is similar to that in the first experiment, but now the agent's movements will be more wide spread because he can only observe the intruder in the nodes next to him. We generated 1000 sample paths using discrete event simulation, following the steps of Algorithm 4. The intruder is caught by the agent with probability 0.78. Thus, the probability of catching the intruder is higher when the sensor has a smaller reach, but with a higher detection probability. This can be explained since not only the intruder is always observed with probability 1, but also that not observing anything indicates that the intruder cannot be in any of the nodes with distance 1, which gives additional information about the possible locations of the intruder.

4.6 Concluding remarks

In this paper, we have introduced a partially observable agent-intruder game (POAIG) to model problems related to security of borders or areas. The POAIG is closely related to a traditional stochastic game, but, in our case, the players only have partial information about the game state. These types of games can be applied to model border patrolling or search games. The proposed model can be used to find an optimal strategy for the agent as well as to find the optimal sensor locations for a given number of sensors. To illustrate the presented approaches, three different border security and search games are discussed.

We have applied a solution method, which combines a sequence form representation and a column generation approach to find Nash equilibria for POAIGs with a finite time horizon. By using the sequence form, the matrix size of the game is reduced and we are able to solve larger instances of the game. We have introduced a column generation approach to solve even larger instances of the POAIG.

For the POAIG with an infinite time horizon, we have shown the existence of the ϵ-optimal strategies. Moreover, we have introduced an approximation algorithm based on belief functions to find strategies for the infinite time horizon POAIG. This

algorithm approximates the payoff of the POAIG by the payoff for the game with complete information.

4.7 Appendix

4.7.1 Proofs

Proof of Lemma 4.1. Let $r^{(1)}, r^{(2)}$ be realization probabilities corresponding to a Nash equilibrium in the sequence form game and $\pi^{(1)}$ and $\pi^{(2)}$ are constructed according to (4.6). The value of the sequence form game is:

$$
\begin{aligned}
&(r^{(1)})^\top M r^{(2)} \\
&= \sum_{\sigma^{(1)}\in\Sigma^{(1)}} \sum_{\sigma^{(2)}\in\Sigma^{(2)}} r^{(1)}(\sigma^{(1)}) r^{(2)}(\sigma^{(2)}) g(\sigma^1,\sigma^2) \\
&= \sum_{t=1}^{T} \sum_{\sigma^{(1)}\in\Sigma_t^{(1)}} \sum_{\sigma^{(2)}\in\Sigma_t^{(2)}} r^{(1)}(\sigma^{(1)}) r^{(2)}(\sigma^{(2)}) \\
&\quad \times R((s_l^{(1)}(\sigma^{(1)}), s_l^{(2)}(\sigma^{(2)})), (a_l^{(1)}(\sigma^{(1)}), a_l^{(2)}(\sigma^{(2)}))) p_{\sigma^{(1)}\sigma^{(2)}} \\
&= \sum_{t=1}^{T} \sum_{\sigma^{(1)}\in\Sigma_{t-1}^{(1)}} \sum_{\sigma^{(2)}\in\Sigma_{t-1}^{(2)}} \sum_{s_t\in S} \sum_{o_t\in O} \sum_{a_t\in A} r^{(1)}(\sigma^{(1)}_{s_t^{(1)} o_t^{(1)} a_t^{(1)}}) r^{(2)}(\sigma^{(2)}_{s_t^{(2)} o_t^{(2)} a_t^{(2)}}) \\
&\quad \times R(s_t,a_t) P(s_t|(s_l^{(1)}(\sigma^{(1)}), s_l^{(2)}(\sigma^{(2)})), (a_l^{(1)}(\sigma^{(1)}), a_l^{(2)}(\sigma^{(2)}))) \\
&\quad \times Q^{(1)}(o_t^{(1)}|s_t^{(1)}) Q^{(2)}(o_t^{(2)}|s_t^{(2)}) p_{\sigma^{(1)}\sigma^{(2)}} \\
&= \sum_{t=1}^{T} \sum_{\sigma^{(1)}\in\Sigma_{t-1}^{(1)}} \sum_{\sigma^{(2)}\in\Sigma_{t-1}^{(2)}} \sum_{s_t\in S} \sum_{o_t\in O} \sum_{a_t\in A} r^{(1)}(\sigma^{(1)}) r^{(2)}(\sigma^{(2)}) R(s_t,a_t) \\
&\quad \times P(s_t|(s_l^{(1)}(\sigma^{(1)}), s_l^{(2)}(\sigma^{(2)})), (a_l^{(1)}(\sigma^{(1)}), a_l^{(2)}(\sigma^{(2)}))) Q^{(1)}(o_t^{(1)}|s_t^{(1)}) Q^{(2)}(o_t^{(2)}|s_t^{(2)}) \\
&\quad \times \pi^{(1)}(a_t^{(1)}|\mu, \sigma^{(1)}_{s_t^{(1)} o_t^{(1)}}) \pi^{(2)}(a_t^{(2)}|\mu, \sigma^{(2)}_{s_t^{(2)} o_t^{(2)}}) p_{\sigma^{(1)}\sigma^{(2)}} \\
&\quad + \sum_{t=1}^{T} \sum_{s_1..s_{t-1}\in S^{t-1}} \sum_{o_1..o_{t-1}\in O^{t-1}} \sum_{a_1..a_{t-1}\in A^{t-1}} \sum_{s_t\in S} \sum_{o_t\in O} \sum_{a_t\in A} R(s_t,a_t) \\
&\quad \times \pi^{(1)}(a_t^{(1)}|\mu, \{s_1^{(1)}, o_1^{(1)}, a_1^{(1)}.., a_{t-1}^{(1)}, s_t^{(1)}, o_t^{(1)}\}) \\
&\quad \times \pi^{(2)}(a_t^{(2)}|\mu, \{s_1^{(2)}, o_1^{(2)}, a_1^{(2)}.., a_{t-1}^{(2)}, s_t^{(2)}, o_t^{(2)}\}) \mu(s_1) \prod_{t'=2}^{t} P(s_{t'}|s_{t'-1}, a_{t'-1}) \times \\
&\quad \prod_{t'=1}^{t-1} Q^{(1)}(o_{t'}^{(1)}|s_{t'}^{(1)}) Q^{(2)}(o_{t'}^{(2)}|s_{t'}^{(2)}) \prod_{t'=1}^{t-1} \pi^{(1)}(a_{t'}^{(1)}|\mu, \{s_1^{(1)}, o_1^{(1)}, a_1^{(1)}.., a_{t'-1}^{(1)}, s_{t'}^{(1)}, o_{t'}^{(1)}\}) \\
&\quad \times \pi^{(2)}(a_{t'}^{(2)}|\mu, \{s_1^{(2)}, o_1^{(2)}, a_1^{(2)}.., a_{t'-1}^{(2)}, s_{t'}^{(2)}, o_{t'}^{(2)}\}) \\
&= \sum_{t=1}^{T} \sum_{F_t=\{s_1,o_1,a_1..,a_{t-1},s_t,o_t\}\in\mathcal{F}_t} \sum_{a_t\in A} R(s_t,a_t) \pi^{(1)}(a_t^{(1)}|\mu, F_t^{(1)}) \pi^{(2)}(a_t^{(2)}|\mu, F_t^{(2)}) \\
&\quad \times \mu(s_1) \prod_{t'=2}^{t} P(s_{t'}|s_{t'-1}, a_{t'-1}) \prod_{t'=1}^{t-1} Q^{(1)}(o_{t'}^{(1)}|s_{t'}^{(1)}) Q^{(2)}(o_{t'}^{(2)}|s_{t'}^{(2)})
\end{aligned}
$$

$$\times \prod_{t'=1}^{t-1} \pi^{(1)}(a_{t'}^{(1)}|\mu, F_{t'}^{(1)})\pi^{(2)}(a_{t'}^{(2)}|\mu, F_{t'}^{(2)}) = \mathbb{E}^{\mu}_{\pi^{(1)},\pi^{(2)}} \sum_{t=1}^{T} R(s_t, a_t).$$

The first equality follows from the definition of M and the second equality follows from (4.7). For the third equality, we separate the last position, observation and action from each sequence such that the length of $\sigma^{(1)}$, $\sigma^{(2)}$ is $t-1$ and we use (4.8). The fourth equality follows because of (4.6). The fifth equality is obtained by repeatedly applying (4.6) and use of (4.8). Finally, the last equalities follow from rearranging terms and by definition of the expectation.

Following the steps above, we can rewrite the value of the sequence form game with strategies $r^{(1)}$ and $r^{(2)}$, to the value of the POAIG $V_{\pi^{(1)},\pi^{(2)}}(\mu)$ as in (4.1) and, due to the correspondence between strategies and sequences, vice versa. Moreover, $\pi^{(1)}$ and $\pi^{(2)}$ can be constructed according to (4.6). It follows that if $r^{(1)}$ is a best response to $r^{(2)}$, then $\pi^{(1)}$ is also a best response to $\pi^{(2)}$ and vice versa. Thus if $r^{(1)}$ and $r^{(2)}$ are the strategies in a Nash equilibrium of the sequence form game, $\pi^{(1)}$ and $\pi^{(2)}$ are strategies in a Nash equilibrium in the original POAIG and their values coincide. ■

Proof of Lemma 4.2. Using the definition of the payoff matrix, an upper bound on the expected payoff at time t can be obtained using the following steps:

$$\begin{aligned}
&(r_t^{(1)})^\top M_t r_t^{(2)} \\
&= \sum_{\sigma^{(1)}\in\Sigma_{t-1}^{(1)}} \sum_{\sigma^{(2)}\in\Sigma_{t-1}^{(2)}} \sum_{s_t\in S} \sum_{a_t\in A} \sum_{o_t\in O} r^{(1)}(\sigma^{(1)}_{s_t^{(1)}o_t^{(1)}}a_t^{(1)})r^{(2)}(\sigma^{(2)}_{s_t^{(2)}o_t^{(2)}}a_t^{(2)}) \\
&\quad \times R(s_t,a_t)P(s_t|(s_l^{(1)}(\sigma^{(1)}),s_l^{(2)}(\sigma^{(2)})),(a_l^{(1)}(\sigma^{(1)}),a_l^{(2)}(\sigma^{(2)}))) \\
&\quad \times Q^{(1)}(o_t^{(1)}|s_t^{(1)})Q^{(2)}(o_t^{(2)}|s_t^{(2)})p_{\sigma^{(1)}\sigma^{(2)}} \\
&= \sum_{\sigma^{(1)}\in\Sigma_{t-1}^{(1)}} \sum_{\sigma^{(2)}\in\Sigma_{t-1}^{(2)}} r^{(1)}(\sigma^{(1)})r^{(2)}(\sigma^{(2)})p_{\sigma^{(1)}\sigma^{(2)}} \sum_{s_t\in S}\sum_{a_t\in A} R(s_t,a_t) \\
&\quad \times P(s_t|(s_l^{(1)}(\sigma^{(1)}),s_l^{(2)}(\sigma^{(2)})),(a_l^{(1)}(\sigma^{(1)}),a_l^{(2)}(\sigma^{(2)}))) \times \\
&\quad \pi^{(1)}(a_t^{(1)}|\mu,\sigma^{(1)}_{s_t^{(1)}o_t^{(1)}})\pi^{(2)}(a_t^{(2)}|\mu,\sigma^{(2)}_{s_t^{(2)}o_t^{(2)}}) \sum_{o_t\in O}\left(Q^{(1)}(o_t^{(1)}|s_t^{(1)})Q^{(2)}(o_t^{(2)}|s_t^{(2)})\right) \\
&= \sum_{\sigma^{(1)}\in\tilde{\Sigma}_{t-1}^{(1)}} \sum_{\sigma^{(2)}\in\tilde{\Sigma}_{t-1}^{(2)}} r^{(1)}(\sigma^{(1)})r^{(2)}(\sigma^{(2)})p_{\sigma^{(1)}\sigma^{(2)}} \sum_{s_t\in S}\sum_{a_t\in A} R(s_t,a_t) \\
&\quad \times \pi^{(1)}(a_t^{(1)}|\mu,\sigma^{(1)}_{s_t^{(1)}o_t^{(1)}})\pi^{(2)}(a_t^{(2)}|\mu,\sigma^{(2)}_{s_t^{(2)}o_t^{(2)}}) \\
&\quad \times P(s_t|(s_l^{(1)}(\sigma^{(1)}),s_l^{(2)}(\sigma^{(2)})),(a_l^{(1)}(\sigma^{(1)}),a_l^{(2)}(\sigma^{(2)}))) \\
&\leq \sum_{\sigma^{(1)}\in\tilde{\Sigma}_{t-1}^{(1)}} \sum_{\sigma^{(2)}\in\tilde{\Sigma}_{t-1}^{(2)}} r^{(1)}(\sigma^{(1)})r^{(2)}(\sigma^{(2)})p_{\sigma^{(1)}\sigma^{(2)}} \sum_{s_t\in S}\sum_{a_t\in A} \max_{a\in A,s\in S} R(s,a) \\
&\leq (1-p_{s_A})^{t-1} \max_{a\in A,s\in S} R(s,a),
\end{aligned}$$

where $\tilde{\Sigma}_t^{(i)}$, $i=1,2$, is the set of all sequences that do not contain s_A. The first equality follows from (4.7) according to the same reasoning as in Lemma 4.1. The second equality is obtained by rearranging terms. The third equality is true because the probability that something is observed is always 1. Moreover, $\Sigma_{t-1}^{(i)}$, $i=1,2$

can be replaced by $\tilde{\Sigma}^{(i)}_{t-1}$ because for all sequences that contain s_A, either $p_{\sigma^{(1)}\sigma^{(2)}}$ or $P(s|(s_l^{(1)}(\sigma^{(1)}), s_l^{(2)}(\sigma^{(2)})), a)$ equals zero. The first inequality is true because the reward is always smaller than the maximal reward and all the other terms are less than or equal to 1. The last inequality is obtained as follows:

$$
\begin{aligned}
&\sum_{\sigma^{(1)}\in\tilde{\Sigma}^{(1)}_{t-1}} \sum_{\sigma^{(2)}\in\tilde{\Sigma}^{(2)}_{t-1}} r^{(1)}(\sigma^{(1)})r^{(2)}(\sigma^{(2)})p_{\sigma^{(1)}\sigma^{(2)}} \\
&= \sum_{s_1..s_{t-1}\in\tilde{S}^{t-1}} \sum_{o_1..o_{t-1}\in O^{t-1}} \sum_{a_1..a_{t-1}\in A^{t-1}} r^{(1)}(\sigma^{(1)})r^{(2)}(\sigma^{(2)})p_{\sigma^{(1)}\sigma^{(2)}} \\
&= \sum_{s_1..s_{t-2}\in\tilde{S}^{t-2}} \sum_{o_1..o_{t-2}\in O^{t-2}} \sum_{a_1..a_{t-2}\in A^{t-2}} \sum_{s_{t-1}\in\tilde{S}} \sum_{a_{t-1}\in A} \sum_{o_{t-1}\in O} r^{(1)}(\sigma^{(1)}_{s^{(1)}_{t-1}o^{(1)}_{t-1}} a^{(1)}_{t-1}) \\
&\quad \times r^{(2)}(\sigma^{(2)}_{s^{(2)}_{t-1}o^{(2)}_{t-1}} a^{(2)}_{t-1}) p_{\sigma^{(1)}\sigma^{(2)}} P(s_{t-1}|(s_l^{(1)}(\sigma^{(1)}), s_l^{(2)}(\sigma^{(2)})), (a_l^{(1)}(\sigma^{(1)}), a_l^{(2)}(\sigma^{(2)}))) \\
&\quad \times Q^{(1)}(o^{(1)}_{t-1}|s^{(1)}_{t-1}) Q^{(2)}(o^{(2)}_{t-1}|s^{(2)}_{t-1}) \\
&= \sum_{s_1..s_{t-2}\in\tilde{S}^{t-2}} \sum_{o_1..o_{t-2}\in O^{t-2}} \sum_{a_1..a_{t-2}\in A^{t-2}} r^{(1)}(\sigma^{(1)})r^{(2)}(\sigma^{(2)})p_{\sigma^{(1)}\sigma^{(2)}} \\
&\quad \times \sum_{s_{t-1}\in\tilde{S}} \sum_{a_{t-1}\in A} \sum_{o_{t-1}\in O} \pi^{(1)}(a^{(1)}_{t-1}|\mu, F^{(1)})\pi^{(2)}(a^{(2)}_{t-1}|\mu, F^{(2)}) \times \\
&\quad P(s_{t-1}|(s_l^{(1)}(\sigma^{(1)}), s_l^{(2)}(\sigma^{(2)})), (a_l^{(1)}(\sigma^{(1)}), a_l^{(2)}(\sigma^{(2)})))Q^{(1)}(o^{(1)}_{t-1}|s^{(1)}_{t-1})Q^{(2)}(o^{(2)}_{t-1}|s^{(2)}_{t-1}) \\
&\leq \sum_{s_1..s_{t-2}\in\tilde{S}^{t-2}} \sum_{o_1..o_{t-2}\in O^{t-2}} \sum_{a_1..a_{t-2}\in A^{t-2}} r^{(1)}(\sigma^{(1)})r^{(2)}(\sigma^{(2)})p_{\sigma^{(1)}\sigma^{(2)}} \\
&\quad \times \sum_{s_{t-1}\in\tilde{S}} P(s_{t-1}|(s_l^{(1)}(\sigma^{(1)}), s_l^{(2)}(\sigma^{(2)})), (a_l^{(1)}(\sigma^{(1)}), a_l^{(2)}(\sigma^{(2)}))) \\
&\leq (1-p_{s_A}) \sum_{s_1..s_{t-2}\in\tilde{S}^{t-2}} \sum_{o_1..o_{t-2}\in O^{t-2}} \sum_{a_1..a_{t-2}\in A^{t-2}} r^{(1)}(\sigma^{(1)})r^{(2)}(\sigma^{(2)})p_{\sigma^{(1)}\sigma^{(2)}} \\
&\leq (1-p_{s_A})^{t-1},
\end{aligned}
$$

where $\tilde{S} = S\backslash s_A$. The first three equalities follow from rearranging terms and using (4.8) and (4.6). The first inequality is true because the probability that something is observed is always 1 for each state and all other term are less than or equal to 1. The second inequality is true because the probability of a transition to a state that is not s_A is less than or equal to $1 - p_s$. The last inequality can be obtained by repeating the above-mentioned arguments $t - 2$ times. ■

Proof of Theorem 4.1. First consider a finite POAIG with payoff matrix M and with time horizon $T < \infty$. Let x^* and y^* be the strategies in a Nash equilibrium for the agent and intruder. The value of the game is given by $v^* = (x^*)^\top M y^*$.

Now consider the infinite POAIG with payoff matrix M_∞. Let $x^*_\infty = [x^*\ \bar{x}_\infty]$ and $y^*_\infty = [y^*\ \bar{y}_\infty]$ and $\bar{x}_\infty$ and $\bar{y}_\infty$ can be any vector such that they comply with (4.4) and (4.5). M_∞ can be written as:

$$
M_\infty = \begin{pmatrix} M & 0 & \cdots \\ 0 & M_{T+1} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} = \begin{pmatrix} M & 0 \\ 0 & \bar{M}_\infty \end{pmatrix}.
$$

The payoff for the infinite POAIG with strategies x^*_∞ and y^*_∞ is:

$$v^*_\infty = (x^*_\infty)^\top M_\infty y^*_\infty = (x^*)^\top M y^* + (\bar{x}_\infty)^\top \bar{M}_\infty \bar{y}_\infty.$$

Using Lemma 2, the following holds:

$$\begin{aligned} v^*_\infty &= v^* + (\bar{x}_\infty)^\top \bar{M}_\infty \bar{y}_\infty \\ &\leq v^* + \sum_{t=T+1}^{\infty} (1 - p_{s_A})^t \max_{a \in A, s \in S} R(s,a) \\ &\leq v^* + \max_{a \in A, s \in S} R(s,a) \cdot \frac{(1-p_s)^T}{p_s}. \end{aligned}$$

Since $\max_{a \in A, s \in S} |R(s,a)| < \infty$ and $p_s > 0$, T can be chosen such that $\max_{a \in A, s \in S} R(s,a) \cdot \frac{(1-p_s)^T}{p_s} < \epsilon$ for each $\epsilon > 0$. Since all elements of $\bar{M}_\infty$, $\bar{x}_\infty$ and $\bar{y}_\infty$ are non-negative, we also know that $v^*_\infty \geq v^*$.

We claim that x^*_∞ and y^*_∞ are ϵ-optimal strategies for the POAIG with payoff matrix M_∞, where $x^*_\infty = [x^* \ \bar{x}_\infty]$ and $y^*_\infty = [y^* \ \bar{y}_\infty]$, and x^* and y^* are the strategies in the Nash equilibrium for the finite matrix M with T such that for any $\bar{x}_\infty$ and $\bar{y}_\infty$:

$$(\bar{x}_\infty)^\top \bar{M}_\infty \bar{y}_\infty \leq \epsilon.$$

This means that for all x_∞ and y_∞:

$$\begin{aligned} (x_\infty)^\top M_\infty y^*_\infty &\leq (x^*_\infty)^\top M_\infty y^*_\infty + \epsilon, \\ (x^*_\infty)^\top M_\infty y_\infty &\geq (x^*_\infty)^\top M_\infty y^*_\infty - \epsilon. \end{aligned}$$

We prove this claim by contradiction. Suppose that x^*_∞ and y^*_∞ are not ϵ-optimal strategies and there exists a strategy x_∞ such that $(x_\infty)^\top M_\infty y^*_\infty > (x^*_\infty)^\top M_\infty y^*_\infty + \epsilon$. Let $x_\infty = [x \ \bar{x}_\infty]$, such that x has the same size as the number of rows in M:

$$\begin{aligned} (x_\infty)^\top M_\infty y^*_\infty &= (x)^\top M y^* + (\bar{x}_\infty)^\top \bar{M}_\infty \bar{y}_\infty \\ &> (x^*_\infty)^\top M_\infty y^*_\infty + \epsilon \\ &\geq v^* + \epsilon. \end{aligned}$$

Since $(\bar{x}_\infty)^\top \bar{M}_\infty \bar{y}_\infty \leq \epsilon$, it follows that $(x)^\top M y^* > v^*$. However, this is in contradiction with the fact that x^* is a Nash Equilibrium strategy for the finite game with payoff matrix M. Therefore, our assumption is incorrect and $(x_\infty)^\top M_\infty y^*_\infty \leq (x^*_\infty)^\top M_\infty y^*_\infty + \epsilon$.

A similar argument holds when assuming that y^*_∞ is not ϵ optimal. Combining these two results, gives that x^*_∞ and y^*_∞ are indeed ϵ-optimal strategies. Moreover, the value, v^*_∞, of the infinite game is bounded by $v^* \leq v^*_\infty \leq v^* + \epsilon$. ■

4.7.2 Derivation (counter) belief

The belief function is constructed as follows:

$$\begin{aligned} &\mu^{(1)}_t(s_t | o^{(1)}_k, k \leq t, a_k, k \leq t-1) = P(s_t | o^{(1)}_k, k \leq t, a_k, k \leq t-1) \\ &= \frac{P(o^{(1)}_t | s_t) P(s_t | o^{(1)}_k, a_k, k \leq t-1)}{\sum_{s'_t} P(o^{(1)}_t | s'_t) P(s'_t | o^{(1)}_k, a_k, k \leq t-1)} \end{aligned}$$

$$= \frac{P(o_t^{(1)}|s_t)\sum_{s_{t-1}} P(s_t|s_{t-1},a_{t-1})P(s_{t-1}|o_k^{(1)},k\leq t-1,a_k,k\leq t-2)}{\sum_{s'_t} P(o_t^{(1)}|s'_t)\sum_{s_{t-1}} P(s'_t|s_{t-1},a_{t-1})P(s_{t-1}|o_k^{(1)},k\leq t-1,a_k,k\leq t-2)}$$
$$= \frac{P(o_t^{(1)}|s_t)\sum_{s_{t-1}} P(s_t|s_{t-1},a_{t-1})\mu_{t-1}^{(1)}(s_{t-1}|o_k^{(1)},k\leq t-1,a_k,k\leq t-2)}{\sum_{s'_t} P(o_t^{(1)}|s'_t)\sum_{s_{t-1}} P(s'_t|s_{t-1},a_{t-1})\mu_{t-1}^{(1)}(s_{t-1}|o_k^{(1)},k\leq t-1,a_k,k\leq t-2)},$$

The first equality gives the definition of the belief. In the second equality, we use Bayes' rule. In the third equality, we condition on the previous state and observing that a_{t-1} occurs after observing state s_{t-1} and can therefore be removed from the conditioning argument in the last conditional probability. Finally, the last equality follows from the definition of the belief function.

The derivation of the counter belief is given by:

$$\tilde{\mu}_t^{(1)}(\hat{s}_t|o_k^{(1)},k\leq t,a_k,k\leq t-1) = P(\hat{s}_t|o_k^{(1)},k\leq t,a_k,k\leq t-1)$$
$$= \sum_{o_t^{(2)}}\sum_{s'_t} P(o_t^{(2)}|s'_t)P(s'_t|o_k^{(1)},k\leq t,a_k,k\leq t-1)$$
$$\times P(\hat{s}_t|o_t^{(2)},o_k^{(1)},k\leq t,a_k,k\leq t-1)$$
$$= \sum_{o_t^{(2)}}\sum_{s'_t} P(o_t^{(2)}|s'_t)P(s'_t|o_k^{(1)},k\leq t,a_k,k\leq t-1)$$
$$\times \frac{P(o_t^{(2)}|\hat{s}_t)\sum_{\hat{s}_{t-1}} P(\hat{s}_t|\hat{s}_{t-1},a_{t-1})P(\hat{s}_{t-1}|o_k^{(1)},k\leq t-1,a_k,k\leq t-2)}{\sum_{\bar{s}} P(o_t^{(2)}|s'_t)\sum_{\hat{s}_{t-1}} P(s'_t|\hat{s}_{t-1},a_{t-1})P(\hat{s}_{t-1}|o_k^{(1)},k\leq t-1,a_k,k\leq t-2)}$$
$$= \sum_{o_t^{(2)}}\sum_{s'_t} P(o_t^{(2)}|s'_t)\mu_t^{(1)}(s'_t|o_k^{(1)},k\leq t,a_k,k\leq t-1)$$
$$\times \frac{P(o_t^{(2)}|\hat{s}_t)\sum_{\hat{s}_{t-1}} P(\hat{s}_t|\hat{s}_{t-1},a_{t-1})\tilde{\mu}_{t-1}^{(1)}(\hat{s}_{t-1}|o_k^{(1)},k\leq t-1,a_k,k\leq t-2)}{\sum_{\bar{s}} P(o_t^{(2)}|s'_t)\sum_{\hat{s}_{t-1}} P(s'_t|\hat{s}_{t-1},a_{t-1})\tilde{\mu}_{t-1}^{(1)}(\hat{s}_{t-1}|o_k^{(1)},k\leq t-1,a_k,k\leq t-2)},$$

where $P(o_t^{(2)}|s_t) = Q^{(2)}(o_t^{(1)}|s_t)$ and $P(\hat{s}_t|\hat{s}_{t-1},a_{t-1})$ follow the same distribution as $P(s_t|s_{t-1},a_{t-1})$. The first equality gives the definition of the counter belief. In the second equality, we condition on the last observation of player 2 and on the real state. For the third equality, we use Bayes' rule and condition on the second to last belief state. Finally, we plug in the definition of the counter belief.

CHAPTER 5

Optimal deployment for anti-submarine warfare operations with time dependent strategies

This chapter resulted in [72].

5.1 Introduction

In this chapter, we discuss a model to optimize anti-submarine warfare (ASW) operations. There are different ASW operations, for example area search operations, barrier search operations and transit operations. In area search operations, an area has to be cleared of enemy submarine threats, for example before another operation takes place in that region. The goal of a barrier search operation is to make sure that an enemy submarine is not crossing a certain barrier. Finally, a transit operation is an operation in which one or multiple high value units (HVUs) are performing a transit operation moving from one side of the area to the other side. During this transit operations the HVUs have to be protected from attacks of enemy submarines. In this chapter, we focus on transit operation where one or multiple HVUs have to be protected. However, by adjusting the input parameters, the proposed model can also be used for (barrier) search operation.

The game discussed in this chapter is a special variant of a search game (e.g., [11, 37, 58, 78, 95, 120]). In search games, an intruder is attacking one or multiple cells in a graph while an agent is searching this graph to prevent the attack. An overview of search games is given by Hohzaki in [58]. The main difference with these traditional search games is that we consider an agent that is able to deploy multiple assets of different types (frigates and helicopters), while in the traditional search games only a single asset type is used.

Several ASW models are discussed in literature (e.g., [19, 56, 91, 92, 109, 119]). In [91], the authors give an overview of research related to ASW problems and they describe several issues, such as dimensionality of the resulting model and coordination of multiple assets, that are encountered in these kind of problems. In [119], the author described different models for the planning of multiple ASW platforms for the protection of a HVU. The coordination of multiple platforms is considered using a game theoretic approach to take into account the intruder's behavior. In [56], the authors

consider the patrolling at choke points. By patrolling at choke points, the enemy submarine is deterred or has to take a longer route. They develop an interdiction game to determine the optimal randomized allocation of resources to different choke points.

In this chapter, we consider two approaches for ASW operations, a simulation framework from TNO and a game theoretical model introduced in [19], and extend several aspects of these approaches to overcome their limitations.

The underwater warfare testbed (UWT) is a detailed simulation framework for underwater warfare, developed by TNO [64]. It can be used to develop and evaluate operational tactics and future concepts for underwater operations. The UWT accepts many existing or parametric platforms and underwater systems which can be used to test several tactics and concepts. However, the modeling of the enemy submarine (intruder) is limited by two strategies: a kamikaze-like approach in which the submarine chooses the shortest path towards the HVU and a cautious approach in which the submarine tries to avoid detection. In order to better model the intruder as a intelligent intruder taking into account the strategies of his opponent, we use game theoretic modeling.

In [19, 92] the authors describe a game theoretical ASW model for the protection of a single HVU. The intruder chooses a path from outside the area to the HVU. The agents can deploy multiple assets: frigates, helicopters or submarines. The intruder can observe the frigates, so it is assumed that the location of the frigates are fixed and common knowledge. The allocation of the helicopters and submarine follows a probability distribution. The work of [19, 92] only models a fixed HVU and a static strategy of the agent. However, usually the HVU is moving from one point to another and also the frigate should be able to patrol at several points over time. Therefore, we extend the static model by adding a time aspect. The HVU is moving over time and the strategies of both the agent and the intruder are time dependent. This allows us to model more realistic instances and better react to a moving intruder.

In [19, 92], it is assumed that the frigate's location is known and can be observed by the intruder, while the current position of the helicopters and submarines is not. Therefore, for the static model, the frigate can only be assigned to a position with probability zero or one. However, the position of the other assets is unobservable to the intruder and is allowed to follow a probability distribution. When modeling this problem as a dynamic model, two different strategies are used. First, similar to [19, 92], it is assumed that the frigate's location is known for the complete time window. Second, we develop a model where the frigate's current location becomes known to the intruder at the start of each time interval. Therefore, the frigate's location is also allowed to follow a probability distribution in this case. For both models, the agent always aims at maximizing the probability that the intruder is detected.

The rest of this chapter is organized as follows. In Section 5.2, we consider the game with complete information about the frigate's position. First, we shortly describe the static game model and thereafter, we develop the model used for dynamic allocation of assets where the strategies are time dependent. In Section 5.3, we consider a sequential game approach in which the location of the frigate is known to the intruder at the beginning of each time interval. In Section 5.4, we give computational results and show results for different instances and in Section 5.5, we give some concluding remarks.

5.2 Complete information of frigate's location

In this section, we describe the game in which the frigate's location is known for each time step. We first describe the static model as introduced by [19] in Section 5.2.1 and describe the extended model with time dependent strategies in Section 5.2.2.

5.2.1 Previous work: protection of a static HVU

The static game is modeled as a zero-sum game where the agent is maximizing the detection probability for each route that the intruder, representing the enemy submarine, can choose. For the agent, different assets (frigates, helicopters and submarines) are modeled and the detection rates are specified for each asset. We only consider frigates and helicopters for the agent; submarines are modeled similar to the helicopters.

The game is played over a network with cells C. The set of possible start cells of the intruder is C_S and the target cell is C_T. The strategy set of the agent is the set of all possible allocations for the frigates and helicopters. For the frigates of the agent, there is a decision variable x^F that specifies the probability that the frigate is located in each cell. Because the exact location of the frigates is assumed to be known to the intruders, these probabilities are only allowed to equal 0 or 1. There are N^F frigates and the variable x^F_{mi} is 1 if frigate m, $m = 1, .., N^F$ is located at cell i, $i \in C$, and 0 otherwise. Additionally, for each frigate m, there are N^H_m helicopters. The exact location of the helicopter is not known in advance for the intruder. The allocation of the helicopters is given by a variable x^H_{mnij} that specifies for each cell i the probability that helicopter n, corresponding to frigate m allocated in cell i is allocated to cell j.

By choosing the allocation of frigates and helicopters, the detection rate at each cell d_i, $i \in C$, can be decided. Let D^F_{ij} be the detection rate in cell i of a frigate that is located in cell j and D^H_{ijk} the detection rate in cell i by a helicopter corresponding to a frigate in j that is located in cell k. The intruder chooses a route over a set of routes with minimal detection rate. Let v_i be the expected detection rate from a start point to cell i for the intruder. B^I gives the possible moves of the intruder where B^I_{ij} equals 1 if cell j is adjacent to cell i, 0 otherwise. The time to move from cell i to cell j for the intruder is τ_{ij}.

The agent's strategy can be found by solving (see [19]):

$$
\begin{aligned}
\max_{x^F, x^H} \quad & v_{C_T} \\
\text{s.t.} \quad & \sum_i x^F_{mi} = 1, && m = 1, ..., N^F, \\
& \sum_j x^H_{mnij} = x^F_{mi}, && m = 1, ..., N^F, n = 1, ..., N^H, \\
& d_i = \sum_{j,m} D^F_{ij} x^F_{mj} + \sum_{i,j,n,m} D^H_{ijk} x^H_{mnjk}, && i \in C, \\
& v_j \leq v_i + \tau_{ij} \frac{(d_i + d_j)}{2} + (1 - B^I_{ij})M, && j \in C \backslash C_S, i \in C, \\
& v_i = 0, && i \in C_S, \\
& x^F_{mi} \in \{0, 1\}, && m = 1, ..., N^F, i \in C, \\
& x^H_{ni} \geq 0, && n = 1, ..., N^H, i \in C.
\end{aligned}
$$

The first constraint ensures that each frigate is only assigned to one location. The second constraint makes sure that all helicopters are assigned to a cell such that helicopters corresponding to a frigate in a specific cell can only be assigned if the frigate is in that cell. The third constraint calculates the total detection probability for each cell. The fourth constraint is used to calculate for each cell the probability of detecting the intruder when that intruder is choosing a route to that cell. Here, it is assumed that the intruders will always choose the route to that cell minimizing this detection probability. M is a large number used to ensure that only possible routes are chosen by the intruder. The fifth constraint says that this detection probability is zero for the starting cells of the intruder.

The agent wants to maximize the probability of detecting the intruder. The agent is only interested in maximizing the detection probabilities over the routes that end at the target cell C_T.

5.2.2 Dynamic game with time dependent stategies

In this section, we extend the static model of Brown et al. [19] as described in Section 5.2.1 by modeling a moving HVU and time dependent strategies for both the agent and the intruder. The HVU is moving during the game and the position of the HVU is known in advance to both the agent and the intruder.

For the agent's strategy, we only consider frigates and helicopters. Other assets such as submarines, can be added in a similar way. At each time step, the agent can decide on a new position for the frigates and the helicopter similar to the static game. The location of the frigates is limited to the speed of the frigates. As in the static game, the location of each frigate is known to the intruder, while the exact locations of the helicopters are not. The intruder is representing the enemy submarine that is aiming at attacking the HVU. The intruder's strategy also changes compared to the static game. The intruder is still allowed to choose any path starting in one of the staring locations. Additionally, it might be optimal for the intruder to wait at a cell for a fixed amount of time.

Model description

The game takes place over a network of $N + 1$ cells. The set of possible cells is given by $C = \{0, 1, ..., N\}$. The possible start cells for the intruder are C_S, the frigate can start at every cell. During the game, both the frigates of the agent and the intruder follow a route. The time it takes for a frigate to move from cell i to cell j is τ_{ij}^F and similar for the intruder τ_{ij}^I. For modeling convenience, we assume that the time is discrete and that the game takes place during a fixed time window T. By choosing the length of each interval small enough, the real time can be approximated.

The HVU follows a fixed path that is known to both the agent and the intruder. For each time $t = 0, 1, ..., T$, the target cells (corresponding to the HVU's location) are given by $C_T(t)$.

The actions of the agents consist of two elements: the allocation of the frigates, which is known to the intruder, and the allocation of the helicopters, which is randomized. There are N^F frigates and N_m^H helicopters for each frigate m. For each frigate, the agents decided on a fixed route. The time it takes for a frigate to move from i to j is given by τ_{ij}^F. The frigates are only allowed to move between connected cells given by the matrix B^F, where B_{ij}^F is 1 if i and j are connected and 0 otherwise.

The location of the frigates during each time window is given by x^F_{mit}, so that x^F_{mit} equals 1 if frigate m is in cell i during time t and 0 otherwise. The frigate's location has to be chosen such that it complies with τ^F_{ij} and B^F, which will be taken into account by the construction of the mathematical program.

Each frigates has a number of helicopters that can be deployed. An action of a single helicopter is a probability distribution over the cells depending on the frigates location. Let x^H_{mnijt} be the probability that a helicopter n corresponding to frigate m located in cell i is deployed at cell j during time window t. The strategy space of a single helicopter n corresponding to frigate m is given by:

$$\{x^H | x^H_{mnijt} \geq 0, \sum_j x^H_{mnijt} = x^F_{mit}, j \in C, t = 0, ..., T\}. \tag{5.1}$$

The action of the intruder is also given by a route through the network. Similar to the frigate, the time to move between cells is given by τ^I_{ij}, and possible moves are displayed in the matrix B^I, where B^I_{ijt} equals 1 if the intruder is allowed to move from cell i to cell j during t and 0 otherwise. The routes of the intruder are given by x^I such that x^I_{ijt} equals 1 if the intruder moves from i to j during t. Similar to the frigate, the intruder's location has to comply with τ^I_{ij} and B^I.

The detection rate of the frigates and helicopters is given by D^F and D^H respectively, where D^F_{ij} is the detection rate in cell i if the frigate is in cell j and D^H_{ijk} is the detection rate in cell i for a helicopter that is located in cell j corresponding to a frigate in cell k. The total detection rate d_{it} can be different for each time period depending on the allocation of the frigates and helicopters

$$d_{it} = \sum_{j,m} D^F_{ij} x^F_{mjt} + \sum_{j,m,n,k} D^H_{ijk} x^H_{mnjkt}, \quad i \in C,\ t = 0, ..., T, \tag{5.2}$$

Cell 0 is a fictive cell where both frigates and intruders start. Also the intruder moves to this cell as soon as one of the targets is reached. The detection rates D^F and D^H at this cell equal 0. The frigate can start at every possible cell, such that cell 0 is adjacent to all $i \in C$ and B^F_{0j} equals 1 for each $j \in C$. The intruder can only start from the start cell, so B^I_{0jt} equals 1 for all $j \in C_s$, $t = 0$. Moreover, the intruder can move to cell 0 after reaching a target cell, so $B^I_{i0t} = 1$ if $i \in C_T(t)$.

Given the detection probabilities d_{it} and an intruder's strategy x^I, the game value, which is the total detection rate, is:

$$f(d, x^I) = \begin{cases} \sum_{t=1}^{T} \sum_{i \in C} d_{it} x^I_{it}, & \text{if } \exists i \in C_T(t) \text{ s.t. } x^I_{it} = 1, \\ \infty, & \text{otherwise.} \end{cases}$$

The agent is maximizing this function by choosing the routes for the frigates and allocations for the helicopters, while the intruder is minimizing this by deciding on a route.

Construction of the integer linear program

We write $\max_d \min_{r \in R^I} f(d, r^I)$ as a mathematical program by constructing the number of cells and matrices B^F and B^I of the agent and intruder such that the travel time to each possible cell takes exactly 1 time step. Note that by adjusting the number of cells, different travel times can be modeled. We show this with the following example.

Example 5.1. Consider the original ASW game with a single frigate. The frigate moves twice as fast as the intruder, so $\tau^F_{ij} = 1$ and $\tau^I_{ij} = 2$ for each $i, j \in C$. B^F and B^I give the possible movements of the frigate and intruder respectively. Now we construct a new game such that $\tilde{\tau}^F_{ij} = \tilde{\tau}^I_{ij} = 1$. To ensure that this game is equivalent with the original game, we construct $\tilde{C}$, $\tilde{B}^F$ and $\tilde{B}^I$. in the following way. The number of cells is twice as large as the number of cells in the original game, so $\tilde{C} = \{0, 1, 2, ..., 2N\}$, where each second element corresponds with an element from C, so $i \in \tilde{C}$ corresponds with $i/2$ in C. $\tilde{B}^F_{ij}$ equals 1 if i and j are even and $B^F_{(i/2)(j/2)} = 1$, and 0 otherwise. Finally, $\tilde{B}^I_{ij}$ equals 1 if i is odd and $j = i+1$ or when i is even and $B^I_{(i/2)((j+1)/2)} = 1$. By constructing $\tilde{B}^F$ and $\tilde{B}^I$ in this way, the frigates only move between the even cells and the intruder always has to travel through one additional (odd) cell. Therefore, the intruder needs two times steps to move between cells corresponding to the original game. □

Consider the agent's allocation of the frigates and helicopters to determine the detection rates d_{it}. The strategies of the agent can then be found by solving the following maxmin formulation:

$$
\begin{aligned}
\max_{x^F, x^H} \quad & \min_{x^I} f(d, x^I) \\
\text{s.t.} \quad & \sum_i x^F_{mit} = 1, && m = 1, .., N^F, t = 0, ..., T, \\
& x^F_{mit} \leq (1 - x^F_{mit'}) + B^F_{mij}, && i, j \in C, t' = t - 1, t = 1, ..., T, \\
& \sum_j x^H_{mnijt} = 1, && m = 1, .., N^F, n = 1, .., N^H_m, \\
& && i \in C, t = 0, ..., T, \\
& d_{it} = \sum_{j,m} D^F_{ij} x^F_{mjt} \\
& \qquad + \sum_{j,m,n,k} D^H_{ijk} x^H_{mnjkt}, && i \in C, t = 0, ..., T, \\
& x^F_{mit} \in \{0, 1\}, && m = 1, ...N^F, i \in C, t = 0, ..., T, \\
& x^H_{mnijt} \geq 0, && m = 1, ...N^H, i \in C,\ t = 0, ..., T,
\end{aligned}
$$

where the first three constraints ensure that each frigate follows a feasible route and each helicopter is scheduled with probability one. With the fourth constraint, the total detection rate for each cell and each time step is calculated.

Intruder's program To construct a linear program, we formulate the intruders program $\min_{x^I} f(d, x^I)$ separately. Thereafter, we can show that relaxing the integer constraints gives the same value. By taking the dual of the resulting LP, we can rewrite the maxmin problem as a maximization problem. Given the detection probability d_{it} the intruder's strategy can be determined by:

$$
\min_{x^I_{ijt}} \quad \sum_{ijt} d_{jt} x^I_{ijt} \tag{5.3}
$$

$$\text{s.t.} \sum_j x^I_{jit} = \sum_j x^I_{ijt'}, \quad i \in C, t' = t+1, t = 0, ..., T-1, \tag{5.4}$$

$$\sum_{j \in C_S} x^I_{ijt} = 1, \qquad i = 0, t = 0, \tag{5.5}$$

$$x^I_{ijt} \leq B^I_{ijt}, \qquad i, j \in C, t = 0, ..., T, \tag{5.6}$$

$$x^I_{ijt} \in \{0, 1\}, \qquad i, j \in C, t = 0, ..., T. \tag{5.7}$$

The first constraint ensures that the flow into a cell equals the flow out of a cell. The second constraint makes sure that the intruders start at one of the start cells. The third constraint ensures that only allowed routes are chosen by the intruder.

Theorem 5.1. *The matrix corresponding to Constraints* (5.4)-(5.7) *of the intruders problem is totally unimodular.*

Proof. To show that A is totally unimodular, we use the following property. A matrix is totally unimodular if (1) each column contains at most two nonzero elements and (2) there is a subset R of the rows of such that (2a) if a column has two non-zero elements that with the same sign, one of these elements is in K and the other not or (2b) if the two non-zero elements have opposite sign then they either both contained in R or both not contained in R.

For each combination of ijt, $i, j \in C$, $t \geq 0$, there is a column in A. Consider the submatrix $\tilde{A}$ of A which consist of the rows corresponding to Constraints (5.4). We first show that $\tilde{A}$ is totally unimodular. For the first rows representing Constraint (5.4), each combination of ijt, $i, j \in C$ and t appears at most one time on the left hand side and one time on the right hand side. So there are at most two non-zero elements in each column and if there are exactly two, they have opposite signs. Therefore, $\tilde{A}$ is totally unimodular.

Now consider $\hat{A}$ consisting of $\tilde{A}$ and one additional row representing Constraint (5.5). Only columns representing $i = 0$ are contained in these constraints. Note that these columns have at most one non-zero element from Constraints (5.4) with positive sign. Now, at most one non-zero element is added also with positive sign. By choosing the last row as R, the conditions for total unimodularity are still satisfied.

To prove that A is totally unimodular, we use the following property. Total unimodularity is preserved under the following operation: adding a row or column with at most one non-zero entry. This is exactly what is done by adding the Constraints (5.6) to $\hat{A}$. So, it follows that A, representing Constraints (5.4)-(5.6) is totally unimodular which proves our theorem. ■

Complete integer linear program From Theorem 5.1, we know that relaxing the integer constraints of x^I_{ijt} still gives an integer solution. Therefore, we can take the dual of the relaxed intruders problem to reformulate the maxmin formulation. By replacing the intruder's problem of the maxmin formulation with the dual, the maxmin formulation can be rewritten as a single maximization problem. Then, the agent's strategy can be found by solving the following ILP, where Equations 5.8, 5.13-5.16, 5.19 correspond to the dual formulation:

$$\max_{y, x^F, x^H} \quad y^{(2)} + \sum_{i,j,t} y^{(3)}_{ijt} B^I_{ijt} \tag{5.8}$$

$$s.t. \sum_i x^F_{mit} = 1, \qquad m = 1,..,N^F, t = 0,...,T, \tag{5.9}$$

$$x^F_{mjt} \leq (1 - x^F_{mit'}) + B^F_{mij}, i,j \in C, t' = t-1, t = 1,...,T, \tag{5.10}$$

$$\sum_j x^H_{mnijt} = x^F_{mjt}, \qquad m = 1,..,N^F, n = 1,..,N^H_m, \; i \in C, t = 0,...,T, \tag{5.11}$$

$$d_{it} = \sum_{j,m} D^F_{ij} x^F_{mjt} + \sum_{j,m,n,k} D^H_{ijk} x^H_{mnjkt}, \; i \in C, t = 0,...,T, \tag{5.12}$$

$$y^{(1)}_{jt} - y^{(1)}_{it'} + y^{(3)}_{ijt} \leq d_{jt}, \qquad i,j \in C, t = 1,...,T-1, \; t' = t-1, \tag{5.13}$$

$$y^{(1)}_{jt} + y^{(3)}_{ijt} \leq d_{jt}, \qquad i,j \in C, t = 1, \tag{5.14}$$

$$-y^{(1)}_{it'} + y^{(3)}_{ijt} \leq d_{jt}, \qquad i,j \in C, t = T, t' = t-1, \tag{5.15}$$

$$y^{(1)}_{jt} + y^{(2)} + y^{(3)}_{ijt} \leq d_{jt}, \qquad i = 0, j \in C_s, t = 0, \tag{5.16}$$

$$x^F_{mit} \in \{0,1\}, \qquad m = 1,...N^F, i \in C, t = 0,...,T, \tag{5.17}$$

$$x^H_{mnijt} \geq 0, \qquad m = 1,...N^H, i \in C, t = 0,...,T, \tag{5.18}$$

$$y^{(3)}_{ijt} \leq 0, \qquad i \in C, t \geq 0. \tag{5.19}$$

Theorem 5.2. *Solving* (5.8)-(5.19) *to optimality is NP-hard.*

Proof. We give a reduction from the set covering problem. The decision problem of set cover is NP-hard and described by the following. Given a universe $U = \{1,...,n\}$, a set of sets S such that for each $S_i \in S$ holds that $S_i \subset U$ and $\bigcup_{i=1}^{|S|} S_i = U$, and an integer k, does there exist a subset of S with at most k elements such that the universe is covered by the union of this subset? We show that each instance of the set covering problem with universe size n, sets S and integer k can be reduced to an instance of the ILP used to found a strategy for the agent. This proves that finding an optimal strategy for the agent is NP-hard.

The ILP instance is constructed as follows. Let $N^F = 1$ and $N^H = 0$. The game is played on a network of $nk+|S|+1$ cells, $C = \{1,..,nk, nk+1,...,nk+|S|, nk+|S|+1\}$, over a time period k. The HVU is always at the same target cell, so $C_T(t) = nk+|S|$, $t = 0,...,T$. For each element i, $i = 1,...,n$ from the universe there is a path from a start point $(i-1)k+1$ to the target cell $nk+|S|$. For example, for $i = 1$, there is a path $1,2,...,k,nk+|S|$. B^I is constructed according to this, so B^I_{ijt} equals 1 if $j = i+1$ and 0 otherwise. Additionally, there is a cell $nk+i$ corresponding to each element i, $i = 1,...,|S|$ from the set S and all these cells are connected for the frigates, so $B^F_{ij} = 1$ for all $i,j = nk+1,...,nk+|S|$. Finally, construct D^F in the following way: for each $j \in nk+1,...,nk+|S|$ corresponding to a set S_j let D^F_{ij} equal 1 if i is a cell in the path corresponding to an element from the universe that is also contained in S_i, and 0 otherwise.

Solving this instance of the ILP with parameters described above gives a solutions for the decision of the set cover problem. If it results in a solution with value larger than 0, this means that for each path at least one time the detection probability was

larger than 0 and thus there exists a set cover with at most k element. If the optimal solution equals 0 then there is no set cover with k sets. ■

To overcome the complexity of the ILP and to speed up the solving process, we use a branch and price approach. The branch and price algorithm is a combination of column generation and branch and bound, where first the relazation of the ILP is solved using column generation and then branch and bound is applied to create integer solutions (see [10]). To be able to apply column generation, we introduce fixed routes for the frigate. The variables x^F that describe the movement of the frigate in each time interval are replaced by a set of fixed routes that considers the frigate's routes at once. Let S be the set of frigate's routes. A route s is given by the parameters $\tilde{x}^F_{mits}$ similar to the variables x^F_{mit}, such that $\tilde{x}^F_{mits}$ equals 1 if in route s frigate f is in cell i during t and zero otherwise. To construct the ILP with routes, Constraints (5.9), (5.10) and (5.17) are replaced by:

$$x^F_{mit} = \sum_s q_s \tilde{x}^F_{mits}, \quad i,j \in C, t = 0, ..., T,$$

$$\sum_s q_s = 1,$$

$$q_s \in \{0,1\}, \qquad s \in S,$$

If S is the set of all routes, this ILP finds the agent's optimal strategy. However, since the number of routes is exponential in the number of cells and therefore, we approximate the optimal solution using branch and price. Computational results are given in Section 5.4.

5.3 Sequential game approach

In the previous section, we assumed that the frigate's route is completely known to the intruder at the beginning of the game. In this section we consider the case where the frigate's position is only known at the beginning of each time step. This game can be modeled as a sequential game, where the intruder decides on his next move at the beginning of each time interval. Since the agent does not get any information about the intruder's action during the game, he can decide on a complete strategy at the beginning of the game. The intruder's strategy depends on the movement of the frigate. Therefore, the variable x^I depends on the position of the frigate. For this model, we assume that there is only one frigate, but the model can be easily extended in a similar way for multiple frigates.

The agent's strategy consists of the route for the frigates and the allocation of the helicopters. In contrast to the approach discussed in the previous section, the agent is allowed to randomize over the routes of the frigates. Similar as in the column generation method described in the previous section, the agent can choose a route for the frigates out of a the set of all possible routes S, where $\tilde{x}^F_{mits}$ equals 1 if in route s frigate m is in cell i during t and 0 otherwise. The set S and values of $\tilde{x}^F_{mits}$ are given. Additionally, we introduce for each s the parameter o_{kts} which equals 1 if the frigate is observed in k, $k \in C$, by the intruder during t and 0 otherwise. This parameter is used to determine the intruder's strategy for each s.

Since the helicopters allocation and detection probability depends on the frigate's route, the total detection rate d_{its} and the helicopter's allocation x^H_{mnijts} depend on the route s. The probability that route s is chosen by the agent is given by q_s. So, the agent's decision variables are q_s and x^H_{mnijts}, which results in a value for d_{its}. Similarly to the approach used in Section 5.2.2, a linear program can be formulated by first considering the intruders problem separately.

Intruders problem The intruder's strategy x^I depends on the location of the frigate. Let x^I_{ijkt} equals 1 if intruder moves from i to j after observing the frigate at k during time t. The route that is eventually chosen by the intruder depends on the the route s and is determined by o_{kts}. Let $\tilde{x}^I_{ijts}$ be the intruders route if the agent selects route s. The intruder's optimization problem with d_{its} and q_s is:

$$
\begin{aligned}
\min_{\tilde{x}^I_{ijts}, x^I_{ijkt}} \quad & \sum_s q_s \sum_{ijt} d_{jts} \tilde{x}^I_{ijts} \\
\text{s.t. } & o_{kts}\tilde{x}^I_{ijts} \leq x^I_{ijkt}, && i,j,k \in C, t = 0,...,T, s \in S, \\
& \sum_j x^I_{ijkt} = 1, && i,k \in C, t = 0,...,T, \\
& \sum_j \tilde{x}^I_{ijts} = \sum_j \tilde{x}^I_{ijt's}, && i \in C, t = 0,...,T-1, t' = t+1, s \in S, \\
& \sum_{j \in C_S} \tilde{x}^I_{ijts} = 1, && i = 0, t = 0, s \in S, \\
& \tilde{x}^I_{ijts} \leq B^I_{ijt}, && i,j \in C, t = 0,...,T, s \in S, \\
& \tilde{x}^I_{ijts}, x^I_{ijkt} \geq 0, && i,j,k \in C, t = 0,...,T, s \in S.
\end{aligned}
$$

The route that is actually executed is $\tilde{x}^I$ and depends on x^I, which is the intruder's strategy. The first constraint ensures that x^I corresponds with the choice of $\tilde{x}^I$ and the second constraint ensures that for each possible combinations of locations and observations a choice is made. The third, fourth and fifth constraints ensure that the executed routes also satisfy the flow equations and only possible routes are chosen.

Complete linear program By taking the dual of relaxation of the intruder's problem, the complete mathematical program to find the agent's strategy is given by:

$$
\begin{aligned}
\max_{y, x^F, x^H} \quad & \sum_s y^{(2)}_s + \sum_{i,j,t,s} y^{(3)}_{ijts} B^I_{ijt} + \sum_{ikt} y^{(5)}_{ikt} \\
\text{s.t. } & \sum_j x^H_{mnijts} = \tilde{x}^F_{mjts}, && m = 1,..,N^F, n = 1,..,N^H_m, \\
& && i \in C, t = 0,...,T, s \in S, \\
& d_{its} = \sum_{j,m} D^F_{ij} \tilde{x}^F_{mjts} + \sum_{j,m,n,k} D^H_{ijk} x^H_{mnjkts}, && i \in C, t = 0,...,T, \\
& -\sum_s y^{(4)}_{ijkts} + y^{(5)}_{ikt} \leq 0, && i,j,k \in C, t = 0,...,T, \\
& y^{(1)}_{jts} - y^{(1)}_{it's} + y^{(3)}_{ijts} + \sum_k o_{kts} y^{(4)}_{ijkts} \leq q_s d_{jts}, && i,j \in C, t = 1,...,T-1, \\
& && t' = t-1, s \in S,
\end{aligned}
$$

$$
\begin{aligned}
& y^{(1)}_{jts} + y^{(3)}_{ijts} + \sum_k o_{kts} y^{(4)}_{ijkts} \leq q_s d_{jts}, && i,j \in C, t = 0, s \in S, \\
& -y^{(1)}_{it's} + y^{(3)}_{ijts} + \sum_k o_{kts} y^{(4)}_{ijkts} \leq q_s d_{jts}, && i,j \in C, t = T, \\
& && t' = t-1, s \in S, \\
& y^{(1)}_{jts} + y^{(2)} + y^{(3)}_{ijts} + \sum_k o_{kts} y^{(4)}_{ijkts} \leq q_s d_{jts}, && i = 0, j \in C_s, t = 0, s \in S, \\
& \sum_s q_s = 1, && \\
& q_s \geq 0, && s \in S, \\
& x^H_{mnijt} \geq 0, && m = 1,..,N^F, n = 1,..,N^H_m, \\
& && i \in C, t = 0,...,T, s \in S, \\
& y^{(3)}_{ijts}, y^{(4)}_{ijkts}, y^{(5)}_{ikt} \leq 0, && i,j \in \{0, C\}, t \geq 0, s \in S.
\end{aligned}
$$

This mathematical problem can be linearized by introducing the variables $d^q_{jts} = q_s d_{jts}$ and $x^{H,q}_{mnijts} = q_s x^H_{mnijts}$, replacing the first two constraints by:

$$
\begin{aligned}
& \sum_j x^{H,q}_{mnijts} = \tilde{x}^F_{mjts} q_s, \quad m = 1,..,N^F,\ n = 1,..,N^H_m,\ i \in C\ t = 0,...,T, s \in S, \\
& d^q_{its} = \sum_{j,m} D^F_{ij} \tilde{x}^F_{mjts} q_s + \sum_{j,m,n,k} D^H_{ijk} x^{H,q}_{mnjkts}, \quad i \in C,\ t = 0,...,T.
\end{aligned}
$$

In the sequential game approach, the probability that a route is chosen, q_s, is allowed to be non integer. This is in contrast with the approach used in the previous section, because there the complete frigate's path is assumed to be known by the intruder in advance, while this is not the case for the sequential game approach. However, since the number of routes grows exponentially in the number of cells, this linear program is still difficult to solve. Note that we cannot apply column generation to this LP, since the allocation of the helicopters also depends on the chosen route. Therefore, we use the routes generated by using column generation for complete information games as input for this LP. Computational results are given in the next section.

5.4 Results

In this section, we evaluate the models introduced in this chapter and we give results for several instances. First, we consider the model described in Section 5.2.2, where the complete path of the frigates is known to the intruder. For small instances, we give the optimal agent's allocation of the frigate(s) and the helicopter(s) and we describe the intruder's strategy. Since finding the optimal agent's strategies is NP-hard, we are unable to solve the model for larger instances efficiently. Therefore, we use a branch and price algorithm to approximate optimal solutions and evaluate the quality of this algorithm. Second, we give results for the second method described in Section 5.3 using the sequential game approach. Finally, we compare the results obtained by our model with the UWT simulation of TNO.

5.4.1 Complete path frigate known

In this section, we evaluate different instances for the game with complete information of the frigate's location. All figures can be found in Section 5.6.

Small examples for different asset configurations

We test our model on a small instance for different asset configurations. Consider a game on a 5x5 grid and a time horizon $T = 12$. There is an intruder that can start at any bottom cell in the area (see Figures 5.1-5.3), which is also known to the agent. We test the model for three asset configurations: one frigate (Figure 5.1), two frigates (Figure 5.2), and one frigate and one helicopter (Figure 5.3). The frigate is twice as fast as the intruder meaning that the frigate can move one step per time interval and the intruder can only move one step per two time intervals. The frigate has a detection probability of 0.75 if the intruder and frigate are located at the same cell. The helicopter is able to move two steps from the frigate and has a detection probability of 0.5 if the helicopter and intruder are at the same cell. The optimal solutions are displayed in Figures 5.1-5.3 in the following way. Each subfigure gives the allocations of frigates and helicopters during one time interval. The letter F gives the position of the frigate and the gray circles give the helicopter allocation. The larger the circle, the higher the probability that the helicopter is allocated to that cell. Given the frigate's and helicopter's allocation of the agent, an optimal strategy for the intruder can be calculated. This route is given by the letter I. Note that the intruder always stays at least two time steps at the same node, since the intruder is twice as slow as the frigate. In the previous sections, we have used with detection rates in order to make it easier to add the rates over for different cells and time windows. In this section, these rates are translated to equivalent detection probabilities. The detection probabilities and running times can be found in Table 5.1.

Table 5.1: Results for different asset configurations.

Instance	Detection probability	Running time (sec)
1 frigate	0.68	56
2 frigates	0.95	574
1 frigate, 1 helicopter	0.93	33

One can see in Figures 5.1-5.2 that the frigate starts in or nearby one of the possible start locations of the intruder and then moves towards the HVU while scanning a large number of cells. Since the intruder can only start at one of the lower cells, it is not possible for the intruder to already attack the HVU in the beginning. Therefore, the agent can use this time to actively search the intruder and since the frigate is faster than the intruder, the frigate is able to move towards the HVU before it can be reached by the intruder. As to be expected, two frigates will have a higher detection probability and are able to cover both one side of the area as can be seen in Figure 5.2. Also, when an additional helicopter can be deployed (Figure 5.3), the detection probability will increase. However, this increase is slightly smaller than when an additional frigate is used, since the frigate is more flexible and has a higher detection probability.

Realistic sized example

We now investigate a larger instance of the ASW model to illustrate a strategy for a moving target. Consider a grid of 7x10 with a HVU that is moving over a time window of 16 (see Figure 5.4). The detection probability of the frigate is 0.5 if the intruder is in the same cell. The helicopter can be located two steps from the frigate and observes the intruder with probability 0.5. The possible start locations of the intruder are given by an $\times$ in the first subfigure of Figure 5.4. The agent's optimal strategy is displayed in Figure 5.4 and this results in a total detection probability of 0.61. Again, the frigate and the helicopter are first searching the area where the intruder could be and are thereafter moving in the direction of the HVU. In Section 5.4.3, we use this example to compare our method with the UWT simulation.

As the problem is NP-hard by Theorem 5.2, the running time increases exponentially as the number of cells increase. To solve the instance on a 7x10 grid, the running time is more than two weeks on an Intel(R) Core(TM) i5 CPU, 2.2GHz, 8 GB of RAM.. We use a branch and price algorithm to speed up the solving time. With this algorithm, we are able to find an approximate solution with an error of 14.1% within a couple of hours. We have tested the branch and price algorithm for smaller instances and from these tests, it follows that the approximate solutions is always within 15% of the optimal solution and for several instance, the optimal solution is found. However, due to the number of routes and the fact that only one of these routes is used in an optimal solution, it is not possible to always guarantee a high solution quality. For the examples with different asset configurations considered in Section 5.4.1 an approximate solution within respectively 14.2%, 3.2% and 3.1% of the optimal solution is found by the branch and price algorithm.

5.4.2 Sequential game approach

In Section 5.3, we deviated from the assumption that the intruder has complete information about the position of the frigate during the complete time interval. Therefore, the agent is allowed to randomize over the different frigate's routes and will be less predictable, resulting in a higher payoff. However, since the number of routes is exponential in the number of cells, the number of variables is large and the LP to find optimal agent's solutions cannot be solved efficiently. Therefore, we only optimize over a limited set of routes and we use the column generation method explained at the end of Section 5.2.2 to determine these routes. However, even with a limited number of routes, the number of variables is very high because for each route, we need separate variables for the helicopter's allocation, detection probability and the corresponding intruder's routes.

First, we give a small example to show what the impact of this sequential approach can be. Thereafter, we compare the sequential game approach to the instances with complete information.

Example 5.2 (Small sequential game)**.** Consider a game on a 3x1 grid with the target at cell 2. The possible start cells of the intruder are 1 and 3 from where he can immediately move to the target cell. The agent has a single frigate available with a detection probability of 1 if the frigate is at the intruder's location and 0 otherwise. All possible strategies for the agent are starting at cell 1, 2 or 3. When the complete location of the frigate is known in advance for the intruder, the intruder can always

start at a cell where the agent is not present and the total detection probability equals 0. However, if the frigate's location is not known to the intruder in advance, the optimal strategy of the agent is patrolling cell 1 with probability 0.5 and patrolling cell 3 with probability 0.5 resulting in a total detection probability of 0.5.

In the above example, we described an extreme case where the sequential game approach will lead to a significant higher detection probability than the game in which complete information of the frigate's location is considered. We now investigate the impact on a more realistically sized instance.

Consider the instance on a 5x5 grid discussed in Section 5.4.1. We test the same asset configurations, but with the sequential game approach. We use the first 20 routes that are generated using column generation for the model with complete information about the frigate's location.

Table 5.2: Impact sequential approach.

Instance	Game value	Running time (sec)
1 frigate	0.79	1077
2 frigates	0.96	3472
1 frigate, 1 helicopter	0.95	2201

As these results show, the agent's detection probability increases a lot for the first instance, since the agent can also randomize over the frigate's position and is therefore less predictable. However, since the number of variables is very large as the number of cells increase, the running time increases quickly and we are not able to solve this model for larger instances.

5.4.3 Comparision with UWT approach

The instance with one frigate and one helicopter was simulated in the UWT (Underwater Warfare Testbed) as well, to find out whether the optimal path that was found in the game theoretical approach yield a higher performance in the UWT simulation

The UWT is a simulation model developed by TNO [64]. It can be used to develop and evaluate operational tactics and future concepts for underwater operations. In the model, platforms that are part of an underwater operation, such as frigates, helicopters, submarines, mines, torpedoes and unmanned systems are modeled as agents. They are equipped with sensors and possibly separate weapons.

The behavior of the platforms (agent) is modeled in two ways. First, they follow a predefined pattern, e.g., defined by way points, through the modeled operational area. When an agent detects an adversary, it can react by attacking or avoiding this adversary, or ignoring it. Agents of the same side can coordinate their actions. These reactions are scripted (e.g., reduce speed to v when a helicopter sonar is detected within R nautical miles).

For the detection of a contact, several levels of detail exist. The simplest model is using a cookie-cutter sensor with fixed detection ranges. The sensor detects a contact as soon as it is within its sensor range. The most advanced model is calculating the acoustic propagation through the environment and translating this into a probability of detection. Several intermediate levels of detail are also possible. For the comparison

in this chapter, it assumed that the sensor can detect a contact with probability p when it is within the sensor range and not if it is outside the sensor range.

Typically the waypoints that are followed, depend on the tactic that is chosen. Usually a set of tactics is defined in close cooperation with operational experts, taking into account operational requirements and limitations (e.g. the tactic is not too complicated and can be carried out during an operation). The effectiveness of each tactic is determined with the simulation. The main outcome of a simulation run is whether the defense against the incoming submarine was successful or not. Monte Carlo simulation provides an estimation for the probability of success. Other output of the model include ranges at which platforms were detected and by which agents, number of possibilities to launch an attack.

Because of this approach, the UWT does not automatically determine the optimal tactic. The definition of several different tactics is the task of the analyst. After execution the simulations, it can be determined which of these tactics as the highest performance. This might lead to a new set of tactics that are analyzed or an advice about which tactic to prefer. For this advice the outcome of the simulation might not be the only factor.

For the intruder, two types of behavior are available: a kamikaze-like approach in which the submarine chooses the shortest path towards the HVU and a cautious approach in which the submarine tries to avoid detection. It must be noted that both types of behavior do not correspond exactly to the optimal path found in the game theoretic approach. However, due to time constraints, it was decided to use both types of existing behavior in the Monte Carlo simulation.

Detection of contacts in the UWT is modeled by the transmission of sonar pings. Each ping can lead to a detection. The probability of a detection in the UWT simulation depends on the distance between sensor and contact. The grid cells are not modeled explicitly in the UWT, but the speed of the HVU, frigates and intruders are modeled in such a way that they correspond with the game model. Moreover, the detection rates of the frigate and helicopter are modeled such that they correspond with the game theoretic approach. However, since we use a square grid for our model and a detection radius is used in the simulation, they are not exactly the same.

The behavior of the intruder and the way in which the detection process is modeled, will cause a difference in outcome between the game theoretic approach and the simulation with the UWT. Furthermore, tactics are input for the UWT instead of output. Therefore, the optimal search pattern founded by the game theoretic models, is compared to three other tactics (also see Figure 5.5):

1. Frigate in front of the HVU covers the right part of the area in front of the HVU, helicopter covers the left part.

2. Frigate moving from left to right and back in front of the HVU. Helicopter dipping at positions near the frigate.

3. Frigate in front of the HVU covers the right path in front of the HVU, helicopter covers the left part of the area before the HVU.

For the simulation, we used 55 different starting positions of the intruder to count for the uncertainty in where the submarine starts its attack. For each starting point, we simulated 25 runs. The average agent's detection probability for the 2 tactics of the intruder and 4 agent's tactics are given in Table 5.3.

Table 5.3: Impact sequential approach.

	Kamikaze approach	Cautious approach
Optimal agent's game tactic	0.58	0.59
Tactic 1	0.43	0.50
Tactic 2	0.44	0.49
Tactic 3	0.47	0.68

Because of the modeling aspects mentioned above, it is likely that the performance determined by the simulation differs from the performance of the optimal tactic as determined by the sequential game approach. Therefore it is more useful to look at the difference in performance between the selected tactics. It can be observed that the optimal tactic of the game approach performs in most cases better than the alternative tactics 1 and 2. Tactic 3 performs slightly better for the cautious intruder.

An advantage of using the game theory approach is that the analyst does not have to define tactics in advance. In the approach of selecting tactics beforehand that will be compared, it is very likely that the analyst limits himself to tactics that are already well-known or widely accepted. The game theory approach can come up with unconventional alternatives that might be difficult to carry out in practice, but may lead to innovating ideas about new tactics. For example, the optimal tactic from the game theory approach shows that it is advantageous to deploy frigate and helicopter far upfront if it is known that the submarine did not have time to approach the HVU. This enables early detection and warning. Later in the scenario, the submarine could have come close to the HVU and the frigate and helicopter need to be closer to the HVU to cover the area around the HVU to prevent an attack on the HVU.

5.5 Concluding remarks

In this chapter, we have introduced a new model for the protection of large areas against enemy submarines. In this model, we address several limitations of current models that are proposed for anti-submarine warfare (ASW) operations. By introducing a time aspect, we extend the model of [19, 92] such that also time dependent strategies and moving HVUs can be modeled. Moreover, by using a game theoretic approach, our model is able to model an intelligent intruder.

For modeling ASW operations with time dependent strategies, we have proposed two different approaches: one with complete information about the frigates location for the complete time interval and one sequential game approach where the intruder only becomes aware of the frigate's location during each time interval. The sequential game approach gives higher solution quality for the agent, since the agent has the possibility to randomize over frigate routes. However, since for every possible route a helicopters strategy has to be specified, the number of variables is very large and we are unable to solve realistically sized instances. Future research include investigating methods to solve large instances efficiently.

For the approach with complete information about the frigate's location, we are able to solve large instances and compare these with the UWT simulation. However, some aspects of ASW operations are modeled different in our game approach than in the UWT and the exact outcomes are difficult to compare. In the UWT approach,

more parameters and features can be modeled than with our model. On the other hand, we are able to model the intruder as an intelligent adversary with our game approach. An advantage of our game approach compared with the UWT is that we do not have to specify the agent's tactics in advance. Therefore, our game approach generated unconventional tactics that may serve as a starting point for new simulations. Simulating the optimal strategy in the UWT shows that the detection probability is higher than when standard tactics are used. With our game approach we are able to generate agent's routes that may serve as a starting point for simulation and to give insights to the structure of new tactics.

The running time of our model increases in the number of cells. Therefore, the game theoretical approach not practically usable for large areas. In order to reduce the computational time, we have introduced aa branch and price algorithm, which generates solutions within 15% of the optimal solution. For future research, it would be interesting to improve the approximation algorithms.

5.6 Appendix

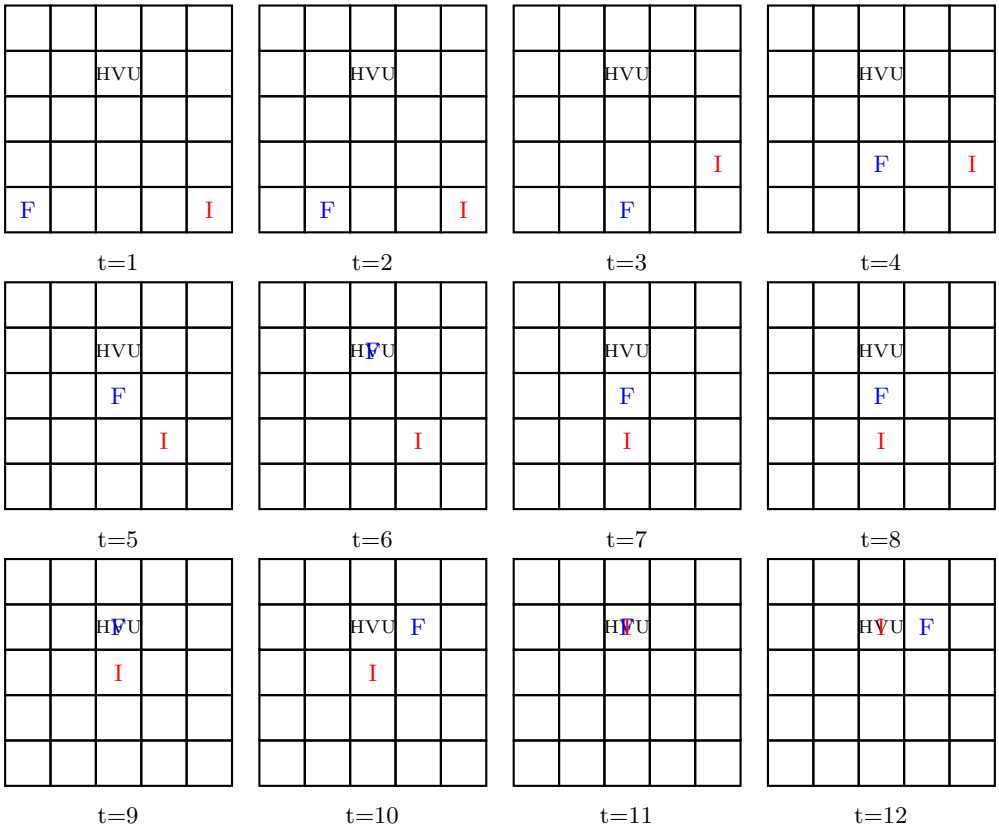


Figure 5.1: 1 frigates, 0 helicopters.

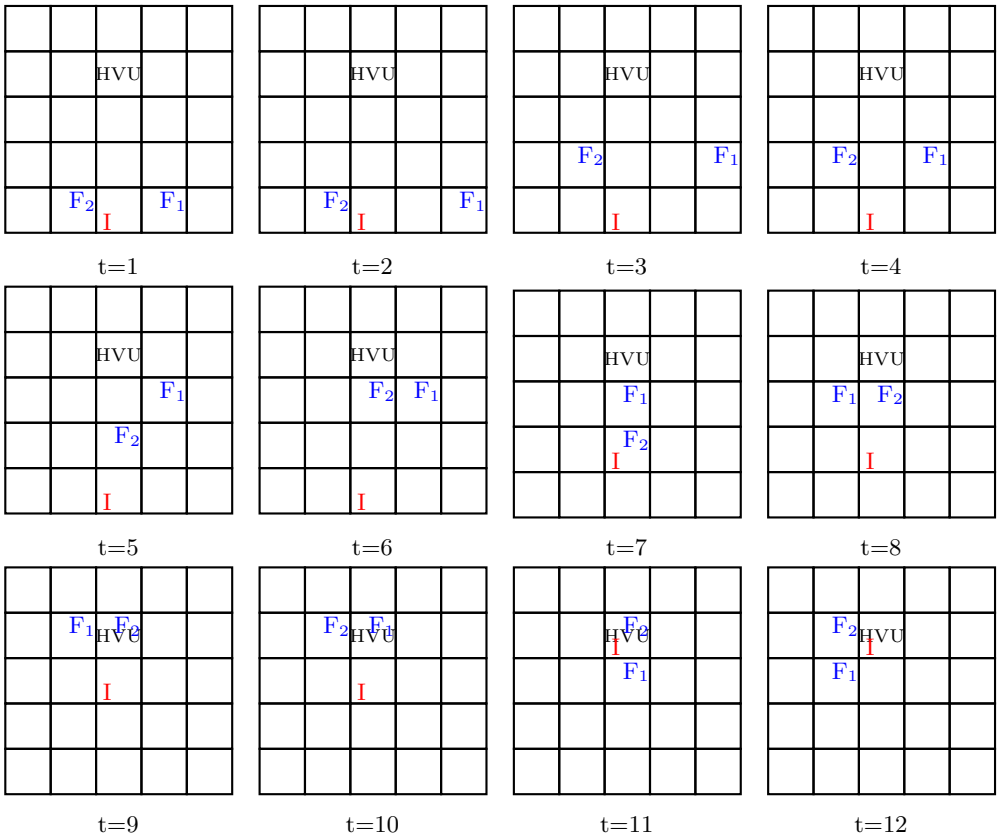


Figure 5.2: 2 frigates, 0 helicopters.

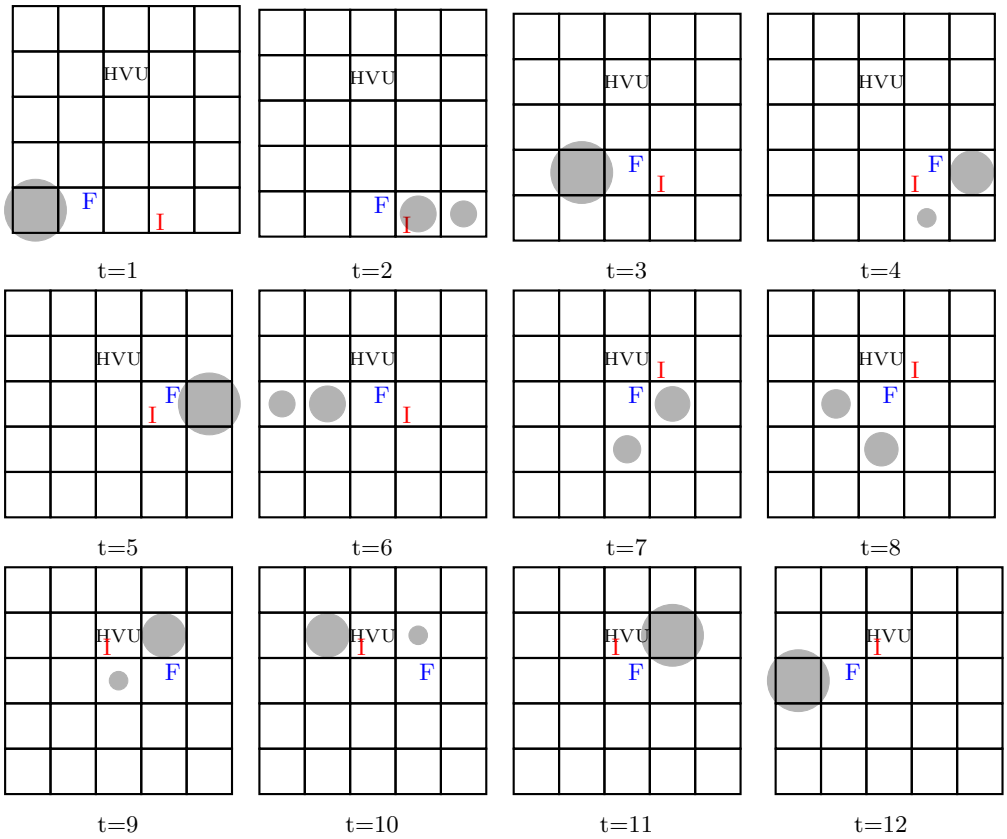


Figure 5.3: 1 frigates, 1 helicopters.

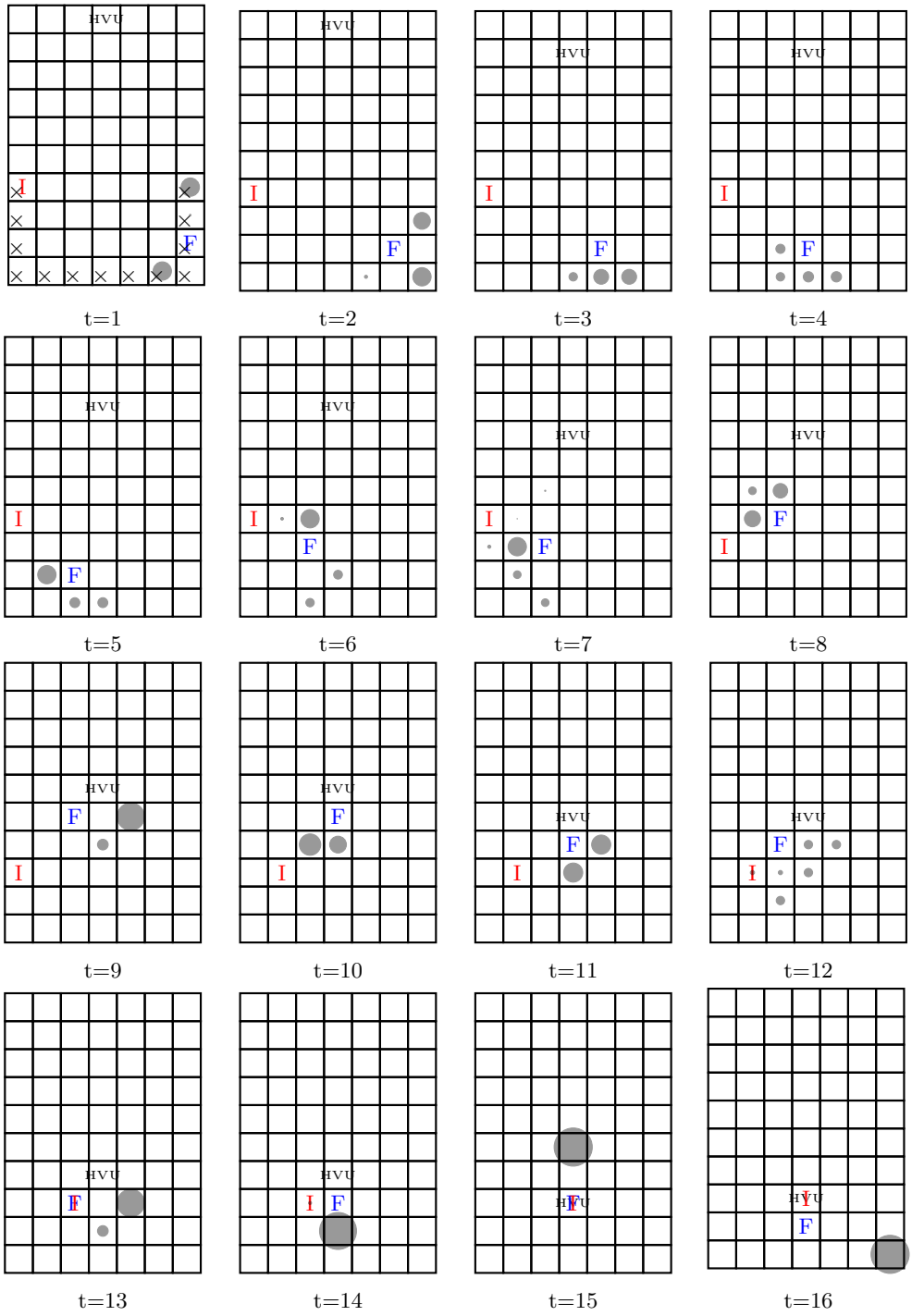
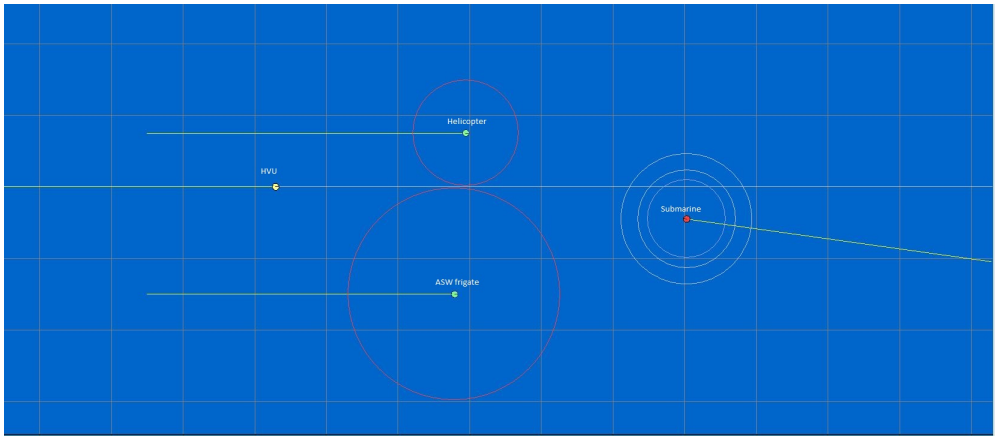
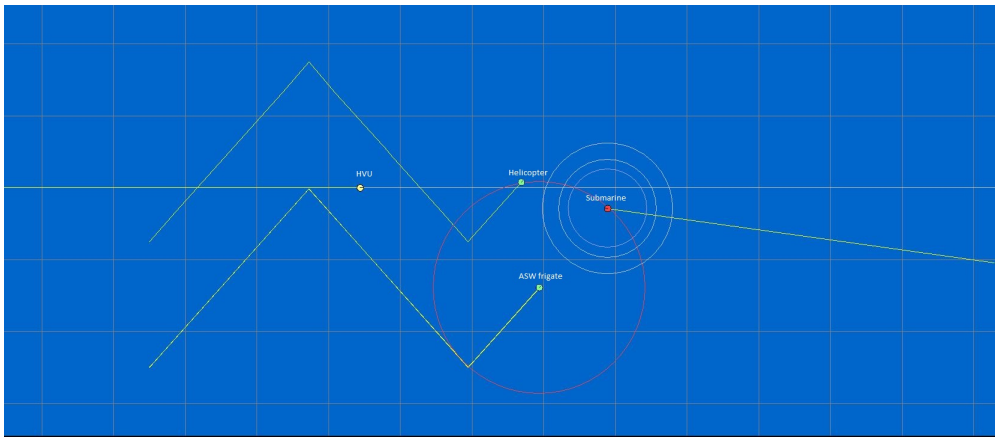


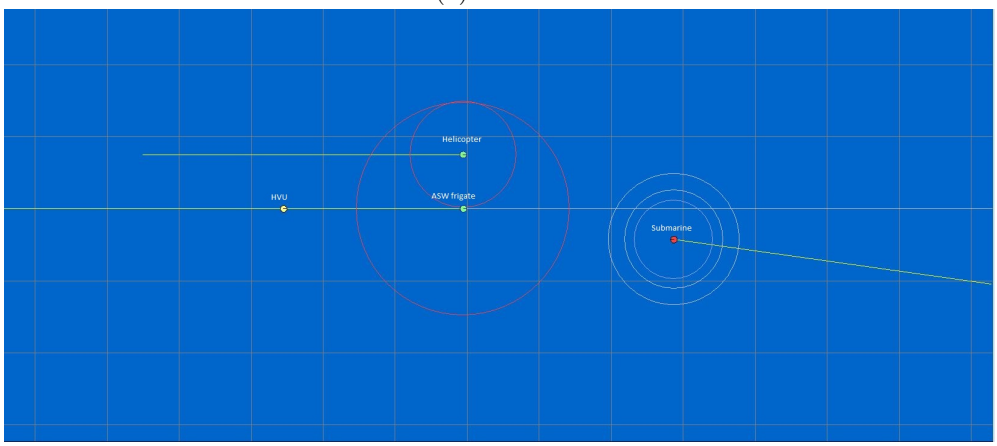
Figure 5.4: Large instance: 1 frigates, 1 helicopter.



(a) Tactic 1



(b) Tactic 2



(c) Tactic 3

Figure 5.5: Different agent tactics in the UWT simulation.

Part III

Security games with restrictions on the strategies

CHAPTER 6

Security games with probabilistic constraints on the agent's strategy

The results in this chapter were published in [70].

6.1 Introduction

The coast guard is responsible for patrolling the coastal waters. Patrolling strategies should be unpredictable, cover the entire region, and must satisfy operational requirements on e.g. the frequency of visits to certain vulnerable parts of the region (cells). We develop a special security game dealing with the protection of a large area in which the agent's strategy set is restricted. This area consists of multiple cells that have to be protected during a fixed time period. The agent has to decide on a patrolling strategy, which is constrained by governmental requirements that establish a minimum number of visits for each cell. Some cells have to be visited more often than others because these regions are more vulnerable. For example, cells close to a port have to be visited more often

In this chapter, we consider a static version of this problem. In the next chapter, a dynamic variant is considered. The problem can be modeled as a two-player zero-sum game with probabilistic constraints, which requires that the conditions are met with high probability. In practice, during the planning period, the payoff of attacking the cells can change over time. Therefore, we also introduce a variant of the basic constrained game in which the payoff may change over time.

In the literature there are several models considering patrolling games (e.g., [4, 39, 77]). Also many models consider constraints on the agent's or intruder's strategy set. For example in [18, 45, 125], the authors require constraints on the agent's strategy because only a limited number of resources is available, and in [130] the authors consider constraints on both the agent's and the intruder's strategy set.

Often, linear constraints are considered in constrained games. For instance, in [22] a two-person zero-sum game with linear constraints is introduced. More recently, [88] described a bimatrix game with linear constraints on the strategy of both players. In [115], the author considers nonlinear ratio type constraints. Our security game models situations where operational conditions have to be met with high probability, which results in nonlinear probabilistic constraints.

An example application of our model lies in countering illegal or unreported and

unregulated fishing. These illicit activities endanger the economy of the fishery sector, fish stocks and the marine environment and require the monitoring of large areas with scarce resources subject to national regulations. To support the development of patrols against illegal fishing, a decision support system is developed in [52]. This system models the interaction between different types of illegal fishermen and the patrolling forces as a repeated game. Moreover, in order to cope with the uncertainty on the adversary's strategy a robust optimization approach is used. More recently, [34] introduced a new game theoretical approach, the Green Security Games, wherein a generalization of Stackelberg games is used to derive sequential agent strategies that learn from adversary behavior. However, in these papers constraints, to the patroller's strategy are not considered.

The main contribution of this research is that we introduce a new model to cope with the conditions on the agent's random strategy that have to be met with high probability. The agent has conditions on the number of visits to each cell over the planning period. Because of the random nature of the strategies, it cannot be guaranteed that the conditions are always met. By introducing probabilistic constraints, we assure that the conditions are met with high probability. In practice the payoff matrices may change over time, in the fishery case, due to weather conditions, seasonal fluctuations or other circumstances. Therefore, we introduce an extension of the model to deal with multiple payoff matrices.

This chapter is organized as follows. In Section 6.2, we introduce the new security game model with constraints on the agent's strategy, wherein only a single payoff matrix during the planning period is considered. In Section 6.3, we present an extension of the model in which multiple payoff matrices are considered. Finally, in Section 6.4 we give examples of the model and present some computational results, and give some concluding remarks in Section 6.5.

6.2 Model with constant payoff

This section describes the model assuming that the gain an intruder obtains by successfully visiting a cell is constant over the planning period. We first provide a general description of a constrained security game over multiple cells in Section 6.2.1. For each cell, there is a condition on the minimal number of visits per time period for that cell. We discuss the probability that these conditions are met for each cell separately in Section 6.2.2, which gives a lower bound for the game value. In the application of countering illegal fishing, governmental guidelines require that some cells should be patrolled more than others because some regions are more vulnerable. The conditions on the number of cells have to be met for all cells simultaneously. These simultaneous conditions are discussed in Section 6.2.3.

6.2.1 Constrained game

We consider a security game with constraints on the strategy sets (see [96], Chapter 3.7). Let $C = \{1, ..., N_C\}$ be the set of cells that can be attacked by an intruder and let $R = \{1, ..., N_R\}$ be the set of routes that can be chosen by the agent. The matrix B indicates which cells are visited by each route, such that b_{ij} equals 1 if route i includes cell j and 0 otherwise. Let M be the payoff matrix, such that m_{ij} is the payoff for the intruder if the agent chooses route i and the intruder attacks cell j, $i = 1, ..., N_R$,

$j = 1, ..., N_C$. If the agent's chosen route i includes cell j chosen by the intruder, the intruder is caught with probability d_j. In this case the payoff is:

$$m_{ij} = ((1 - d_j)b_{ij} + (1 - b_{ij}))\, g_j, \quad i = 1, ..., N_R,\ j = 1, ..., N_C, \tag{6.1}$$

where g_j is the intruder's gain if the intruder successfully attacks cell j.

The game is repeated N_D times (e.g. days), our planning period. We assume that only one intruder is present in the area. If that intruder is caught, then another will replace him. The overall aim from an intruder's perspective is to maximize the total payoff over the time period.

Remark 6.1. Note that the model described in this section assumes that each intruder attacks only one cell each day. By changing the payoff matrix M and the actions of the agent and the intruder, the model can be extended to other matrix games. For example, if we want to allow the intruders to sequentially attack multiple cells, e.g., one in the morning and one in the afternoon, the intruder's action space becomes $C \times C$. If matrices A^m and A^a specify for each route which cells are visited in the morning and afternoon, respectively, the payoff matrix can be adjusted to:

$$m_{i(jk)} = \left((1 - d_j)a^m_{ij} + (1 - b_{ij})\right) g_j + ((1 - d_k)b_{ij} + (1 - b_{ik}))\, g_k,$$

$i = 1, ..., N_R,\ j, k = 1, ..., N_C$, where j and k are the cells attacked by the intruder in the morning and the afternoon. □

The intruder attempts to maximize the payoff by choosing which cell to attack, s the action set of the intruder is given by C. The agent tries to catch the intruder by selecting a route, so the action set of the agent is given by R. The agent minimizes the payoff by deciding on the probability p_i, $i = 1, ..., N_R$, that route i is chosen, while the intruder maximizes the payoff by selecting the probability q_j, $j = 1, ..., N_C$, that cell j is attacked. The strategy of the agent is constrained by the conditions $f(p) \geq 0$, determined by the minimum number of times each cell is visited by the agent. In Sections 6.2.2 and 6.2.3, we will elaborate on these conditions. The value of the game, V, equals the expected payoff per day. Optimal strategies can be found by solving the following mathematical program:

$$\begin{aligned} V = \min_p \max_q \quad & p^T M q \\ \text{s.t.} \quad & f(p) \geq 0, \\ & \sum_{i=1}^{N_R} p_i = 1, \\ & \sum_{j=1}^{N_C} q_j = 1, \\ & p, q \geq 0. \end{aligned} \tag{6.2}$$

Taking the dual of the inner linear program $\max_q \{p^T M q | \sum_{j=1}^{N_C} q_j = 1, q \geq 0\}$, the minmax formulation (6.2) can be rewritten to obtain the value of the game and optimal

strategies for the agent:

$$\begin{aligned} V = \min_{p,z} \quad & z \\ \text{s.t. } & e^T z \geq p^T M, \\ & f(p) \geq 0, \\ & \sum_{i=1}^{N_R} p_i = 1, \\ & p \geq 0, \end{aligned} \tag{6.3}$$

where e is the row vector with only ones. Note that there only exists a value for this game if the set $\{p | f(p) \geq 0, \sum_{i=1}^{N_R} p_i = 1, p \geq 0\}$ is not empty.

Remark 6.2. For clearness of presentation, we model the game as a zero-sum game. Note that a similar model applies if we consider a bimatrix game in which the agent and the intruder have different payoff matrices. In bimatrix games, the game value is calculated using quadratic programming (see for example [104], Chapter 13.2) instead of linear programming, but the probabilistic constraints can be implemented similarly. In addition, in the same manner, conditions on the intruder's strategy set can be added. □

6.2.2 Conditions on the number of visits to a cell

In this subsection, we consider conditions on the number of visits for each cell separately to obtain a lower bound for V. Let N_D be the number of days in the planning period. The strategy of the agent is constrained by the minimum number of visits v_j to each cell j, $j = 1, ..., N_C$, over the entire period N_D, that must be realized with at least probability $1 - \epsilon$. Given any strategy p, the probability that cell j is visited by the agent is $b_j p$, where b_j is the row vector of the j-th column of B.

Let X_j, $j = 1, ..., N_C$, be the random variable that records the number of visits to cell j during the planning period. The probability that cell j is visited equals $b_j p$. As there are N_D successive days, X_j is binomially distributed with parameters N_D and $b_j p$. The constraint on the number of visits then reads $P(X_j \geq v_j) \geq (1 - \epsilon)$, i.e.,

$$\sum_{k=v_j}^{N_D} \frac{N_D!}{k!(N_D - k)!} (b_j p)^k (1 - b_j p)^{N_D - k} \geq 1 - \epsilon,$$

which can be implemented in (6.3) by choosing $f(p) = (f_1(p), f_2(p), ... f_{N_C}(p))$ with $f_j(p) = P(X_j \geq v_j) - (1 - \epsilon)$.

For large N_D, the binomial distribution becomes intractable for implementation. Therefore, we use the following approximation. For large N_D, the binomially distributed X_j can be approximated by the normally distributed $\tilde{X}_j$ with mean $N_D b_j p$ and variance $N_D b_j p(1 - b_j p)$ (see [112], Chapter 1.8):

$$P(X_j \geq v_j) = 1 - P(X_j < v_j) \approx 1 - P(\tilde{X}_j \leq v_j),$$

yielding

$$f_j(p) = \epsilon - \Phi\left(\frac{v_j - N_D b_j p}{\sqrt{N_D b_j p(1 - b_j p)}} \right), \tag{6.4}$$

where $\Phi(x)$ is the cumulative distribution function for the standard normal distribution.

Considering the conditions for each cell separately gives a relaxation of the original conditions, where the minimum number of visits has to be obtained for all cells simultaneously. If we replace $f(p)$ in (6.3) by the constraints in (6.4), we obtain the following lower bound for the game value V:

$$\begin{aligned} V_L = \min_{p,z} \quad & z \\ \text{s.t.} \; & e^T z \geq p^T M, \\ & \Phi\left(\frac{v_j - N_D b_j p}{\sqrt{N_D b_j p(1-b_j p)}}\right) \leq \epsilon, \quad j = 1, ..., N_C, \\ & \sum_{i=1}^{N_R} p_i = 1, \\ & p \geq 0. \end{aligned} \tag{6.5}$$

In order to linearize these constraints, we determine for each cell j all possible values of $b_j p$ such that $\epsilon - P(\tilde{X}_j \leq v_j) \geq 0$. This is illustrated in the following example:

Example 6.1. Suppose the planning period is $N_D = 100$, $v_j = 40$ and $\epsilon = 0.05$. We are interested in all values of $b_j p$ such that:

$$\Phi\left(\frac{v_j - N_D b_j p}{\sqrt{N_D b_j p(1-b_j p)}}\right) \leq \epsilon.$$

Using the table of the standard normal distribution, this is the case when

$$\frac{v_j - N_D b_j p}{\sqrt{N_D b_j p(1-b_j p)}} \leq -1.65 \quad \Rightarrow \quad b_j p \geq 0.4772.$$

□

As described in the example above, the constraints in (6.5) can be replaced by the linear constraint $p^T A \geq \tilde{b}$, where $\tilde{b}_j$ is determined by the minimum probability for each cell such that the conditions are met with probability $1 - \epsilon$.

Visits to cells are correlated via the routes. Therefore, we are interested in the joint probability:

$$P(X_1 \geq v_1, X_2 \geq v_2, ..., X_{N_C} \geq v_{N_C}),$$

that we will discuss in the next section.

6.2.3 Conditions on all cells simultaneously

In this section, we discuss the conditions on the minimum number of visits for all cells simultaneously. Let Y_i, $i = 1, ..., N_R$, be the random variable that specifies the number of times that route i is selected. $Y = (Y_1, Y_2, ..., Y_{N_R})$ is multinomially distributed with parameters N_D and p:

$$P(Y_1 = v_1, Y_2 = v_2, ..., Y_{N_R} = v_{N_R}) = N_D! \prod_{i=1}^{N_R} \frac{p_i^{v_i}}{v_i!}.$$

For large N_D, Y can be approximated by the multivariate normally distributed $\tilde{Y}$ with the following expectation, variance and covariance (see [112], Chapter 1.8), $i = 1, ..., N_R$,:

$$E(\tilde{Y}_i) = N_D p_i,$$
$$Var(\tilde{Y}_i) = N_D p_i(1 - p_i),$$
$$Cov(\tilde{Y}_i, \tilde{Y}_{i'}) = -N_D p_i p_{i'}.$$

The number of times cell j is visited, X_j, can then be expressed as:

$$X_j = \sum_{i=1}^{N_R} b_{ij} Y_i,$$

and using the approximation $\tilde{Y}$ for Y, X_j can be approximated by a normally distributed $\tilde{X}_j$ with expectation, variance and covariance (see [112], Chapter 1.4), $j = 1, .., N_C$,:

$$E(\tilde{X}_j) = N_D b_j p, \tag{6.6}$$
$$Var(\tilde{X}_j) = N_D b_j p(1 - b_j p), \tag{6.7}$$
$$Cov(\tilde{X}_j, \tilde{X}_{j'}) = \sum_{i=1}^{N_R} \sum_{i'=1}^{N_R} b_{ij} b_{i'j'} Cov(\tilde{Y}_i, \tilde{Y}_{i'}). \tag{6.8}$$

The probability that the conditions are met for all cells is:

$$P(X_1 \geq v_1, X_2 \geq v_2, ..., X_{N_C} \geq v_{N_C}) \approx P(\tilde{X}_1 \geq v_1, \tilde{X}_2 \geq v_2, ..., \tilde{X}_{N_C} \geq v_{N_C})$$
$$= \frac{1}{\sqrt{|\Sigma|(2\pi)^{N_C}}} \int_{v_1}^{\infty} \int_{v_2}^{\infty} ... \int_{v_{N_C}}^{\infty} e^{-\frac{1}{2}(v-\mu)' \Sigma^{-1} (v-\mu)} dv_{N_C} ... dv_1, \tag{6.9}$$

where Σ is the covariance matrix and μ is a vector with all expected values. This can be implemented in (6.3) by choosing $f(p)$ as

$$f(p) = P(\tilde{X}_1 \geq v_1, \tilde{X}_2 \geq v_2, ..., \tilde{X}_{N_C} \geq v_{N_C}) - (1 - \epsilon). \tag{6.10}$$

The constraint described above is not linear and cumbersome to implement in a mathematical program. To simplify the model, we use a lower bound for the probability that the conditions are met and implement this lower bound.

A lower bound for the probability that the conditions for all cells are met is:

$$P(\tilde{X}_1 \geq v_1, \tilde{X}_2 \geq v_2, ..., \tilde{X}_{N_C} \geq v_{N_C})$$
$$= 1 - P(\tilde{X}_1 < v_1 \vee \tilde{X}_2 < v_2 \vee ... \vee \tilde{X}_{N_C} < v_{N_C})$$
$$\geq 1 - \sum_{j=1}^{N_C} P(\tilde{X}_j < v_j). \tag{6.11}$$

This lower bound can be used to simplify the mathematical program as follows:

$$f(p) = \epsilon - \sum_{j=1}^{N_C} \Phi \left(\frac{v_j - N_D b_j p}{\sqrt{N_D b_j p(1 - b_j p)}} \right). \tag{6.12}$$

Replacing $f(p)$ in (6.3) by a lower bound in the condition, results in an upper bound for the game value V:

$$\begin{aligned} V_U = \min_{p,z} \quad & z \\ \text{s.t. } & e^T z \geq p^T M, \\ & \sum_{i=1}^{N_R} p_i = 1, \\ & \sum_{j=1}^{N_C} \Phi\left(\frac{v_j - N_D b_j p}{\sqrt{N_D b_j p(1-b_j p)}}\right) \geq \epsilon, \\ & p \geq 0, \end{aligned} \tag{6.13}$$

Combining this upper bound and the lower bound obtained in Section 6.2.2, we readily obtain the following result:

Lemma 6.1. *For V_L given in* (6.5) *and V_U given in* (6.13) *we have:*

$$V_L \leq V \leq V_U.$$

In Section 6.4, we investigate the impact of this approximation modeling approach on the game value.

Remark 6.3. We may linearize this program by approximating the normal distribution for each cell j by a piecewise linear function $\tilde{h}_j$ with N_B breakpoints b_l, $l = 1, ..., N_B$ (see [128], Chapter 9.2):

$$\begin{aligned} \min_{p,z} \quad & z \\ \text{s.t. } & e^T z \geq p^T M, \\ & \sum_{i=1}^{N_R} p_i = 1, \\ & Bp = c, \\ & 1 - \sum_{j=1}^{N_C} \tilde{f}_j \geq 1 - \epsilon, \\ & \sum_{l=0}^{N_b} \lambda_{lj} h_j(b_l) = \tilde{h}_j, \qquad j = 1, ..., N_C, \\ & \sum_{l=0}^{N_b} \lambda_{lj} b_l = c_j, \qquad j = 1, ..., N_C, \\ & \sum_{l=0}^{N_b} \lambda_{lj} = 1, \qquad j = 1, ..., N_C, \\ & \lambda_{lj} \leq y_{lj} + y_{(l+1)j}, \qquad j = 1, ..., N_C, l = 1, ..., N_B, \\ & \sum_{l=0}^{N_b} y_{lj} = 1, \qquad j = 1, ..., N_C, \end{aligned}$$

$$p, \lambda \geq 0,\ y_{lj} \in \{0,1\}, \quad j = 1, ..., N_C, l = 1, ..., N_B,$$

where $h_j(b_l) = \Phi\left(\frac{v_j - N_D c_j}{\sqrt{N_D c_j (1-c_j)}}\right)$. However, we use the mathematical program stated in (6.13) in the result section since this model is still solvable for realistic instances.□

6.3 Generalization: multiple payoff matrices

The previous section considers games with constant payoff. This section considers a generalization to situations where payoff can change over time due to, e.g., weather conditions or seasonal fluctuations resulting in multiple payoff matrices.

6.3.1 Constrained game

Consider the game with multiple payoff matrices $M^{(k)}$, $k = 1, ..., N_M$, of size $N_R \times N_C$. Let $\mu^{(k)}$ be the probability that the payoff matrix is $M^{(k)}$, with $\sum_{k=1}^{N_M} \mu^{(k)} = 1$. Moreover let $q^{(k)}$ and $p^{(k)}$ be strategies of the agent and the intruder when the payoff matrix is $M^{(k)}$. The value of the game is the expected payoff per day and can be found by solving the following optimization problem:

$$\begin{aligned} V = \min_p \max_q \quad & \sum_{k=1}^{N_M} \mu^{(k)} (p^{(k)})^T M^{(k)} q^{(k)} \\ \text{s.t. } & f(p) \geq 0, \\ & \sum_{i=1}^{N_R} p_i^{(k)} = 1, \qquad k = 1, ..., N_M, \\ & \sum_{i=1}^{N_C} q_i^{(k)} = 1, \qquad k = 1, ..., N_M, \\ & p, q \geq 0, \end{aligned} \tag{6.14}$$

where $p^T = (p^{(1)}, ..., p^{(N_M)})$ and $q^T = (q^{(1)}, ..., q^{(N_M)})$. In the next section, we discuss how the constraint $f(p) \geq 0$ changes if multiple payoff matrices are considered.

6.3.2 Conditions for games with multiple payoff matrices

The conditions on the minimal number of visits for all cells during the planning period can be constructed following the same reasoning as in Section 6.2. Now, the number of visits for cell j is the sum of the number of visits for cell j for each payoff matrix. Let $X_j^{(k)}$, $j = 1, ..., N_C$, $k = 1, ..N_M$, be the random variable describing the number of visits to cell j when the payoff matrix is M_k and let $\tilde{X}_j^{(k)}$ be the approximation of $X_j^{(k)}$. $N_D^{(k)}$ is the number of periods that the payoff matrix is $M^{(k)}$. We are interested in the following probability:

$$P(\tilde{X}_1^{(1)} + ... + \tilde{X}_1^{(N_M)} \geq v_1, \tilde{X}_2^{(1)} + ... + \tilde{X}_2^{(N_M)} \geq v_2, ..., \tilde{X}_{N_C}^{(1)} + ... + \tilde{X}_{N_C}^{(N_M)} \geq v_{N_C}),$$

with $E(\tilde{X}_j^{(k)})$, $Var(\tilde{X}_j^{(k)})$, and $Cov(\tilde{X}_j^{(k)})$ calculated as in Section 6.2.3 (Equations (6.6)-(6.8)), but with $N_D^{(k)}$ and $p^{(k)}$, instead of N_D and p. Since $\tilde{X}_j^{(k)}$ and $\tilde{X}_{j'}^{(k)}$ are independent if $j \neq j'$, we have:

$$E(\tilde{X}_j) = \sum_{k=1}^{N_M} N_D^{(k)} b_j p^{(k)},$$
$$Var(\tilde{X}_j) = \sum_{k=1}^{N_M} N_D^{(k)} b_j p^{(k)} (1 - b_j p^{(k)}),$$
$$Cov(\tilde{X}_j, \tilde{X}_{j'}) = \sum_{k=1}^{N_M} \sum_{k'=1}^{N_M} Cov(X_j^{(k)}, X_{j'}^{(k')}).$$

To make sure that the conditions are met with high probability we define,

$$f(p) = P(\tilde{X}_1 \geq v_1, \tilde{X}_2 \geq v_2, ..., \tilde{X}_{N_C} \geq v_{N_C}) - (1 - \epsilon),$$

where $P(\tilde{X}_1 \geq v_1, \tilde{X}_2 \geq v_2, ..., \tilde{X}_{N_C} \geq v_{N_C})$ equals (6.9) with Σ and μ the covariance matrix and expected value as described in the above equations. Similarly as in Section 6.2.3, a lower bound for this probability is given in (6.11). Taking the dual of the inner LP of (6.14) and using this lower bound, optimal strategies for the agent and the intruder can be found by:

$$\begin{aligned}
V_L = \min_{p,z} \quad & \sum_{k=1}^{N_M} z^{(k)} \\
\text{s.t. } & e^T z^{(k)} \geq \mu^{(k)} (p^{(k)})^T M^{(k)}, && k = 1, ..., N_M. \\
& \sum_{j=1}^{N_C} \Phi \left(\frac{v_j - \sum_{k=1}^{N_M} N_D^{(k)} b_j p^{(k)}}{\sqrt{\sum_{k=1}^{N_M} N_D^{(k)} b_j p^{(k)} (1 - b_j p^{(k)})}} \right) \geq \epsilon, && (6.15) \\
& \sum_{i=1}^{N_R} p_i^{(k)} = 1, && k = 1, ..., N_M, \\
& p, q \geq 0,
\end{aligned}$$

where $z = (z^{(1)}, ..., z^{(N_M)})$. In the next section, we will show some examples to illustrate this model.

6.4 Results

In this section, we give computational results and examples to illustrate our models. In Section 6.4.1, we compare the models discussed in this chapter to investigate the approximation error introduced in Section 6.2.3. Thereafter, we give two examples that illustrate our model in Section 6.4.2.

6.4.1 Computational results

This section investigates the error introduced by using the lower bound in (6.11). Solving (6.3) with $f(p)$ given in (6.10) numerically is computationally intractable for

networks with more than two or three routes and cells. Therefore, we have compared the relative difference between the lower and upper bounds of V, see Lemma 6.1. We have randomly generated 100 payoff matrices, conditions and routes for different network sizes. Table 6.1 shows the average relative difference between the upper and lower bound with 0.95%-confidence interval between brackets. The last columns gives the average running time in seconds for (6.13). The results are implemented in Matlab version R2016b [85] on an Intel(R) Core(TM) i7 CPU, 2.4GHz, 8 GB of RAM. As the

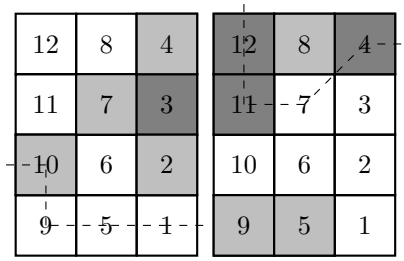
Table 6.1: Average relative difference between upper bound V_U and lower bound V_L ($\epsilon = 0.05$).

# Cells	# Routes	Error	Running time
10	5	0.8% (± 1.0%)	0.217 s
20	15	1.9% (± 1.8%)	0.347 s
30	25	2.2% (± 1.4%)	0.819 s

results in Table 6.1 show, (6.13) gives a good approximation of the game value V and can be solved in reasonable time. The size of more realistic examples, as encountered in the patrolling against illegal fishing context, is comparable to the size of these randomly generated instances.

6.4.2 Illustrative examples

In this section, we present some examples to illustrate the models described in this chapter. The results in this section are obtained by implementation of (6.13) and (6.15). Consider an area with 12 cells and 9 routes. The routes are chosen such that the cells are evenly spread over all routes, see Figure 6.1. Suppose $N_M = 2$ and the payoff matrices are constructed using (6.1), where $d_j = 0.9$, $j = 1, ..., N_C$ and $g^{(k)}$ is the intruder's gain. Figure 6.1 depicts payoff matrices $M^{(1)}$, $M^{(2)}$ and two example routes, Routes 1 and 8. The white cells have a gain of 1, the light gray cells have a gain of 2 and the dark gray cell have a gain of 3.



Routes	Cells visited by route
1	1, 5, 9, 10
2	2, 3, 8, 12
3	3, 7, 6, 10
4	4, 7, 6, 9
5	1, 2, 3, 4
6	3, 4, 7, 12
7	2, 5, 6, 9
8	4, 7, 11, 12
9	2, 5, 10, 11

Figure 6.1: Payoff matrices and routes.

Constant payoff matrix

Consider the games with payoff matrices $M^{(1)}$ and $M^{(2)}$ separately. Suppose that the planning period for both payoff matrices is $N_D = 100$. Table 6.2 shows the game values for different conditions. For example, a condition of 0.1 means that the minimum number of visits equals 10. The second and the third column give the game value of both games for the conditions specified in the first column. The first row shows the value of the game without conditions on the number of visits to the cells, the second row considers the game in which all nodes must be visited at least 10 times, and the third row considers the game in which nodes 1-4 must be visited at least 30 times and the other nodes at least 10 times.

Table 6.2: Expected payoff per day for different conditions ($\epsilon = 0.05$).

Conditions (fraction)	Payoff $M^{(1)}$	Payoff $M^{(2)}$	Average	Combined
None	1.10	1.58	1.34	1.34
All nodes: 0.1	1.23	1.64	1.44	1.34
1-4: 0.3, 5-12: 0.1	1.23	2.14	1.69	1.35

Table 6.2 indicates that if more conditions are imposed on the agent's strategy, the game value will increase. However, the increase of the game value depends on the payoff matrix. For example, the extra condition on nodes 1-4 does not increase the game value for payoff matrix $M^{(1)}$, since the intruder's gain for these nodes is high and the agent is already patrolling these cells more often, as the results below indicate.

Figure 6.2 displays the agent's strategy for the different payoff matrices without conditions. The color of each cell is determined by the gain of the intruder and the number within each cell shows the fraction of the time period that the cell should be visited. The agent's strategy is shown by the circles in each cell. The probability that a cell is visited is proportional to the radius of the circle in that specific cell. For example in Figure 6.2, the probability that cell 3 is visited equals 1 for $M^{(1)}$ and 0.24 for $M^{(2)}$. Figure 6.3 displays the agent's strategy when conditions as given in Table 6.2 are considered. For all cases, it is clear that cells with a high gain for the intruder are visited more often.

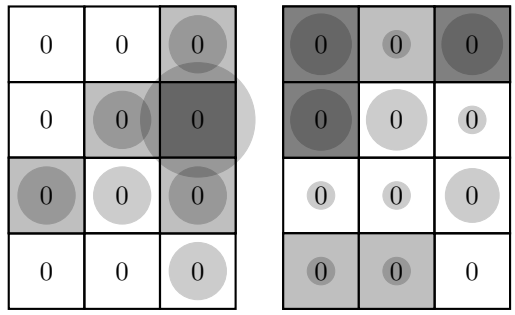
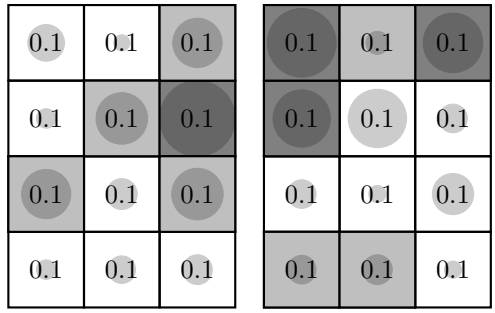
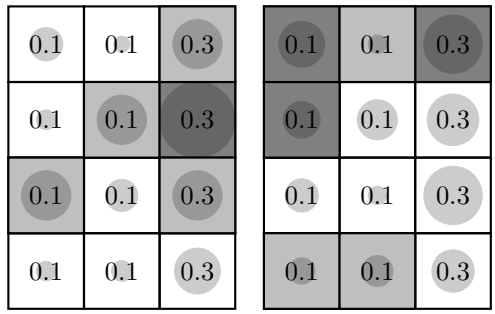


Figure 6.2: Agent's strategy for the game without conditions.



(a) All nodes: 0.1



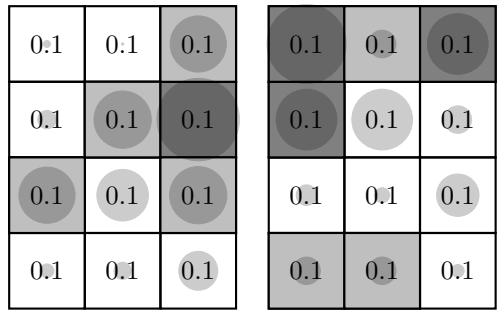
(b) Nodes 1-4: 0.3, nodes 5-12: 0.1

Figure 6.3: Agent's strategy for different conditions.

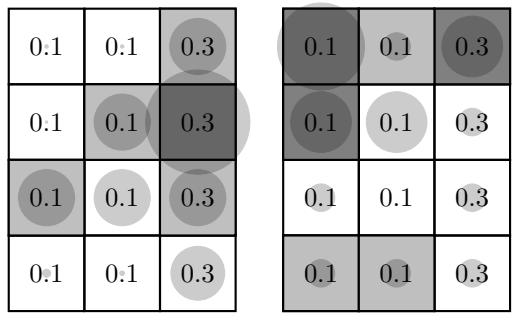
Multiple payoff matrices

The previous example considers the game with a constant payoff matrix such that for each game the conditions on the minimum number of visits have to be met. Now, we consider multiple payoff matrices simultaneously. Suppose that the total planning period $N_D = 200$ and both payoff matrices $M^{(1)}$ and $M^{(2)}$ have equal probability, so $\mu^{(1)} = \mu^{(2)} = 0.5$. Again, routes and conditions are given in Figure 6.1 and Table 6.2. A condition of 0.1 means that the total number of visits is 20, but it is, for example, allowed that there are only 5 visits when the payoff matrix is $M^{(1)}$ and 15 when the payoff matrix is $M^{(2)}$. This is the benefit of playing the game repeatedly and considering multiple payoff matrices simultaneously. In the last column of Table 6.2 the value of the game in which the conditions are combined for multiple payoff matrices is shown. If there are no conditions on the number of visits to the cells, the game value is just the average of both games with constant payoff, which is shown in the second last column of Table 6.2. However, when conditions are considered, the value of the combined game is lower than the average of both games with constant payoff, because the agent has more flexibility in meeting the conditions.

Figure 6.4 shows the agent's strategy for the combined game with conditions given in Table 6.2. Comparing the results with those in Figure 6.3 reveals that the agent has more flexibility in meeting the constraints when multiple payoff matrices are considered. Indeed the agent visits a cell less often when the gain is low and compensates this lack of visits when the gain of that cell is high.



(a) All nodes: 0.1



(b) Nodes 1-4: 0.3, nodes 5-12: 0.1

Figure 6.4: Agent's strategy if multiple payoff matrices are considered simultaneously.

6.5 Concluding remarks

Patrolling a region with conditions on the frequency of visits to specific parts of that area while taking into account the optimal payoff of the intruder or agent can be modeled as a zero-sum security game with probabilistic constraints on the agent's strategy. These constraints prohibit exact solutions for large (realistic) instances. Therefore, we have developed a model yielding an upper bound and a lower bound for the game value. Computational results reveal that the relative difference between the upper and lower bound for the instances considered is less than 2.5% and that instances of realistic size can be solved within seconds.

In practice, the agent's strategy is constrained by existing guidelines. Numerical examples show that as the number of conditions increases, the agent's loss will increase. However, if multiple payoff matrices are considered, the agent has more flexibility in meeting the conditions and the loss of the agent is reduced.

In this chapter, we have assumed that only one intruder is present in the area, that the payoff of intruders is known and that the agent decides on a strategy in advance. For future research, it would be interesting to investigate the case where not all payoff matrices are known in advance and multiple intruders attack simultaneously. Also, considering a more dynamic strategy of the agent, for example by taking into account extra information about the payoff and cells that already have been visited, should be pursued.

CHAPTER 7

Security games with restricted strategies: an approximate dynamic programming approach

The results in this chapter were published in [71].

7.1 Introduction

In the previous chapter, we considered a special case of a security game dealing with the protection of a large area for a given time period where the agent's strategy set is restricted. The large area consists of several cells containing assets to be protected. An intruder decides on which cell to attack, while the agent needs to select a patrol route that visits multiple cells. The agent's strategy is constrained by existing governmental guidelines that require that some cells should be patrolled more often than others. This problem is modeled as a two-player zero-sum game with probabilistic constraints, which requires that the conditions are met with high probability.

The model described in Chapter 6 is a static version of the problem, where a strategy for the complete time period is identified before the game starts. The requirements are modeled such that they are met with high probability. However, that model does not allow patrolling strategies adjusted to the current situation. In this chapter, we consider a dynamic approach to the security game with restricted strategies in which the agent decides on his strategy for each day taking into account expected future rewards. This allows finding a more flexible strategy for the agent, where current payoffs and number of visits to each cell can be taken into account.

We model the dynamic variant of the security game with restricted strategies as a finite-time stochastic game in which the state depends on both the current payoff matrix and the remaining minimum number of visits left to each cell. The direct reward is given by the intruder's payoff and at the end of the time period a penalty is given if the operational requirements are not met. Solving stochastic games can be done by iterating over all states and time periods. However, the state space grows exponentially in the number of cells and we are unable to solve realistic sized games. Therefore, we develop an approximate dynamic programming (ADP) approach to find approximate solutions.

Due to the curse of dimensionality, many stochastic optimization models cannot

be solved by iterating over all possible states. ADP is a technique that can be used to solve large scale Markov decision processes (MDPs). We develop an ADP framework to find approximate solutions for our stochastic game. A brief introduction to ADP can be found in [107] and various examples are given in [89]. In the ADP framework, the optimal solutions are not found using standard backward dynamic programming, but by using a forward dynamic programming approach over only a fixed number of iterations. In this forward approach, different value function approximations can be used. In this chapter, we use multiple aggregation levels of the state space to approximate the value functions as discussed in [43]. In the basic ADP algorithm, only a very limited number of states will be updated during each iteration. In our method using aggregation of the state space, multiple value function approximations are updated at the same time, possibly with different weight for different aggregation states. In this way, the value functions are updated more often and will converge faster.

Although most of the research in ADP focuses on solving MDPs, some models focus on solving games. In [103], the authors consider the error propagation for different approximation schemes of zero-sum stochastic games. However, this paper does not provide a clear procedure that can be used to solve stochastic games using ADP. A solution technique that is very similar to ADP is reinforcement learning (RL), see for example [21]. The main difference between ADP and RL is that RL is considered to be model-free, which means that information about transition probabilities is not necessarily required. In the field of RL, there is also limited research about applications to stochastic games. In [79], the authors use RL to approximate unknown rewards and use an iterative algorithm to find a policy for both players, while we are interested in calculating this policy using approximation algorithms.

The main contribution of this chapter is twofold. First, we develop a model to solve security games with restrictions on the agent's strategy. Formulating this model as a stochastic game enables the agent to adjust the strategy to the current situation and actions that already have been chosen in the past. Second, we approximate optimal solutions of this stochastic game via an ADP approach. We adjust the standard ADP model that is often used to solve large scale MDPs to analyze stochastic games. Experimental results show that this method gives better payoffs for the agent than using a static approach where strategies are fixed for the complete planning period.

The remainder of this chapter is organized as follows. In Section 7.2, we introduce the model and give the elements of the stochastic game. In Section 7.3, we first give a brief introduction to ADP and then describe our formulation for stochastic games. In Section 7.4, we give computational results and compare the static and dynamic approach. Finally in Section 7.5 we summarize the main findings and provide directions for future research.

7.2 Model description

In this section, we give the formulation of the security game with restrictions on the agent's strategy. We first describe the basic model in Section 7.2.1. In Chapter 6, we have explained the solution method that is used to solve this game with a static strategy for the complete time period. In Section 7.2.2, we describe a new stochastic game approach which is used to analyze strategies over the entire planning period.

7.2.1 Basic model

The game is played between an agent and an intruder over a time period of N_D days. The area is given by a finite set of cells $C = \{1, ..., N_C\}$. Each day, an intruder selects one cell to attack while the agent chooses a route from a finite set of routes $R = \{1, ..., N_R\}$. The agent and intruder choose their actions simultaneously. Routes consist of multiple cells where the agent is allowed to move between adjacent cells. The matrix B indicates which cells are visited by each route, such that b_{ij} equals 1 if route i visits cell j and 0 otherwise.

The risk of a cell is displayed in the payoff matrices. Cells with a high risk have higher payoffs than low risk cells. This payoff is interpreted as the intruder's gain: the higher the gain for the intruder, the higher the probability that the intruder will attack there. The payoff matrix can change over time, due to, e.g., weather conditions or seasonal fluctuations, resulting in multiple payoff matrices. We assume that we have some information about how these payoff matrices change. Let $M^{(k)}$ be the k-th payoff matrix of size $N_R \times N_C$ out of a finite set of payoff matrices, $k = 1, ..., N_M$. The element $m_{ij}^{(k)}$ is the expected payoff if the agent uses route i and the intruder attacks cell j, $i = 1, ..., N_R$, $j = 1, ..., N_C$. We consider the payoff given by the intruder's expected gain:

$$m_{ij}^{(k)} = ((1 - d_j)b_{ij} + (1 - b_{ij}))\, g_j^{(k)}, \quad i = 1, ..., N_R,\ j = 1, ..., N_C, k = 1, ..., N_M,$$

where d_j is the detection probability for cell j and $g_j^{(k)}$ is the intruder's gain if the intruder successfully attacks cell j. If the agent successfully intercepts the intruder, the payoff is 0.

There are operational requirements on the number of visits to the cells: the agent's strategy is restricted by the requirements that impose a minimum number of visits v_j for each cell j, $j = 1, ..., N_C$. During the time period, the agent has to decide on his actions such that cell j is visited at least j times.

Note that the model described in this chaper only describes a basic security game with one intruder. The methods developed in this chapter may be applied to extensions to matrix games obtained by changing the payoff matrices, such as including more (cooperating) intruders or detection probabilities depending on the cell or chosen action.

7.2.2 Dynamic approach

When considering a dynamic approach, strategies can change during the time window depending on the current payoff matrix and the number of times each cell already has been visited. We model this as a finite-time zero-sum stochastic game. We now describe the elements of this game.

The state space S of the game is given by the current payoff matrix and the number of visits that are still required for each cell:

$$S = \{s | s = (k, \bar{v}_1, ..., \bar{v}_{N_C}),\ k = 1, ..., N_M,\ 0 \leq \bar{v}_j \leq v_j,\ j \in C\}.$$

The action space of the agent and intruder are given by A_A and A_I. The intruder attempts to maximize the payoff by choosing which cell to attack, so the action set of the intruder is given by C. The agent tries to catch the intruder by selecting a route,

so the action set of the agent is given by R:

$$A_A = R, \quad A_I = C.$$

The matrix $\tilde{P}$ gives the transitions between the payoff matrices. These transitions do not depend on the actions of the agent and the intruder. If the current payoff matrix is $M^{(k)}$, then with probability t_{kl} the next payoff matrix is $M^{(l)}$. The transition matrix of the game P depends on both $\tilde{P}$ and the action i of the agent:

$$P(s'|s,i) = \begin{cases} \tilde{p}_{kl}, & \text{if } \bar{v}'_j = \max\{\bar{v}_j - b_{ij}, 0\}, \text{ for all } j \in C, \\ 0, & \text{otherwise.} \end{cases}$$

where $s = (k, \bar{v}_1, ..., \bar{v}_{N_C})$ and $s' = (l, \bar{v}'_1, ..., \bar{v}'_{N_C})$ and i is the agents action.

The direct reward is given by $R(s,(i,j))$ and depends on the agent's strategy i, the intruder's strategy j and the current payoff matrix $M^{(k)}$:

$$R(s,(i,j)) = m_{ij}^{(k)}.$$

To ensure that the requirements are met, we introduce a final reward which is a penalty for the requirements that are not met. This can either be a penalty for each requirement that is not met or one penalty if the requirements are not met. For the results considered in this chapter, we consider the last option:

$$R_f(s) = \begin{cases} \bar{B}, & \text{if } \sum_{j \in C} \bar{v}_j > 0, \\ 0, & \text{otherwise,} \end{cases}$$

where $\bar{B}$ is chosen large enough such that it is never beneficial to violate one of the requirements.

Optimal strategies can be found by solving the game iteratively (see [96], Chapter V.3). Let $V_t(s)$ be the game value at time period t, when the game is in state s.

$$V_{N_D}(s) = \text{Val}\left(M^{(k)} + \sum_{s' \in S} P(s'|s,\cdot) R_f(s') \right), \tag{7.1}$$

$$V_t(s) = \text{Val}\left(M^{(k)} + \sum_{s' \in S} P(s'|s,\cdot) V_{t+1}(s') \right), \quad t < N_D. \tag{7.2}$$

where $s = (k, \bar{v}_1, ..., \bar{v}_{N_C})$, so $M_{(k)}$ depends on the first element of state s, and $P(s'|s,\cdot)$ is the matrix consisting of the values $P(s'|s,i)$ for all agent's actions. The expression between brackets defines a matrix game. Val gives the value of this matrix game, so this is the game value when both players choose a strategy corresponding to a Nash equilibrium.

Solving equations (7.1) and (7.2) will give an optimal value of the game. However, the size of the state space is exponentially increasing in the number of cells and conditions and we are unable to solve these equations analytically. In the next section, we present a model to deal with this large state space.

7.3 Solution approach: approximate dynamic programming

In this section, we present a method that can be used to overcome the large state space of the stochastic game formulation in Section 7.2. Approximate dynamic programming (ADP) is a technique that is often used to solve large scale MDPs. In Section 7.3.1, we give a short introduction in ADP for solving MDPs based on [107]. In Section 7.3.2, we develop ADP to solve our stochastic game.

7.3.1 Introduction to ADP

Consider an MDP over time horizon T, with states s_t, actions a_t, transition matrix P, and cost functions C_t. The value of an MDP can be found by solving the Bellman equations:

$$V_t(s_t) = \min_{x_t} \left(C_t(s_t, x_t) + \sum_{s_{t+1}} P(s_{t+1}|s_t, x_t) V_{t+1}(s_{t+1}) \right).$$

When the state space is large, solving the Bellman equations is too time consuming. The main idea of ADP is not to solve the model by enumerating over all possible states but only over a limited number of states using a forward dynamic programming approach over a fixed number of iterations N. For each iteration, the random information is sampled using Monte Carlo experiments.

The random information that is revealed after action a_t is chosen, is given by w_{t+1}. Both the action and the random information define the next state. For ADP, the post-decision state s_t^a is introduced. A post-decision state is the state after an action a_t is chosen, but before the new random information w_{t+1} is revealed:

$$V_t(s_t) = \min_{a_t} \left(C_t(s_t, a_t) + V_t^a(s_t^a) \right),$$
$$V_t^a(s_t^a) = \sum_{w_{t+1}} P(w_{t+1}) V_{t+1}(s_{t+1}|s_t^a, w_{t+1}).$$

By the use of the post-decision state, we only have to evaluate the possible outcomes over w_{t+1} for each action and not over all possible states s_{t+1}. This decreases the number of possible outcomes that have to be evaluated during each iteration significantly.

The output of the algorithm is an approximation $\bar{V}_t^n(s_t^a)$ of the value of the post-decision states. During each step, the approximation $\bar{V}$ is updated using the following update rule:

$$\bar{V}_t^n(s_t) = \begin{cases} (1-\alpha)\bar{V}_t^{n-1}(s_t^n) + \alpha \hat{v}_t^n, & \text{if } s_t = s_t^n, \\ \bar{V}_t^{n-1}(s_t), & \text{otherwise.} \end{cases} \tag{7.3}$$

where α is a step size between 0 and 1. In the next section, we discuss the value of α.

The basic structure of an ADP is given by the following algorithm.

Algorithm 5 ADP algorithm

1: Initialize:

- Choose an initial approximation $\bar{V}_t^0(s_t^a)$ for each t.
- Set $n = 1$ and choose an initial state s_0^1.

2: Choose a sample path $w^n = (w_1^n, ..., w_T^n)$.

3: For $t = 0, ..., T$

- Solve

$$\hat{v}_t^n = \min_{a_t} \left(C_t(s_t^n, a_t) + \mathbb{E}^w \left(\bar{V}_{t+1}^{n-1}(s_t | s_t^{a_t}, w) \right) \right),$$

  and let a_t^n be the action that solves this minimization.
- Update $\bar{V}_t^n(s_t)$ using (7.3).
- Compute the next state to visit from the action a_t^n.

4: Set $n = n + 1$ and go to Step 2.

This algorithm is a basic outline and will in general not always give good approximation results. There are some methods for improving the algorithm, mainly in the step of choosing the next state (random or not), the choice of α and in the steps of the value function approximation. We discuss these methods in the next section.

7.3.2 ADP for a stochastic game

The ADP approach described in the previous section is used to solve large scale MDPs. In this section, we describe the adjustments we make to the ADP to solve the stochastic game in Section 7.2.2. The difference with MDPs is that we deal with multiple players. Therefore, the Bellman equations are replaced by:

$$V_t(s) = \text{Val} \left(M^k + \sum_{s' \in S} P(s'|s, \cdot) V_{t+1}(s') \right).$$

As a consequence, we have to optimize over both the intruder's and the agent's actions. However, in our case, the next state does not depend on the intruder's action. Therefore we can use an ADP algorithm similar to the ADP algorithm which is used to solve MDPs.

Due to the introduction of multiple players, we are not dealing with discrete actions. Both agent and intruder choose a probability distribution over the action spaces at each time step. Therefore, we are not able to calculate a value for each combination of actions and states. We modify the formulation and use of the post-decision state, which in our case only depends on the agent's actions. Let s_t^i be the post-decision state at time t and state s when the agent chooses pure strategy $i \in R$:

$$\bar{V}_t(s_t^i) = \sum_{l=1}^{N_M} \tilde{p}_{kl} \bar{V}_{t+1} \left((l, (s_t(2) - b_{i1})^+, ..., (s_t(N_C + 1) - b_{iN_C})^+) \right),$$

where $s = (k, \bar{v}_1, ..., \bar{v}_{N_C})$.

We now describe the basic ADP algorithm adjusted to our game:

Algorithm 6 ADP algorithm for stochastic game

1: Initialize:
   - Choose an initial approximation $\bar{V}_t^0(s_t)$ for each t, s_t.
   - Set $n = 1$ and choose an initial state s_0^1.
2: Choose a sample path w^n which describes the payoff matrices.
3: For $t = 0, ..., N_D - 1$
   - Construct M, such that:

$$m_{ij} = m_{ij}^k + \bar{V}_{t+1}^{n-1}(s_t^i), \quad t < N_D,$$
$$m_{ij} = m_{ij}^k + R_f(s_t^i), \quad t = N_D.$$

   - Solve

$$\hat{v}_t^n = \mathrm{Val}\,(M)\,,$$

     and let π_t^n be the agent's strategy that solves this minimization.
   - Update $\bar{V}_t^n(s_t)$ using (7.3).
   - Compute the next state to visit: w.p. β decide on the next state using π_t^n and with probability $1 - \beta$ choose a random action.
4: Set $n = n + 1$ and go to Step 2.

We now discuss the choice of α and β and introduce aggregation, which can be used to speed up the convergence of the algorithm.

Choice of α The value of the step size α can be chosen in different ways. A review of different step sizes that are used in literature is given in [42]. Two popular step sizes that are often used are the harmonic and the polynomial step size [107]. We use a harmonic step size where α depends on the iteration n:

$$\alpha_n = \max\left\{\frac{a}{a+n-1}, \alpha_0\right\},$$

where the value of α_n decreases in the number of iterations. In Section 7.4, we conducted experiments to decide on the value of a and α_0.

Choice of β In Step 3 of the ADP algorithm, the next state is chosen. If the next state only depends on the strategy π_t^n, it is possible that some states will never be visited and the algorithm does not converge to the best possible value. To avoid this, a random action is chosen with probability $1 - \beta$. In Section 7.4 we also show the results of experiments with the value of β.

Aggregation To have a good approximation of the value function of a state, this specific state has to be visited often enough. During one iteration, the value function of only one state is updated and when the number of states is large, a lot of iterations

are necessary to ensure a good approximation. There are different methods that can be used to speed up the convergence by updating multiple states per iteration. Two methods that are commonly used are aggregation and the use of basic functions [107]. We use aggregation with multiple aggregation levels which is proven to work well for large scale MDPs [43]. An example of an aggregation level is to only consider the requirements and not the payoff matrix.

Let G be the number of aggregation levels and $S^{(g)}$, $g = 0, ..., G$, be the state space corresponding to the g-th aggregation level ($S^{(0)} = S$). The state $s^{(g)}$ is the state corresponding to s in the g-th aggregation level and $\bar{V}^{(g)}(s^{(g)})$ is the value function approximation for this state. The value function approximation of each state s is given by a weighted combination of the value functions of all the corresponding states for the different aggregation levels:

$$\bar{V}^n(s) = \sum_{g=0}^{G} w^{(g,n)}(s)\bar{V}^{(g,n)}(s^{(g)}),$$

where $w^{(g,n)}(s)$ is the weight of the g-th aggregation level for state s. We choose the weight by inverse mean squared errors as described in [43] using the bias and variance of each estimator:

$$w^{(g,n)}(s) \sim \frac{1}{\frac{(\tilde{\sigma}^{(g)}(s))^2}{N_s^{(g)}} + (\tilde{\mu}_s^{(g)})^2},$$

where $\tilde{\sigma}^{(g)}(s))^2$ is the sample variance of all the observations corresponding to the estimate $\bar{V}^{(g)}(s^{(g)})$, $N_s^{(g)}$ is the number of all these observations and $\tilde{\mu}_s^{(g)}$ is the bias from the true value $\bar{V}^{(0)}(s)$. A detailed description of these computations can be found in [43]. Experiments with different aggregation levels can be found in Section 7.4.

In this section we have described a framework to enable us to deal with large scale dynamic games. In the next section, we show computational results to illustrate the performance of the ADP framework.

7.4 Experiments

We have developed a model and solution approach to solve a dynamic variant of security games with restrictions on the agent's strategy set. In this section, we perform experiments to see how our model performs. First, we compare the static and dynamic approach in Section 7.4.1. In Section 7.4.2, we experiment with different input variables of the ADP approach and give computational results. Finally, in Section 7.4.3, we explore the model for an instance of realistic size.

7.4.1 Benefits of the dynamic approach

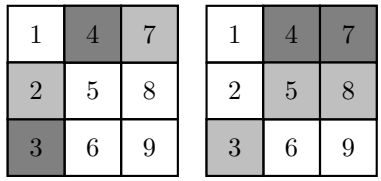
To show the benefits of the dynamic approach studied in this research, we compare it with the static approach as described in Chapter 6.

Consider a game with 9 cells and 8 routes as described in Figure 7.1. In this example, the routes are chosen such that the agent moves right, left or diagonal. The numbers are the cell numbers and the colors of the cells correspond to the intruder's

gain. The darker the color, the higher the intruder's gain: white cells have a payoff of 1, light gray cells have a payoff of 2 and dark gray cells have a payoff of 3. The restrictions are given by v which gives for each cell the minimum number of visits. The transition probabilities for the payoff matrices are:

$$T = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix}.$$

This means that on average both $M^{(1)}$ and $M^{(2)}$ occur with equal probability, so $\mu^{(1)} = \mu^{(2)} = 0.5$. The time period, N_D, equals 80.



Routes	Cells visited by route
1	1, 5, 9
2	2, 3, 6
3	6, 8, 7
4	1, 4, 7
5	1, 2, 3
6	2, 6, 8
7	3, 5, 6
8	4, 5, 9

Figure 7.1: Payoff matrices and routes.

In Table 7.1, the second and third columns show the game value for both the static and dynamic approach for different requirements on the agent's strategy. The value that is given is the expected value per day. The last column gives the running time for the stochastic game. The static game always runs within a second. All experiments in this section are implemented in Matlab version R2016b [85] on an Intel(R) Core(TM) i7 CPU, 2.4GHz, 8 GB of RAM.

Table 7.1: Expected payoff per day for different requirements.

Requirements	Static	Dynamic	
	Game value	Game value	Time (sec)
None	1.45	1.45	1.01
$v = (0,\ 0,\ 40,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0)$	1.64	1.55	10.38
$v = (0,\ 30,\ 0,\ 0,\ 0,\ 0,\ 20,\ 0,\ 0)$	1.88	1.52	143.81
$v = (0,\ 30,\ 40,\ 0,\ 0,\ 0,\ 20,\ 0,\ 0)$	2.31	1.58	6513.91
$v = (0,\ 0,\ 40,\ 0,\ 0,\ 0,\ 30,\ 0,\ 0)$	-	1.85	275.19

Both the static and the dynamic game are played over a time period of N_D days. Note that the strategies from the static game, can always be recreated using the dynamic approach. When there are no restrictions, the dynamic game gives almost the same strategies as the static game because previous actions do not influence the outcome and, when N_D is large enough, the number of times each payoff matrix appears is approximately the expected value μ as used in the static game. As can be seen in Table 7.1, the dynamic game approach gives better results for the agent when there are restrictions. This is because the agent has more flexibility in planning his

strategy. The agent does not have to plan his complete strategy in advance anymore and can adjust his strategy depending on which routes were chosen before.

Moreover, using the stochastic game approach, it is guaranteed that the requirements are met. Another advantage is that we do not have to require that each payoff matrix occurs often enough because we do not need to apply the law of large numbers. Also, some requirements give an infeasible solution for the static approach, while they can be met for the dynamic case. This follows from the fact that for the static approach we use randomized strategies that are the same for each time period. To meet the requirements with high probability, the cells have to be visited more often on average than required. This is not necessary for the dynamic approach, since the strategies can be adjusted to the number of visits in the past. The disadvantage of the stochastic game approach is that the running time increases exponentially in the number of required visits.

7.4.2 Computational results ADP

In this section, we explore the performance of the ADP approach for the dynamic game. Also, we test different input parameters and multiple aggregation levels. Consider the game as described in Section 7.4.1 with the requirements $v = (0,\ 30,\ 40,\ 0,\ 0,\ 0,\ 20,\ 0,\ 0)$. This example will be used to illustrate our experiments.

The running times of the experiments in this section depend on the level of aggregations: the more levels of aggregations, the higher the running time. For the case without aggregation, the running time of the experiments is approximately 1000 seconds. The running time for the case with four levels of aggregation, the running time was approximately 1500 seconds. For all the experiments, we used 3000 as the number of iterations.

First, we test the model without aggregation for different input parameters: the step size parameters, a and α_0, and the probability that a random action is chosen $1 - \beta$. The value function approximation of the initial state is displayed in Figure 7.2 for a selection of different combinations of these parameters without aggregation. The ADP algorithm gives value function approximations for each possible state. Also, for each possible state, a strategy is calculated in Step 3 of the algorithm. We test this strategy by simulating the game after different numbers of iterations. The game is simulated 100 times, where the value approximations and strategies obtained by the ADP algorithm are used.

Tables 7.3-7.6 in Section 7.6 show the results for the model with and without aggregation. Tables 7.3 and 7.5 show the percentage that the requirements are met. In general, it holds that the better the value function approximations are, the higher the probability that the requirements are met. For the dynamic game, it is guaranteed that the conditions are met if this is feasible and the penalty is high enough. However, when using the value function approximations to decide on the strategies, this is not always guaranteed if the approximations are still too far from the optimal values. Tables 7.4 and 7.6 show the average game value for the case that the requirements are met. These tables show that $\beta = 0.75$ gives the best results for all different step values. For the value of the step size, the results are less conclusive. However, higher step sizes give better value function approximation. The choice of α_0 is hereby more important than the choice of a. This can also be seen in Figure 7.2.

In Figure 7.3 the value function approximation for the starting state is shown for

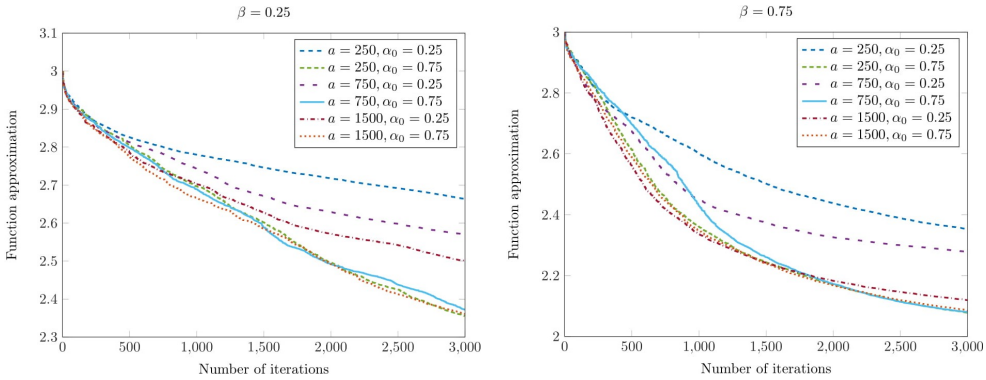


Figure 7.2: Convergence of ADP for different values of α_0 and a and β, no aggregation.

both the ADP with and without aggregation ($a = 750$, $\alpha_0 = 0.75$, $\beta = 0.75$). For the case with aggregation, we use 4 aggregation levels. The first level considers the state without payoff matrix, the second level considers the state with only an even number of visits left, the third level considers the state with the number of visits divided and rounded to the nearest integer above and the fourth level only considers the total number of visits. We use the first level for the case with one level, the first two for the case with two levels, etc.

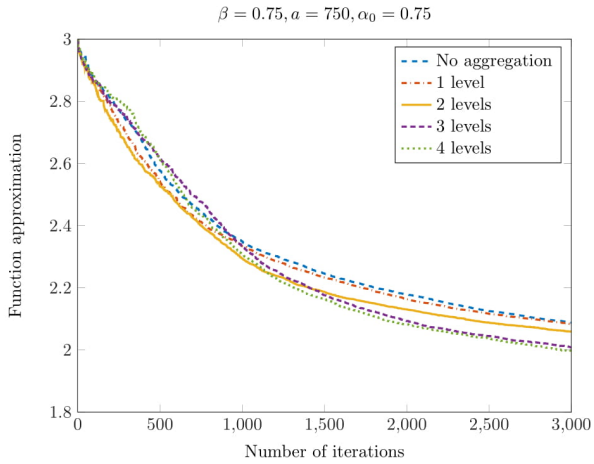


Figure 7.3: Convergence of ADP for different aggregation levels.

Figure 7.3 shows that more aggregation levels give faster convergence. However in this example, the differences between the different aggregation levels are small. This can be explained by the fact that this is a relatively small problem and the convergence for the case without aggregation is already fast. In the next section, we discuss a larger instance where it can be seen that the algorithm with aggregation converges significantly faster than without aggregation.

Recall the results in Section 7.4.1 as shown in Table 7.1. For the game with requirements $v = (0,\ 30,\ 40,\ 0,\ 0,\ 0,\ 20,\ 0,\ 0)$, the game value for the static approach

is 2.31 and for the dynamic approach 1.58. In this section, we approximated the dynamic approach solution by using ADP, where we were able to obtain game values of 1.92 (see Table 7.6). These results show that the expected reward of using the ADP approach is higher than the optimal value of the stochastic game. However, it still outperforms the static approach. Moreover, this method can be used for larger instances where the stochastic game approach is too computationally expensive.

We tested the ADP approach for different instances, which gave similar results. The ADP approach usually outperforms the static approach, but not always in the cases where the game value of the static and dynamic case are close. This can be explained by the fact that in these cases, the requirements do not have a large impact on the optimal strategy, so the static game already gives a solution close to the solution of the dynamic game. Since some approximation error is made in the ADP approach, it might occur that the static game gives a better value. Also, the optimal choice of input parameters vary a bit for different instances, so these have to be chosen carefully depending on the instance. From our computational results, we can say that a high value of β always gives good results and that the value of a needs to be chosen higher for larger instances.

7.4.3 Numerical results for a realistic sized instance

In this section, we give numerical results of a larger instance of the security game for which we cannot solve the stochastic game to optimality. The size of this instance is comparable to real world sized problems. However, we still consider a limited number of payoff matrices such that we can compare the game values with the game value of the static game as described in Chapter 6.

Consider the game as described in Figure 7.4 with two payoff matrices and with requirements on cells 1-5: $v = (10,\ 30,\ 30,\ 30,\ 10,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0)$. The time period, N_D, is 100. The transition probabilities for the payoff matrices are:

$$T = \begin{bmatrix} 0.3 & 0.7 \\ 0.4 & 0.6 \end{bmatrix},$$

which means that on average, the payoff matrix is $M^{(1)}$ with probability 0.36 and $M^{(2)}$ with probability 0.64. Solving the static game gives a game value of 2.57.

We ran the ADP algorithm with 4000 iteration with multiple aggregation levels in different configurations. We used the same aggregation levels as described in Section 7.4.2 with one additional aggregation level. The fifth aggregation level only considers the maximum number of visits over all cells.

The results for a selection of aggregation level configurations are shown in Figure 7.5. This figure shows that aggregation ensures convergence a lot faster in this game. For this instance, the best convergence is obtained with levels 1, 2 and 4 combined. With these levels combined, the problems converge faster than without aggregation levels. This can also be seen in Table 7.2. However, not for all aggregation configurations outperform the algorithm without aggregation. For example, only using levels 3, 4 and 5 decreases the convergence speed. This can be explained by the fact that an error is made when aggregation multiple states. Aggregating many states will speed up the convergence, but may also lead to approximations far from the optimal solution. The right choice of aggregation levels depends on the instance and has to be chosen carefully.

1	6	11	16
2	7	12	17
3	8	13	18
4	9	14	19
5	10	15	20

1	6	11	16
2	7	12	17
3	8	13	18
4	9	14	19
5	10	15	20

Routes	Cells visited by route
1	1, 2, 3, 8, 13, 18
2	5, 10, 14, 15, 19
3	4, 5, 6, 7, 8, 11
4	4, 9, 12, 13, 14, 17
5	16, 17, 18, 19, 20
6	1, 2, 6, 7, 11, 12
7	3, 8, 11, 12, 13
8	4, 5, 10, 15, 20
9	1, 2, 6, 11, 16
10	3, 8, 14, 19, 20
11	6, 7, 12, 16, 17

Figure 7.4: Payoff matrices and routes for realistic sized scenario.

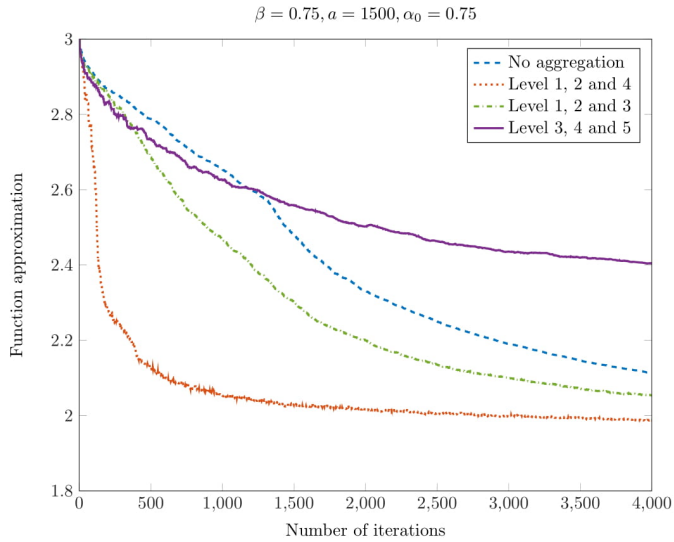


Figure 7.5: Realistic sized scenario with and without aggregation.

Table 7.2: Percentage and average realistic sized scenario ($\beta = 0.75, a = 150, \alpha_0 = 0.75$).

	No aggregation		3 levels	
Iterations	Average	Percentage	Average	Percentage
1000	–	0%	2.47	98%
2000	–	0%	2.33	100%
3000	2.08	83%	2.25	100%
4000	2.03	82%	2.17	100%
5000	2.01	98%	2.07	100%

7.5 Concluding remarks

In this chapter, we have developed a model for the dynamic decision making of an agent when his strategy is restricted by operational requirements. We have formulated the problem as a stochastic game and have shown that the use of a dynamic formulation outperforms the model in which strategies cannot be adjusted to the current situation: better game values for the agent can be obtained. Also, the stochastic game formulation can yield a feasible solution when more operational requirements are considered.

The disadvantage of the stochastic game formulation is that the solution time grows exponentially in the number of cells with requirements. This means that we cannot solve the game to optimality for real world instances. For that reason, we have developed an ADP approach to find approximate solutions. ADP is often used to solve large scale MDPs. With a limited number of adjustments, we have been able to develop a similar approach for our stochastic game.

Experimental results show that the game value which is found by the ADP algorithm is about 25% worse than the optimal solutions. However, using this algorithm, we are able to solve larger instances than when using the full stochastic game. We also compared the ADP approach with a static approach and this showed that the ADP approach outperforms the static approach in our computational experiments. Although experimental results show that the dynamic approach gives better payoffs for the agent's than using a static approach, static strategies can be found faster.

For large instances, the convergence of the value function approximation can be slow, because states have to be visited multiple times before a good approximation can be given. We have used state space aggregation to speed up this convergence. For small instances, we do not gain a lot from this aggregation, because the algorithm without aggregation is already fast. However, for large instances, the speed of convergence is increased considerably with this aggregation.

The convergence of the ADP algorithm also depends on different input parameters which define the step size and the level of randomness. The optimal value of these parameters can vary for each instance and may also depend on the aggregation level. From our computational experiments, we can say that a large step size and a small number of randomness performs the best for our instances.

In this chapter, we have assumed that the evolution of the payoff matrices is defined by a given transition matrix and that at the beginning of each day, the payoff for that day is known. For future research, it would be interesting to investigate the case where not all payoff matrices are knows and only predictions for each day are given.

7.6 Appendix

Table 7.3: Percentage requirements satisfied, no aggregation.

	a	250			750			1500		
β	Iter/α_0	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
	1000	1%	0%	1%	1%	2%	0%	2%	2%	4%
0.25	2000	4%	8%	5%	9%	5%	15%	16%	6%	6%
	3000	13%	8%	7%	17%	7%	5%	30%	14%	4%
	1000	0%	9%	3%	3%	1%	1%	4%	1%	4%
0.5	2000	2%	11%	9%	13%	20%	7%	29%	20%	3%
	3000	6%	20%	26%	18%	11%	68%	47%	18%	4%
	1000	0%	25%	80%	9%	8%	43%	94%	34%	44%
0.75	2000	77%	89%	63%	40%	91%	100%	99%	58%	98%
	3000	88%	100%	96%	99%	100%	99%	100%	99%	100%

Table 7.4: Average game value, no aggregation.

	a	250			750			1500		
β	Iter/α_0	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
	1000	1.73	–	1.76	1.87	2.04	–	1.92	1.84	2.11
0.25	2000	1.84	1.87	1.98	1.97	1.99	1.93	1.97	1.93	1.82
	3000	1.93	1.91	1.92	1.99	1.90	1.86	1.94	1.92	1.85
	1000	–	2.04	2.01	2.09	2.01	1.89	2.05	2.02	2.05
0.5	2000	1.94	1.97	1.93	2.00	1.99	1.98	1.99	2.04	1.95
	3000	1.97	1.98	1.93	2.00	1.99	1.98	1.94	1.97	1.97
	1000	–	2.20	2.17	2.17	2.07	2.22	2.16	2.22	2.16
0.75	2000	2.23	2.13	2.06	2.13	2.14	2.07	2.07	2.15	2.06
	3000	2.16	2.12	2.00	2.13	2.09	2.00	2.00	2.10	2.00

Table 7.5: percentage requirements satisfied, with 3 aggregation levels.

	a	250			750			1500		
β	Iter/α_0	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
	1000	0%	0%	2%	2%	0%	3%	2%	2%	2%
0.25	2000	3%	9%	3%	7%	1%	9%	2%	5%	11%
	3000	18%	11%	9%	20%	9%	20%	10%	20%	21%
	1000	3%	4%	8%	4%	1%	1%	4%	2%	3%
0.5	2000	24%	29%	57%	14%	9%	6%	33%	27%	49%
	3000	48%	60%	88%	23%	27%	12%	62%	31%	72%
	1000	83%	35%	84%	76%	85%	86%	32%	99%	80%
0.75	2000	98%	94%	100%	41%	100%	100%	95%	100%	99%
	3000	100%	99%	99%	100%	100%	100%	100%	100%	100%

Table 7.6: Average game value, with 3 aggregation levels.

	a	250			750			1500		
β	Iter/α_0	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
	1000	–	–	2.06	2.04	–	1.97	1.89	2.06	1.93
0.25	2000	1.85	1.88	1.86	1.98	1.91	1.97	1.93	1.91	1.98
	3000	1.91	1.97	1.90	2.01	1.92	2.01	1.96	1.99	1.92
	1000	2.03	2.19	2.01	2.01	2.13	2.10	2.04	2.12	2.04
0.5	2000	2.03	1.98	2.00	1.99	2.07	1.87	1.97	2.02	1.95
	3000	2.02	1.99	1.93	2.00	1.97	1.91	1.98	2.01	1.93
	1000	2.26	2.21	2.11	2.16	2.18	2.13	2.19	2.12	2.17
0.75	2000	2.16	2.09	1.98	2.06	2.08	1.98	2.09	2.01	2.03
	3000	2.11	2.00	1.92	2.03	2.01	1.92	2.07	1.94	1.99

CHAPTER 8

The price of usability: designing operationalizable strategies for Stackelberg games

The chapter is based on the results in [86].

8.1 Introduction

In this chapter, we consider models that require strategies that are easy to implement. We consider Stackelberg games with a large number of pure strategies for the agent. An optimal mixed strategy typically randomizes over a large number of these strategies. This may result in strategies that are not practical to implement. We propose a framework to construct strategies that are operationalizable by allowing only a limited number of pure strategies in a mixed strategy. However, by restricting the strategy space and allowing only strategies with a small support size, the solution quality might decrease. To investigate the impact of these restrictions, we introduce the price of usability, which measures the ratio between the optimal solution and the internationalization solution.

Several papers discuss the construction of strategies with a small support size in games. Most of these papers are focused on the existence of approximate Nash equilibria and the complexity of finding these. In [35], the authors obtain results for which in general no alpha-approximate equilibria exist when the size of the mixed strategies is smaller than a certain number. In [66], the authors discuss the existence of small support mixed strategies in extensive-form games. The authors use the fact that in extensive-form games, a lot of information that is captured in the mixed strategies will never be used since these nodes will not be reached. They prove that there exist a strategy where the support equals at most the size of the game tree. In [80], the authors prove the existence of epsilon-Nash equilibria for two player games where the support is logarithmic in the number of pure strategies. They prove the existence of epsilon equilibria, using uniform strategies, and give a quasi-polynomial algorithm to find such strategies. In [8], the authors prove NP-hardness results for finding epsilon-approximate Nash equilibria with smaller support than in [80].

For Stackelberg games, only limited research concerning small support strategies is available. However, in many papers, the fact that only a small number of pure strate-

gies are in the support is exploited (for example in column generation algorithms). In [38], the authors consider Stackelberg games and in particular security games. The authors discuss for different games the minimum support size that is required to still obtain optimal strategies. Another paper where Stackelberg games are considered is [102]. Here, the authors construct equilibrium strategies with limited support size and develop algorithms to solve these strategies efficiently.

The research in this chapter is motivated by security games and in particular by threat screening games (TSGs), which tackles the challenge of screening passengers at airports or military control posts. For TSGs, each pure strategy can be viewed as a separate security protocol. These are different configurations and numbers of screening equipment. 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 [57] increasing the likelihood that mistakes are made [98, 26] and making the system vulnerable to exploitation.

The results in this chapter are different from most papers on small support sized strategies in the sense that we restrict ourselves to a maximum number of pure strategies that can be used in the mixed strategy. We discuss what the price of this restriction is, while most papers described above study the minimum number of strategies that is necessary for finding approximate equilibria.

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. For example, the US Coast Guard limited the number of pure strategies used in the Staten Island Ferry security game to avoid cognitive overload for boat operators [33] [Fang private communication 2018]. To the best of our knowledge, [102] 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.

The contributions of this chapter are twofold. First, we develop the concept of operationializable strategies for Stackelberg games and introduce the price of usability as a measure for the cost of the restrictions on the strategy space. We give results for the complexity of finding operationalizable strategies and discuss (computational) bounds on the price of usability. Second, we apply this framework on TSGs and develop algorithms to find operationalizable strategies efficiently for this game. We also extend the TSG game by simultaneously optimizing over the planning of resource teams and the allocation of passengers to resources.

The rest of the chapter is organized as follows. In Section 8.2, we discuss operationalizable strategies in Stackelberg games, introduce the concept of usability and give complexity results. Section 8.3 introduces the SORT-TSG problem. In Section 8.4, we give a heuristic approach for finding operationalizable strategies in general and specifically in SORT-TSG. Finally, in Section 8.5, we give computational bounds on the price of usability in general and discuss results for the SORT-TSGs.

8.2 Usability in Stackelberg games

In this section, we introduce operationalizable strategies for Stackelberg games and give a definition of the price of usability. We first sketch the framework by using

zero-sum Stackelberg games, but these concepts can be applied to general Stackelberg games in a similar way.

8.2.1 Operationalizable strategies in Stackelberg games

Consider a zero-sum Stackelberg game. In a Stackelberg game, the agent first commits to a (randomized) strategy and then an intruder chooses the best response to this strategy. The (finite) set of actions for the agent is A_A and for the intruder A_I. Each action is a pure strategy. The payoff matrix is given by M, such that m_{ij} is the payoff when the agent plays action i and the intruder plays action j. The agent is maximizing this payoff, while the intruder is minimizing. The strategy for the agent is a probability distribution p, where p_i is the probability that the agent plays his i-th pure strategy. The agent is maximizing the payoff by solving the following LP:

$$\begin{aligned} U(p) = \max_{p} \quad & z \\ \text{s.t. } & z \leq \sum_{i \in A_A} p_i m_{ij}, \quad j \in A_I, \\ & \sum_{i \in A_A} p_i = 1, \\ & p_i \geq 0, \qquad\qquad i \in A_A. \end{aligned}$$

In Stackelberg games with a large number of pure strategies, many of these pure strategies can be used in an optimal mixed strategy. To ensure strategies that are easy to implement, we restrict the number of pure strategies that are in the support of any mixed strategy. To this end we introduce the definition of k-operationalizable strategies.

Definition 8.1 (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. $|\{i \in A_A : p_i > 0\}| \leq k$. □

An optimal k-operationalizable strategy for the zero-sum Stackelberg game can be found by solving the following ILP:

$$\begin{aligned} \bar{U}(p) = \max_{p} \quad & z \\ \text{s.t. } & z \leq \sum_{i \in A_A} p_i m_{ij}, \quad j \in A_I, \\ & \sum_{i \in A_A} p_i = 1, \\ & x_i \geq p_i, \qquad\qquad i \in A_A, \\ & \sum_{i \in A_A} x_i \leq k, \\ & p_i \geq 0, \qquad\qquad i \in A_A, \\ & x_i \in \{0, 1\}, \qquad i \in A_A. \end{aligned}$$

However, solving this ILP is computationally unattractive.

Theorem 8.1. *Finding optimal k-operationalizable strategies is NP-hard.*

Proof. We use a reduction from the set covering problem, which is defined as follows. Given a set of elements $\mathcal{U} = \{1, ..., n\}$, a collection of sets $\mathcal{S} = \{S_1, ..., S_m\}$ and an integer k, does there exist a subset $\mathcal{C}$ of $\mathcal{S}$ with size less or equal than k such that the union of all elements from $\mathcal{C}$ equals $\mathcal{U}$?

Given an instance for the set covering problem with $\mathcal{U}$, $\mathcal{S}$ and k, construct an instance of a Stackelberg (security) game that solves this set covering instance in the following way. Let G be a game with a number of targets $\mathcal{U} = \{1, ..., n\}$. The pure strategies for the agent are $\mathcal{S}$ where in the j-th strategy, $i \in \mathcal{U}$ is patrolled by the agent if $i \in S_j$. Each target has value V, so that the agent 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 agent's expected utility is then $E_x[U] = -\max_{i \in \mathcal{U}} V(1 - x_i)$. If there exists a k-operationalizable solution to G with expected utility $E_x[U] > -V$ that means that it is possible for the agent to cover all the targets with some probability using only k pure strategies. These pure strategies in the support of the k-operationalizable strategy then give a set cover of size k (or less). Therefore if the corresponding game has an optimal objective value greater than $-V$ there exists a set cover $\mathcal{C} \subset \mathcal{S}$ such that $|\mathcal{C}| \leq k$. ■

Remark 8.1. We introduced operationalizable strategies for zero-sum stackelberg games. However, for general Stackelberg games, a similar model can be formulated. Consider a general sum Stackelberg game with two players with actions and strategies as defined before. The payoff matrix of the agent is given by M^A and the payoff matrix of the intruder is given by M^I. Both players are maximizing their payoff. The optimal agent's strategy can be found by solving an LP for each intruder's strategy [25]. Similarly, an optimal k-operationalizable strategy Stackelberg game can be found by solving for each $j \in A_I$ the following ILP:

$$
\begin{aligned}
\bar{U}(p) = \max_{p} \quad & \sum_{i \in A_A} p_i m^A_{ij} \\
\text{s.t.} \quad & \sum_{i \in A_A} p_i m^I_{ij} \geq \sum_{i \in A_A} p_i m^I_{ij'}, && j' \in A_I, \\
& \sum_{i \in A_A} p_i = 1, \\
& x_i \geq p_i, && i \in A_A, \\
& \sum_{i \in A_A} x_i \leq k, \\
& p_i \geq 0, && i \in A_A, \\
& x_i \in \{0, 1\}, && i \in A_A.
\end{aligned}
$$

This ILP might be infeasible for some intruder's strategy but must be feasible for some j. Among these, pick the intruders strategy that maximizes the value of the LP. The corresponding p is the optimal strategy for the agent.

8.2.2 Price of usability

When restricting the support size of the mixed strategies and developing strategies that are better usable, the solution quality might decrease. To measure the price of

this restriction, we introduce the price of usability. The price of usability gives a ratio between the payoff in the optimal solution and the payoff with a solution restricted to only k pure strategies.

The price of usability is defined for the agent and we assume that the agent's payoff is always non negative and that the agent's goal is maximizing the payoff. Equivalently, when the agent is minimizing, the payoff is non positive. Note that this can be assumed without loss of generality since every game can be transformed in an equivalent game with non negative payoff, by adding a constant to each element in the payoff matrix. The price of usability is defined as follows.

Definition 8.2 (Price of usability)**.** Let G be a Stackelberg game with optimal mixed strategy solution p for the agent and utility $U(p)$. Let p^k be a 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 $PoU := \frac{U(p)}{U(p^k)}$. □

The higher the price of usability is, the higher the cost of operationalizable strategies are. It is possible to construct examples such that the price of usability is very high. However, in practice, the price of usability is quite small as we will demonstrate in Section 8.5.

8.2.3 High-level algorithmic approach

From Theorem 8.1, it follows that finding k-operationalizable strategies is NP-hard, while the original problem is not. Therefore, an algorithmic approach is used to find k-operationalizable strategies. In this section, we only give a high-level description of this approach.

Our solution approach 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. A similar approach is also used in [102] for general sum Stackelberg game.

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 1 of [80] 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 as long as $k > \frac{12 \log(1+n)}{\epsilon^2}$, where n is the number of pure strategies.

For SORT-TSGs, we use this idea as a starting point, but we need some different techniques that are formalized in Section 8.4. In Section 8.5, we experiment with this algorithmic approach for SORT-TSG and give results on the impact of the price of usability for a general-sum security game and for SORT-TSG.

8.3 Introduction to SORT-TSG

The concept of operationalizable strategies is motivated by threat screening games (TSGs). TSGs are a special variant of security games developed in [20]. We extend this

model by also planning resources simultaneously and introducing k-operationalizable strategies. 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.

8.3.1 Problem Description

A TSG is a game between the screener (agent) and intruder over a finite planning horizon $\mathcal{W}$ consisting of W time windows. The agent is operating a checkpoint through which screenees (passengers) arrive 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 intruder'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 (transportation security administration) pre-check). It is assumed that the number of passengers of category c arriving during each time window, N_c^w, is known.

The intruder attempts to pass through screening by disguising himself 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 intruder 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 agent 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 intruder 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 at the beginning of time window w.

To speed up processing, the agent can increase the number of resources of each type that are available in a particular window (e.g., by 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 agent, 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 screeners are needed. The workforce of the agent consists of S screeners and the agent can decide on the number of screeners available in any window. However, the screeners must follow shifts: they can start in arbitrary time windows but must work for δ consecutive time windows.

8.3.2 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 agent, 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 screeners $b^w \in \mathbb{N}^+$ to start their shift in window w and the number of screeners s^w available in window w. The space of pure strategic strategies can then be expressed as:

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

The first constraint above counts the total number of screeners with shifts currently in progress at time window w. The second constraint stipulates that the total number of screeners assigned to each shift cannot exceed the size of the workforce. The third and fourth constraints enforce that in each time window there must be enough screeners 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 $n_{t,c}^w$. For any choice y of strategic decisions, the space of pure tactical strategies is expressible as:

$$\mathcal{X}_y = \left\{ (n,o) : \begin{array}{ll} \sum_{t\in\mathcal{T}} n_{c,t}^w = N_c^w, & w \in \mathcal{W}, c \in \mathcal{C}, \\ \sum_{t:r\in R(t)} \sum_{c} n_{c,t}^w \le y_r^w C_r - o_r^{w-1} + o_r^w, & w \in \mathcal{W}, r \in \mathcal{R}, \\ n_{t,c}^w, o_r^w \in \mathbb{N}^+, & t \in \mathcal{T}, c \in \mathcal{C}, w \in \mathcal{W}, r \in \mathcal{R}, \end{array} \right\},$$

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 agent 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 agent'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^w_r of passengers waiting to be screened by a resource of type r in time window w are given by:

$$E_p[n^w_{t,c}] = \sum_{i=1}^{S} p_i n^{w,i}_{t,c}, \tag{8.1}$$

$$E_p[o^w_r] = \sum_{i=1}^{S} p_i o^{w,i}_r. \tag{8.2}$$

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

$$\begin{array}{lll} \max_p & \sum_\rho P_\rho \theta_\rho - \sum_w \sum_r \phi_r E_p[o^w_r] & \\ \text{s.t.} & \theta_\rho \le z^w_{c,m} U^+_c + (1 - z^w_{c,m}) U^-_c, & m \in \mathcal{M}, c \in \mathcal{C}, w \in \mathcal{W}, \\ & z^w_{c,m} = \sum_t E_{t,m} \frac{E_p[n^w_{c,t}]}{N^w_c}, & m \in \mathcal{M}, c \in \mathcal{C}, w \in \mathcal{W}, \\ & p \in \mathcal{P}, & \end{array} \tag{$\mathscr{P}$}$$

where $z^w_{c,m}$ is the intruder's detection probability for an intruder 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 $\mathscr{P}$.

8.3.3 Operationalizable Strategies for SORT-TSG

The SORT-TSG problem admits additional usability concerns; not only can the mixed strategy have a very large support, but the number of resource configuration (teams) types 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, hence 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 8.3 ((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 $\sum_{t=1}^{T} l_t \le \tau$. □

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

$$\sum_{t=1}^{T} l_t, \le \tau, \tag{8.3}$$

$$\frac{n^w_{c,t}}{N^w_c} \le l_t, \qquad t \in \mathcal{T}, c \in \mathcal{C}, w \in \mathcal{W}, \tag{8.4}$$

$$l_t \in \{0,1\}, \qquad t \in \mathcal{T}. \tag{8.5}$$

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 (8.1) and (8.2) with:

$$E_p[n^w_{t,c}] = \sum_{i=1}^{k} p_i n^{w,i}_{t,c}, \qquad t \in \mathcal{T}, c \in \mathcal{C}, w \in \mathcal{W}, \tag{8.6}$$

$$E_p[o^w_r] = \sum_{i=1}^{k} p_i o^{w,i}_r, \qquad t \in \mathcal{T}, c \in \mathcal{C}, w \in \mathcal{W}, \tag{8.7}$$

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.,

$$s_i = s_j \qquad i, j \in \{0 \dots k\}. \tag{8.8}$$

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

8.4 Solution approach SORT-TSG

In this section, we develop a heuristic solution approach to find optimal solutions for the SORT-TSG problem. The SORT-TSG problem can be modeled as a mixed integer linear program (MILP). However the resulting operationalizable problem $\mathscr{P}k$ is non-linear, with bilinear terms introduced in (8.6) and (8.7). 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 8.5. In the following, we provide our new solution approach for efficiently solving $\mathscr{P}k$.

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 solution approach is based on the idea described in Section 8.2.3. By fixing $p = \frac{1}{k}$, $\mathscr{P}k$ can be solved directly as an MILP without the creation of extra binary variables. Algorithm 7 outlines this process. To speed up computation, we first solve the full relaxation $\mathscr{P}k^{\,LP}$ 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).

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

Algorithm 7 k-Uniform Strategies

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

we can build more sophisticated and scalable algorithms which we demonstrate in the next section.

While the approach described in Algorithm 7 provides guarantees by [80], in practice the problem can still be slow to solve, as it requires solving an MILP at each step. 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 the used rounding process. 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 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 8 outlines the steps of this solution method. We start by solving the full relaxation $\mathscr{P}k^{\,LP}$ (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 8 Multiple Hierarchical Relaxations

1: **procedure** MHR
2: $\tilde{y}, \tilde{n}, \tilde{o} \leftarrow \mathscr{P}k^{\,LP}$
3: $y_i, \tilde{n}_i, \tilde{o}_i\ \ i = 1, ..., k \leftarrow \text{Strategic}(\mathscr{P}k^{\,LP_n}|q,\ \tilde{y})$
4: $y_i, n_i, o_i\ \ i = 1, ..., k \leftarrow \text{Tactic}(\mathscr{P}k\ |y, \tilde{n})$
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 $\mathscr{P}k^{\ LP_n}$ 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 $\mathscr{P}k^{\ LP_n}$ is still intractable to solve.

Theorem 8.2. *Problem $\mathscr{P}k^{\ LP_n}$ is NP-hard to solve.*

Proof. We use 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 $\mathscr{P}k^{\ LP_n}$. 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 are send to the team with resource $n+1$.

Solving $\mathscr{P}k^{\ LP_n}$ 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 $\mathscr{P}k^{\ LP_n}$ 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, similar to Algorithm 7, we assume a uniform distribution for the mixed strategy. Given $\mathscr{P}k^{\ LP_n}|p$, 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 $\mathscr{P}k^{\ LP_n}|p$ 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 9.

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

Algorithm 9 Determine resource allocations y_s

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

rounding procedure to these variables to obtain integral values n_i. Additionally, 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 (8.6) and (8.7) in $\mathscr{P}k$. 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 (8.6) and (8.7) as:

$$
\begin{aligned}
&E[n^w_{t,c}] = \sum_{i=1}^{k} \left(p_i \left\lfloor \tilde{n}^{w,i}_{t,c} \right\rfloor + z^{w,i}_{t,c} \right), \\
&0 \le z^{w,i}_{t,c} \le p_i, && i = 1, ..., k. \\
&z^{w,i}_{t,c} \ge p_i - (1 - \gamma^{w,i}_{t,c}), && i = 1, ..., k.
\end{aligned}
$$

This subroutine is outlined in Algorithm 10. 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 10 Determing Passenger type allocation n_s

1: **procedure** TACTIC($\mathscr{P}k\ |(y, b),\, \tilde{n}$)
2: $n \leftarrow n_i = \lfloor \tilde{n}_i \rfloor + \delta_{n_i},\ \delta_{n_i} \in \mathbb{N}^+, i = 1, ...k$
3: LinearizeTerms($\mathscr{P}k\ |(y, b)$)
4: **return** $\underset{n,p}{\operatorname{argmax}}\ \mathscr{P}k\ |(y, b)$

8.5 Evaluation

In the previous, we introduced the concept of operationalizable strategies and the price of usability. For the SORT-TSG problem, we have developed an algorithmic to find operationalizable strategies efficiently. In this section, we investigate the impact of operationalizable strategies. First, we consider a more general security game. Thereafter, we evaluate our algorithms for the SORT-TSG problem en run experiments to determine the price of uasbility.

8.5.1 Illustrative examples for price of usability in Stackelberg security game

First we give an example to show that the price of usability is unbounded.

Example 8.1 (PoU can be any number)**.** Consider a zero-sum security game with n targets. The agent decides which target to patrol and the intruders attacks one target. If the intruder attacks a target successfully, the gain of the agent equals 1. Moreover, if the intruder attacks a target that is patrolled by the agent, the gain for the agent is cn, with c a constant. A pure strategy for the agent is to patrol a single target and a pure strategy for the intruder is to attack a single target. Without restrictions on the number of pure strategies, the agent's optimal strategy is to patrol each target with probability $\frac{1}{n}$. The intruder's strategy can then be any strategy and the expected payoff will be $\frac{n-1}{n} + c \approx c + 1$ (when n is large). However, when only $k < n$ pure strategies are allowed, there is always at least one target that will never be patrolled by the agent. The intruder will choose such a target to attack and the expected payoff will be 1, resulting in a PoA larger than c. □

From this example, it follows that the price of usability can not be bounded. However, in many applications, the price of usability is low, which we will illustrate by means of an example security general sum game. In the next section, similar experiments are conducted for SORT-TSG.

To illustrate operationalizable strategies and how they look like, we consider the robbery game which is described in [102]. This is a Stackelberg security game between an agent an a robber, where the robber is attacking a specific house and the agent is able to patrol a set of houses during one night. Each house has a specific value for both the agent and the robber. Catching the robber results in a reward for the agent and in a cost for the robber. The agent visits multiple houses during one night and the earlier he arrives at a specific house, the higher the probability of succesfully catching the robber. There might be multiple robbery types, which all have different values for each house. The agent only has a probability distribution over each type. In the following example, we show the impact of operationalizable strategies.

Example 8.2. Consider a robbery game with 1 robber type, 5 houses and an agent that is a able to patrol 2 houses during the night. For the robber, the houses have a value of 4, 5, 6, 7, and 8 for a successful robbery. Getting caught by the agent results in a gain of 0 independent of the house. If the robber is at the first house that is visited by the agent, he will always catch the robber. If the robber attacks the second house during his patrol, the robber is caught with probability 0.5. The agent receives a gain of 3 if the robber is caught and 0 if the intruders is not caught. In an optimal strategy, the agent randomizes over 4 pure strategies and the expected payoff for the agent equals 2.24. However, when we require k-operationalizable strategies with $k = 1, 2, 3$, the game value equals 0, 1.50 and 1.91, resulting in a price of usibility of ∞, 1.49 and 1.17 respectively. □

As this example shows, the price of usability can be very high for small k, but quickly decreases when k increases. The same trend can be shown for larger instances of the robbery game. We investigate the impact of operationalizable strategies on random instances of the robbery game. In Figure 8.1, for different k the average price of usability for random instances of the robbery game is shown. We have picked

20 random instances with varying number of types, houses and visits, such that the number of pure agent strategies is varying between 1000 and 10000. These results show that the price of usability is small for these types of games, even when the number of pure strategies is small.

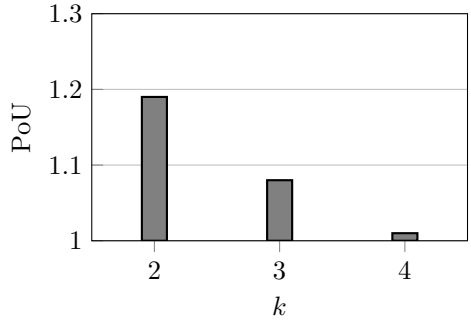


Figure 8.1: PoU for random instances of the robbery game

8.5.2 Results on SORT-TSG

In this section, we consider the SORT-TSG problem. 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. Each experiment is averaged over 20 randomized instances of the remaining parameters.

The Price of Usability In this chapter, we proposed to mitigate the *price of usability (PoU)* by computing (k, τ)-operationalizable strategies. We define 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 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 8.2, 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 [61].

Solution quality We evaluate the solution quality of our algorithms by comparing to (1) the full MIP which optimally solves operationalizable security game $\mathscr{P}k$ and (2) a column generation heuristic, which cuts off after I iterations. Figure 8.3(a) shows the comparison of our methods with the column generation (CG) baseline. When run to convergence, CG optimally solves $\mathscr{P}$, without operationalizable constraints. 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.

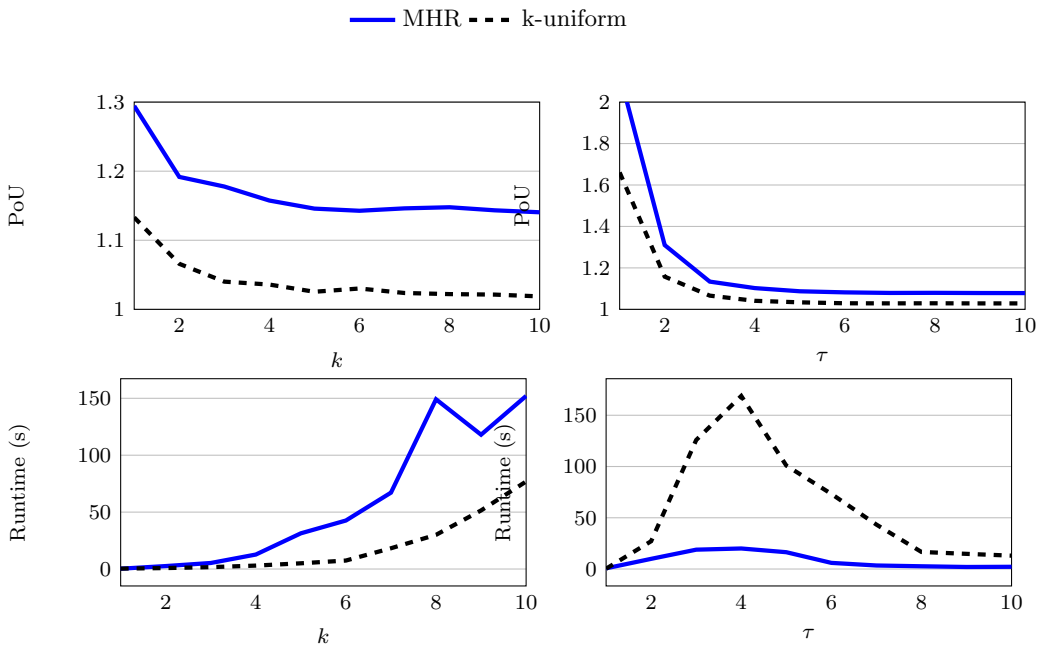


Figure 8.2: Here we show the empirical PoU, as well as the runtimes of both methods with increasing k and τ for both methods (left: $\tau = 10$, right: $k = 2$).

Additionally, we investigate the support size of the mixed strategies computed by CG without operationalizable constraints. Figure 8.3(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 columns as well as after full convergence. The results are shown in Figure 8.4. 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.

In Table 8.1, we compare utility of our algorithms with the utility obtained from solving the full MILP (which optimally solves $\mathscr{P}k$). The full MILP can only be solved for small instances, with a maximum of $k = 3$. For these instances, we see that both methods produce near-optimal solutions and can be executed significantly faster. For moderate and large sized instances, Table 8.1 shows that the k-uniform algorithm outperforms the MHR heuristic, but the MHR heuristic can solve large instances faster.

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 8.5 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.

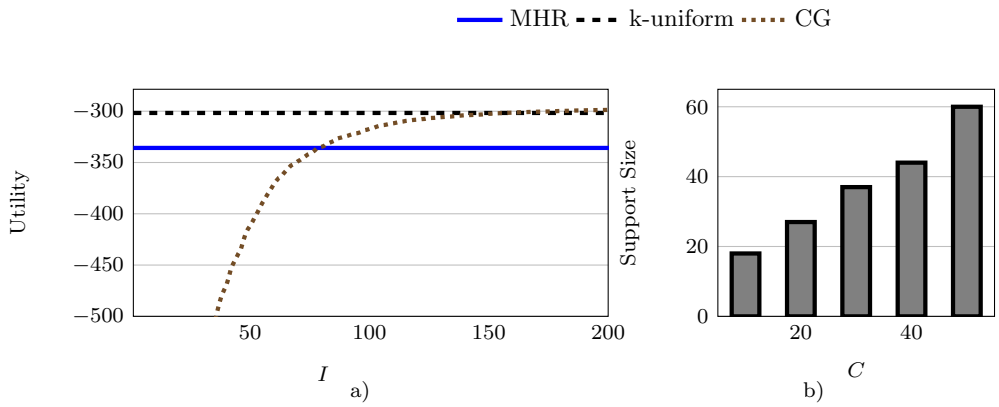


Figure 8.3: a) Comparison of our algorithms with CG which is cut off after I iterations ($k = 5$, $\tau = 10$). b) Support size of CG solutions for increasing problem size.

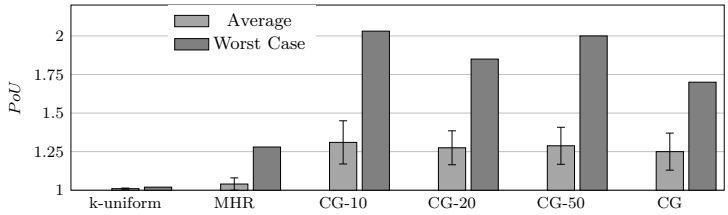


Figure 8.4: Average case price of usability and b) worst case price of usability, for our two methods (k-uniform and MHR) compared to a cutoff column generation baseline. Column generation (CG) was cutoff after 10, 20 and 50 columns and after convergence.

Table 8.1: Runtime and utility u^* of the k-uniform and MHR algorithm compared with the solution of the full MIP (small: $k = 3$, moderate,large: $k = 4$).

	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	-	-	-	-

8.6 Concluding remarks

In this chapter, we have addressed the problem of usability in Stackelberg (security) games and introduced the new problem of operationalizable strategies. We have developed a framework to find mixed strategies using only a small number of pure strategies. To measure the cost of these operationalizable strategies, we defined the price of usability. Although the price of usability can be very high, we have showed for security game example that the price is low, even when only 3 or 4 pure strategies are used.

This research was motivated by threat screening games. For these games, we have developed a single framework which reasons about the three levels of planning:

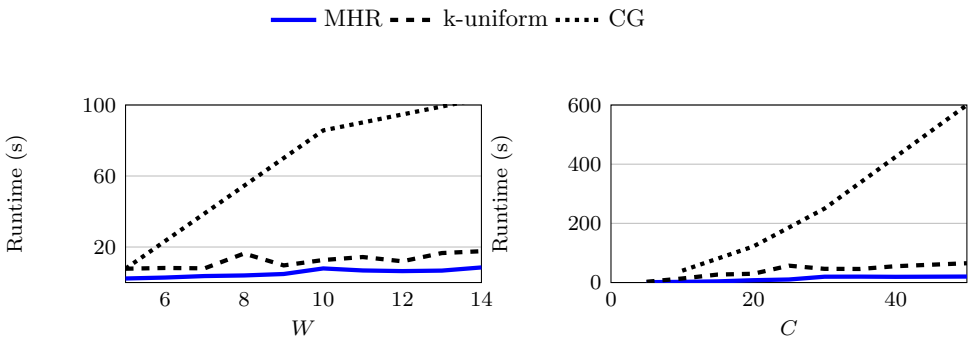


Figure 8.5: Runtime for different values of W and C ($k = 2$, $\tau = 5$, left: $C = 10$, right: $W = 5$)

strategic, operational and tactical level decision problems. Motivated by the important problem of screening for threats, we provided algorithmic solutions to overcome the computational challenges that arise when these planning problems are addressed for TSGs and which mitigate the price of usability.

CHAPTER 9

Conclusion and outlook

In this thesis, we have developed several mathematical models for the optimal deployment of security forces to protect large areas (e.g., land, water, air), infrastructures and high value assets. One of the main challenges we have tackled, is the adaptive behavior of an intelligent intruder and the uncertainty that arises in various model parameters. To this end, we have extended several game theoretic models by incorporating stochastic modeling. We have showed that several modeling techniques, such as queueing theory, approximate dynamic programming and Bayesian belief functions can be used to model the random environment.

Taking into account the stochastic environment and new information that becomes available during the game yields models that are closer to reality and will therefore generate strategies of higher quality. However, it also increases the complexity of the problem and might result in problems that cannot be solved to optimality for realistic sized instances. In this thesis, we have focused on designing algorithms to decrease the computation time of finding optimal strategies, for example by using different game representations, implementing approximate dynamic programming for games or exploiting the underlying structure of the problem. This enables us to solve larger instances. Though, even with these techniques, we are limited in the size of the instances that can be solved.

The last part of this thesis considers models in which the strategy space of the agent is restricted, for example by governmental guidelines or by the intention to construct strategies that are easy to implement. While these restrictions might occur in various situations, there is limited literature available about how to take this into account. By not taking into account these restrictions explicitly in the modeling, non-compliant strategies are generated that need to adjusted in an ad-hoc manner, causing sub-optimal solutions. By incorporating these restrictions in our model explicitly, optimal strategies given the restrictions are found. However, also the additional restrictions on the agent's strategy increase the complexity of the problems significantly, which again limits the size of the instances that can be solved to optimality.

There are several possible research directions that can be explored to overcome the problem of being unable to solve the proposed models to optimality. One possibility is to develop new methods or improve current methods and speed up the (approximation) algorithms even more. However, the question is whether this is sufficient. Nowadays, more and more information becomes available, increasing the complexity of the problems. The suggested approaches might not be enough to solve these

problems to optimality.

Another approach that should be explored further is whether or not all available information should be taken into account. When optimizing certain security problems, it might be attractive to take as much information into account as possible. However, it can occur that some information does not help us to get better strategies and only increases the complexity of the problem. For example, when patrolling a certain area, it might not be useful to consider all information about areas that cannot be reached in the next time step anyway. Therefore, it would be useful to study the need for information and investigate which information contributes to a higher solution quality. In some cases, it will be necessary to make a trade-off between high solution quality and low computation time.

Most of the models developed in this thesis, make use of known probability distributions as input. However, this input is not always known in practice. In the past few year, more data has become available and is collected automatically. Therefore, it would be interesting to investigate the combination of the models from this thesis with the latest techniques in data science to better exploit the value of using extra data and determine the input variables.

Bibliography

[1] H. Ackermann, H. Röglin, and B. Vöcking. Pure Nash equilibria in player-specific and weighted congestion games. volume 410, pages 1552–1563. Citeseer, 2009.

[2] S. Alpern and T. Lidbetter. Approximate solutions for expanding search games on general networks. *Annals of Operations Research*, 2018.

[3] S. Alpern, T. Lidbetter, and K. Papadaki. Optimizing periodic patrols against short attacks on the line and other networks. *European Journal of Operational Research*, 2018.

[4] S. Alpern, A. Morton, and K. Papadaki. Patrolling games. *Operations Research*, 59(5):1246–1257, 2011.

[5] S. Alscher. Knocking at the doors of fortress europe: Migration and border control in southern spain and eastern poland. 2017.

[6] E. Altman, T. Boulogne, R. El-Azouzi, T. Jiménez, and L. Wynter. A survey on networking games in telecommunications. *Computers & Operations Research*, 33(2):286–311, 2006.

[7] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. *SIAM Journal on Computing*, 38(4):1602–1623, 2008.

[8] P. Austrin, M. Braverman, and E. Chlamtáč. Inapproximability of np-complete variants of Nash equilibrium. volume 9, pages 117–142. Theory of Computing Exchange, 2013.

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

[10] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

[11] N. Basilico, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of The*

8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, pages 57–64. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[12] A. Basu and L. Stettner. Finite- and infinite-horizon shapley games with non-symmetric partial observation. *SIAM Journal on Control and Optimization*, 53(6):3584–3619, 2015.

[13] M.S. Bazaraa, H.D. Sherali, and C.M. Shetty. *Nonlinear programming: theory and algorithms*. John Wiley & Sons, Inc, second edition, 1993.

[14] M.G.H. Bell, U. Kanturska, J.-D. Schmöcker, and A. Fonzone. Attacker–defender models and road network vulnerability. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 366(1872):1893–1906, 2008.

[15] D.S. Bernstein, R. Givan, N Immerman, and S Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

[16] V.S. Borkar. N-person noncooperative stochastic games with partial information. *Lecture notes in economics and mathematical systems*, 1(389):98–98, 1992.

[17] B. Bosansky, C. Kiekintveld, V. Lisy, and M. Pechoucek. An exact double-oracle algorithm for zero-sum extensive-form games with imperfect information. *Journal of Artificial Intelligence Research*, 51:829–866, 2014.

[18] G. Brown, M. Carlyle, J. Salmerón, and K. Wood. Defending critical infrastructure. *Interfaces*, 36(6):530–544, 2006.

[19] G. Brown, J. Kline, A. Thomas, A. Washburn, and K. Wood. A game-theoretic model for defense of an oceanic bastion against submarines. *Military Operations Research*, pages 25–40, 2011.

[20] M. Brown, A. Sinha, A. Schlenker, and M. Tambe. One size does not fit all: A game-theoretic approach for dynamically and effectively screening for threats. In *AAAI conference on Artificial Intelligence*, pages 425–431, 2016.

[21] L. Buşoniu, B. De Schutter, and R. Babuška. Approximate dynamic programming and reinforcement learning. In *Interactive collaborative information systems*, pages 3–44. Springer, 2010.

[22] A. Charnes. Constrained games and linear programming. *Proceedings of the National Academy of Sciences*, 39(7):639–641, 1953.

[23] H.L. Chen and T. Roughgarden. Network design with weighted players. *Theory of Computing Systems*, 45(2):302, 2009.

[24] J.E. Cohen and F.P. Kelly. A paradox of congestion in a queuing network. *Journal of Applied Probability*, 27(3):730–734, 1990.

[25] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 82–90. ACM, 2006.

[26] G. Cooper and J Sweller. Effects of schema acquisition and rule automation on mathematical problem-solving transfer. *Journal of educational psychology*, 79(4):347, 1987.

[27] K.J. Cormican, D.P. Morton, and R.K. Wood. Stochastic network interdiction. *Operations Research*, 46(2):184–197, 1998.

[28] Inc. CVX Research. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, December 2016.

[29] J.P. Dickerson, G.I. Simari, V.S. Subrahmanian, and S. Kraus. A graph-theoretic approach to protect static and moving targets from adversaries. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 299–306. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[30] E.B. Dynkin and A.A. Yushkevich. *Controlled Markov processes*. Springer-Verlag New York, first edition, 1978.

[31] J.N. Eagle. The optimal search for a moving target when the search path is constrained. *Operations research*, 32(5):1107–1115, 1984.

[32] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 136–143. IEEE, 2004.

[33] F. Fang, A.X. Jiang, and M. Tambe. 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, 2013.

[34] F. Fang, P. Stone, and M. Tambe. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 2589–2595, 2015.

[35] T. Feder, H. Nazerzadeh, and A. Saberi. Approximating Nash equilibria using small-support strategies. In *Proceedings of the 8th ACM conference on Electronic commerce*, pages 352–354. ACM, 2007.

[36] D.R. Fulkerson and G.C. Harding. Maximizing the minimum source-sink path subject to a budget constraint. *Mathematical Programming*, 13(1):116–118, 1977.

[37] S. Gal. Search games with mobile and immobile hider. *SIAM Journal on Control and Optimization*, 17(1):99–122, 1979.

[38] J. Gan and B. An. Minimum support size of the defenders strong Stackelberg equilibrium strategies in security games. In *Proc. AAAI Spring Symp. on Appl. Computat. Game Theory*, 2014.

[39] N. Gatti. Game theoretical insights in strategic patrolling: Model and algorithm in normal-form. In *Proceedings on the European Conference on Artificial Intelligence*, pages 403–407, 2008.

[40] E. Gelenbe. Product-form queueing networks with negative and positive customers. *Journal of Applied Probability*, 28(3):656–663, 1991.

[41] E. Gelenbe, P. Glynn, and K. Sigman. Queues with negative arrivals. *Journal of Applied Probability*, 28(1):245–250, 1991.

[42] A.P. George and W.B. Powell. Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Machine learning*, 65(1):167–198, 2006.

[43] A.P. George, W.B. Powell, and S.R. Kulkarni. Value function approximation using multiple aggregation for multiattribute resource management. *Journal of Machine Learning Research*, 9(Oct):2079–2111, 2008.

[44] M.K. Ghosh and A. Goswami. Partially observable semi-Markov games with discounted payoff. *Stochastic analysis and applications*, 24(5):1035–1059, 2006.

[45] B. Golany, N. Goldberg, and U.G. Rothblum. A two-resource allocation algorithm with an application to large-scale zero-sum defensive games. *Computers & Operations Research*, 78:218–229, 2017.

[46] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008.

[47] M. Grant, S. Boyd, and Y. Ye. Disciplined convex programming. In *Global optimization*, pages 155–210. Springer, 2006.

[48] E. Halvorson, V. Conitzer, and R. Parr. Multi-step multi-sensor hider-seeker games. In *Proceedings of International Joint Conference on Artificial Intelligence*, volume 9, pages 159–166, 2009.

[49] E.A. Hansen, D.S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715, 2004.

[50] T. Harks and M. Klimm. On the existence of pure Nash equilibria in weighted congestion games. *Mathematics of Operations Research*, 37(3):419–436, 2012.

[51] J.C. Harsanyi. Games with incomplete information played by Bayesian players, I-III: part I. the basic model. *Management science*, 14:159–182, 1967.

[52] W. Haskell, D. Kar, F. Fang, M. Tambe, S. Cheung, and E. Denicola. Robust protection of fisheries with compass. In *Twenty-Sixth IAAI Conference*, 2014.

[53] R. Hassin. *Rational queueing*. Chapman and Hall/CRC, 2016.

[54] J.P. Hespanha and M. Prandini. Nash equilibria in partial-information games on Markov chains. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 3, pages 2102–2107. IEEE, 2001.

[55] J.P. Hespanha, M. Prandini, and S. Sastry. Probabilistic pursuit-evasion games: A one-step Nash approach. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 3, pages 2272–2277. IEEE, 2000.

[56] P. Hew and N. Yiap. Optimally randomized patrolling of chokepoints for theatre antisubmarine warfare. *Military Operations Research*, 23(1):49–56, 2018.

[57] N.M. Hogg. Measuring cognitive load. In *Handbook of Research on Electronic Surveys and Measurements*, pages 188–194. Idea Group Inc, 2007.

[58] R. Hohzaki. Search games: Literature and survey. *Journal of the Operations Research Society of Japan*, 59(1):1–34, 2016.

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

[60] M. Jain, D. Korzhyk, O. Vank, V. Conitzer, M. Pchouek, and M. Tambe. A double oracle algorithm for zero-sum security games on graphs. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, pages 327–334, 2011.

[61] M. Jain, K. Leyton-Brown, and M. Tambe. The deployment-to-saturation ratio in security games. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 1362–1370. AAAI Press, 2012.

[62] E. Jenelius, J. Westin, and Å.J. Holmgren. Critical infrastructure protection under imperfect attacker perception. *International Journal of Critical Infrastructure Protection*, 3(1):16–26, 2010.

[63] E. Karmon. Central asian jihadists in the front line. *Perspectives on Terrorism*, 11(4):78–86, 2017.

[64] D. Keus, F.P.A. Benders, H.J. Fitski, and H.J. Grootendorst. Multi-platform operations in the underwater warfare testbed (uwt). 2009.

[65] W. Knippenberg. *Planning Anti-Submarine Warfare using Partially Observable Markov Games*. MSc. Thesis Maastricht University, 2016.

[66] D. Koller and N. Megiddo. Finding mixed strategies with small supports in extensive form games. *International Journal of Game Theory*, 25(1):73–92, 1996.

[67] D. Koller, N. Megiddo, and B. Von Stengel. Efficient computation of equilibria for extensive two-person games. *Games and economic behavior*, 14(2):247–259, 1996.

[68] K. Kollias and T. Roughgarden. Restoring pure equilibria to weighted congestion games. In *International Colloquium on Automata, Languages, and Programming*, pages 539–551. Springer, 2011.

[69] D. Korzhyk, V. Conitzer, and R. Parr. 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, 2010.

[70] C.M. Laan, A.I. Barros, R.J. Boucherie, and H. Monsuur. Security games with probabilistic constraints on the agents strategy. In *International Conference on Decision and Game Theory for Security*, pages 481–493. Springer, 2017.

[71] C.M. Laan, A.I. Barros, R.J. Boucherie, and H. Monsuur. Security games with restricted strategies: An approximate dynamic approach. *NL ARMS Netherlands Annual Review of Military Studies 2018: Coastal Border Control: From Data and Tasks to Deployment and Law Enforcement*, page 171, 2018.

[72] C.M. Laan, A.I. Barros, R.J. Boucherie, H. Monsuur, and W. Noordkamp. Optimal deployment for anti-submarine warfare operations with time dependent strategies. *submitted to Journal of Defense Modeling and Simulation.*

[73] C.M. Laan, A.I. Barros, R.J. Boucherie, H. Monsuur, and J. Timmer. Solving partially observable agent-intruder games with an application to border security problems. *submitted to Naval Research Logistics.*

[74] C.M. Laan, J. Timmer, and R.J. Boucherie. Non-cooperative queueing games on a jackson network. *submitted to Annals of Operations Research.*

[75] C.M. Laan, T. van der Mijden, A.I. Barros, R.J. Boucherie, and H. Monsuur. An interdiction game on a queueing network with multiple intruders. *European journal of operational research*, 260(3):1069–1080, 2017.

[76] C. Lim and J.C. Smith. Algorithms for discrete and continuous multicommodity flow network interdiction problems. *IIE Transactions*, 39(1):15–26, 2007.

[77] K.Y. Lin, M.P. Atkinson, T.H. Chung, and K.D. Glazebrook. A graph patrol problem with random attack times. *Operations Research*, 61(3):694–710, 2013.

[78] K.Y. Lin, M.P. Atkinson, and K.D. Glazebrook. Optimal patrol to uncover threats in time when detection is imperfect. *Naval Research Logistics*, 61(8):557–576, 2014.

[79] X. Lin, P.A. Beling, and R. Cogill. Multi-agent inverse reinforcement learning for zero-sum games. *IEEE Transactions on Games*, 10(1):56–68, 2017.

[80] R.J. Lipton, E. Markakis, and A. Mehta. Playing large games using simple strategies. In *Proceedings of the 4th ACM conference on Electronic commerce*, pages 36–41. ACM, 2003.

[81] D. Liu, X.F. Wang, and J. Camp. Game-theoretic modeling and analysis of insider threats. *International Journal of Critical Infrastructure Protection*, 1:75–80, 2008.

[82] Y. Liu, C. Comaniciu, and H. Man. A Bayesian game approach for intrusion detection in wireless ad hoc networks. In *Proceeding from the 2006 workshop on Game theory for communications and networks*, page 4. ACM, 2006.

[83] L. MacDermed, C. Isbell, and L. Weiss. Markov games of incomplete information for multi-agent reinforcement learning. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[84] MATLAB. version 8.4 (R2014b). The MathWorks Inc., Natick, Massachusetts, 2014.

[85] MATLAB. version 9.1 (R2016b). The MathWorks Inc., Natick, Massachusetts, 2016.

[86] S.M. McCarthy, C.M. Laan, K. Wang, P. Vayanos, A. Sinha, and M. Tambe. The price of usability: Designing operationalizable strategies for security games. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 454–460, 2018.

[87] H.B. McMahan, G.J. Gordon, and A. Blum. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the 20th International Conference on Machine Learning*, pages 536–543, 2003.

[88] F. Meng and J. Zhan. Two methods for solving constrained bi-matrix games. *Open Cybernetics & Systemics Journal*, 8:1038–1041, 2014.

[89] M.R.K Mes and A.P. Rivera. Approximate dynamic programming by practical examples. In R.J. Boucherie & N.M. van Dijk, editor, *Markov Decision Processes in Practice*, pages 63–101. Springer, 2017.

[90] I. Milchtaich. Congestion games with player-specific payoff functions. *Games and economic behavior*, 13(1):111–124, 1996.

[91] M. Mishra, W. An, D. Sidoti, X. Han, D.F.M. Ayala, J.A. Hansen, K.R. Pattipati, and D.L. Kleinman. Context-aware decision support for anti-submarine warfare mission planning within a dynamic environment. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, (99):1–18, 2017.

[92] H. Monsuur, R.H.P. Janssen, and H.G. Jutte. A game-theoretic attacker-defender model for a sea base: Optimal deployment at the maritime battleground. In P.J. Oonincx & A.J. van der Wal, editor, *NL ARMS - Optimal Deployment of Military Systems: Technologies for Military Missions in the Next Decade*. Asser Press, 2014.

[93] D.P. Morton, F. Pan, and K.J. Saeger. Models for nuclear smuggling interdiction. *IIE Transactions*, 39(1):3–14, 2007.

[94] J. Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

[95] M.F. Neuts. A multistage search game. *Journal of the Society for Industrial & Applied Mathematics*, 11(2):502–507, 1963.

[96] G. Owen. *Game theory*. Academic Press, second edition, 1982.

[97] R. Oyanedel, A. Keim, J.C. Castilla, and S. Gelcich. Illegal fishing and territorial user rights in chile. *Conservation Biology*, 32(3):619–627, 2018.

[98] F.G. Paas. Training strategies for attaining transfer of problem-solving skill in statistics: A cognitive-load approach. *Journal of educational psychology*, 84(4):429, 1992.

[99] F. Pan, W.S. Charlton, and D.P. Morton. A stochastic program for interdicting smuggled nuclear material. In *Network interdiction and stochastic integer programming*, pages 1–19. Springer, 2003.

[100] J.S. Pang, G. Scutari, D.P. Palomar, and F. Facchinei. Design of cognitive radio systems under temperature-interference constraints: A variational inequality approach. *IEEE Transactions on Signal Processing*, 58(6):3251–3271, 2010.

[101] P. Paruchuri, J.P. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S. Kraus. Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 895–902. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[102] P. Paruchuri, J.P. Pearce, M. Tambe, F. Ordonez, and S. Kraus. An efficient heuristic approach for security against multiple adversaries. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 181. ACM, 2007.

[103] J. Perolat, B. Scherrer, B. Piot, and O. Pietquin. Approximate dynamic programming for two-player zero-sum Markov games. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1321–1329, 2015.

[104] H. Peters. *Game theory: A multi-leveled approach*. Springer, first edition, 2008.

[105] J.P. Ponssard and S. Sorin. The lp formulation of finite zero-sum games with incomplete information. *International Journal of Game Theory*, 9:99–105, 1980.

[106] J.R. Poulsen, S.E. Koerner, S. Moore, V.P. Medjibe, S. Blake, C.J. Clark, M.E. Akou, M. Fay, A. Meier, J. Okouyi, et al. Poaching empties critical central african wilderness of forest elephants. *Current Biology*, 27(4):R134–R135, 2017.

[107] W.B. Powell. Approximate dynamic programmingii: Algorithms. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[108] M.L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[109] E. Regnier and D. Singham. Targeting an asymmetric maritime threat: Workshop report. Technical report, Monterey, California. Naval Postgraduate School, 2013.

[110] J.B. Rosen. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica: Journal of the Econometric Society*, pages 520–534, 1965.

[111] R.W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973.

[112] S.M. Ross. *Stochastic processes*. John Wiley & Sons, Inc, second edition, 1996.

[113] J. Salmeron, K. Wood, and R. Baldick. Analysis of electric grid security under terrorist threat. *IEEE Transactions on Power Systems*, 19(2):905–912, 2004.

[114] G. Scutari, D.P. Palomar, F. Facchinei, and J.S. Pang. Convex optimization, game theory, and variational inequality theory. *IEEE Signal Processing Magazine*, 27(3):35–49, 2010.

[115] J. Semple. Constrained games for evaluating organizational performance. *European Journal of Operational Research*, 96(1):103–112, 1997.

[116] A. Smith. Comment: Containerization of contraband: Battling drug smuggling at the fourth busiest container handling facility in the united states. *Georgia Journal of International & Comparative Law*, 45(2):299, 2017.

[117] M.T.J. Spaan. *Partially observable Markov decision processes*, pages 387–414. Springer, 2012.

[118] M. Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.

[119] A.J. Thomas. *Tri-level optimization for anti-submarine warfare mission planning*. PhD thesis, Monterey, California. Naval Postgraduate School, 2008.

[120] L.C. Thomas and A.R. Washburn. Dynamic search games. *Operations Research*, 39(3):415–422, 1991.

[121] J. Timmer and W. Scheinhardt. Cost sharing of cooperating queues in a Jackson network. *Queueing systems*, 75(1):1–17, 2013.

[122] J. Timmer and W. Scheinhardt. Customer and cost sharing in a Jackson network. *International Game Theory Review*, 2018.

[123] L. Tran-Thanh, M. Polukarov, A. Chapman, A. Rogers, and N.R. Jennings. On the existence of pure strategy Nash equilibria in integer–splittable weighted congestion games. In *International Symposium on Algorithmic Game Theory*, pages 236–253. Springer, 2011.

[124] B. Von Stengel. Efficient computation of behavior strategies. *Games and Economic Behavior*, 14(2):220–246, 1996.

[125] A. Washburn and L.T.C. Lee Ewing. Allocation of clearance assets in ied warfare. *Naval Research Logistics*, 58(3):180–187, 2011.

[126] A. Washburn and K. Wood. Two-person zero-sum games for network interdiction. *Operations Research*, 43(2):243–251, 1995.

[127] L.M. Wein and M.P. Atkinson. The last line of defense: designing radiation detection-interdiction systems to protect cities from a nuclear terrorist attack. *Nuclear Science, IEEE Transactions on*, 54(3):654–669, 2007.

[128] W.L. Winston. *Operation research, applications and algorithms*. Brooks/Cole, fourth edition, 2004.

[129] R. Wollmer. Removing arcs from a network. *Operations Research*, 12(6):934–940, 1964.

[130] K.R. Wood. Bilevel network interdiction models: Formulations and solutions. *Wiley Encyclopedia of Operations Research and Management Science*, 2011.

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

[132] Y. Yin, H. Xu, J. Gain, B. An, and A.X. Jiang. 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, 2015.

[133] Z. Yin, D. Korzhyk, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. Nash in security games: Interchangeability, equivalence, and uniqueness. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 1139–1146. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

Summary

In this thesis, we develop mathematical models for the optimal deployment of security forces addressing two main challenges: adaptive behavior of the adversary and uncertainty in the model. We address several security applications and model them as agent-intruder games. The agent represents the security forces which can be the coast guard, airport control, or military assets, while the intruder represents the agent's adversary such as illegal fishermen, terrorists or enemy submarines.

To determine the optimal agent's deployment strategy, we assume that we deal with an intelligent intruder. This means that the intruder is able to deduce the strategy of the agent. To take this into account, for example by using randomized strategies, we use game theoretical models which are developed to model situations in which two or more players interact. Additionally, uncertainty may arise at several aspects. For example, there might be uncertainty in sensor observations, risk levels of certain areas, or travel times. We address this uncertainty by combining game theoretical models with stochastic modeling, such as queueing theory, Bayesian beliefs, and stochastic game theory.

This thesis consists of three parts. In the first part, we introduce two game theoretical models on a network of queues. First, we develop an interdiction game on a network of queues where the intruder enters the network as a regular customer and aims to route to a target node. The agent is modeled as a negative customer which can inspect the queues and remove intruders. By modeling this as a queueing network, stochastic arrivals and travel times can be taken into account. The second model considers a non-cooperative game on a queueing network where multiple players decide on a route that minimizes their sojourn time. We discuss existence of pure Nash equilibria for games with continuous and discrete strategy space and describe how such equilibria can be found.

The second part of this thesis considers dynamic games in which information that becomes available during the game plays a role. First, we consider partially observable agent-intruder games (POAIGs). In these types of games, both the agent and the intruder do not have full information about the state space. However, they do partially observe the state space, for example by using sensors. We prove the existence of approximate Nash equilibria for POAIGs with an infinite time horizon and provide methods to find (approximate) solutions for both POAIGs with a finite time horizon and POAIGs with an infinite time horizon. Second, we consider anti-submarine warfare operations with time dependent strategies where parts of the agent's strategy becomes available to the intruder during the game. The intruder represents an enemy submarine which aims to attack a high value unit. The agent is trying to prevent this by the deployment of both frigates and helicopters.

In the last part of this thesis we discuss games with restrictions on the agent's strategy. We consider a special case of security games dealing with the protection of large areas for a given planning period. An intruder decides on which cell to attack and an agent selects a patrol route visiting multiple cells from a finite set of patrol routes, such that some given operational conditions on the agent's mobility are met. First, this problem is modeled as a two-player zero-sum game with probabilistic constraints such that the operational conditions are met with high probability. Second, we develop a dynamic variant of this game by using stochastic games. This ensures that strategies are constructed that consider both past actions and expected future risk levels. In the last chapter, we consider Stackelberg security games with a large number of pure strategies. In order to construct operationalizable strategies we limit the number of pure strategies that is allowed in the optimal mixed strategy of the agent. We investigate the cost of these restrictions by introducing the price of usability and develop algorithmic approaches to calculate such strategies efficiently.

Samenvatting

In dit proefschrift ontwikkelen we wiskundige modellen voor de optimale inzet van beveiligingseenheden. De twee belangrijkste uitdagingen daarbij zijn het adaptieve gedrag van de tegenstander en onzekerheid in het model.

We behandelen verschillende toepassingen die betrekking hebben op beveiliging en modelleren deze als een spel tussen een beveiliger en een indringer. De beveiliger vertegenwoordigt bijvoorbeeld de kustwacht, douane of militaire middelen. De indringer vertegenwoordigt de tegenstander van deze beveiliger, bijvoorbeeld illegale vissers, terroristen of vijandige onderzeeërs.

Om de optimale strategie van de beveiliger te bepalen gaan we ervan uit dat we te maken hebben met een intelligente indringer. Dit betekent dat de indringer de strategie van de beveiliger kan afleiden en zijn eigen strategie hierop aanpast. Om met dit adaptieve gedrag rekening te houden, gebruiken we speltheoretische modellen die zijn ontwikkeld om situaties te modelleren waarbij twee of meer spelers interactie met elkaar hebben. Naast dit adaptieve gedrag speelt onzekerheid een rol bij verschillende aspecten van beveiliging. Denk daarbij aan onvolledigheid in sensorwaarnemingen, reistijd of onvoorspelbaarheid van risicolevels van gebieden. We modelleren deze onzekerheid door speltheoretische modellen te combineren met stochastische modellen, zoals wachtrijtheorie, Bayesian belief functies en stochastische speltheorie.

Dit proefschrift bestaat uit drie delen. In het eerste deel introduceren we twee speltheoretische modellen op een netwerk van wachtrijen. Eerst ontwikkelen we een interdictiespel op een netwerk van wachtrijen waarbij de indringer het netwerk binnendringt als een gewone klant en zijn optimale route kiest naar een bepaald doel. De beveiliger wordt gemodelleerd als een negatieve klant die de wachtrijen kan inspecteren en indringers kan onderscheppen. Door dit te modelleren als een netwerk van wachtrijen, kan rekening gehouden worden met stochastische aankomsten en reistijden. Het tweede model beschouwt een niet-coöperatief spel op een netwerk van wachtrijen waarbij meerdere spelers een route kiezen die hun verblijfstijd minimaliseert. We kijken naar het bestaan van zuivere Nash evenwichten voor spellen met een continue en discrete strategieruimte en beschrijven hoe zulke evenwichten gevonden kunnen worden.

Het tweede deel van dit proefschrift gaat over dynamische spellen waarbij nieuwe informatie zichtbaar wordt tijdens het spel. Eerst bekijken we spellen tussen een beveiliger en indringer met beperkte observaties. In dit soort spellen hebben zowel de beveiliger als de indringer onvolledige informatie over de toestandsruimte. Echter, beiden nemen de toestandsruimte wel gedeeltelijk waar, bijvoorbeeld door sensoren te gebruiken. We bewijzen dat er benaderingen van Nash evenwichten bestaan voor deze spellen met een oneindige tijdshorizon en ontwikkelen oplossingsmethodes voor zowel spellen met een eindige als met een oneindige tijdshorizon. Daarna bekijken we anti-

onderzeeboot operaties met tijdsafhankelijke strategieën. Hierbij wordt tijdens het spel een deel van de informatie over de strategie van de beveiliger beschikbaar voor de indringer. De indringer vertegenwoordigt een vijandelijke onderzeeër die waardevolle objecten, zoals vrachtschepen, probeert aan te vallen. De beveiliger probeert dit te voorkomen door de inzet van zowel fregatten als helikopters.

In het laatste deel van dit proefschrift bespreken we spellen waarbij de strategieruimte van de beveiliger beperkt is. We bekijken een speciaal geval van spellen in het beveiligingsdomein waarbij de bescherming van grote gebieden gedurende een bepaalde planningsperiode een rol speelt. Een indringer beslist welk deelgebied wordt aangevallen en de beveiliger selecteert een patrouille die meerdere deelgebieden bezoekt, zodanig dat aan bepaalde operationele voorwaarden wordt voldaan. Eerst wordt dit probleem gemodelleerd als een nul-som spel met kansconstraints, zodat met hoge waarschijnlijkheid aan de operationele voorwaarden wordt voldaan. Daarna ontwikkelen we een dynamische variant van dit spel door stochastische speltheorie te gebruiken. Dit zorgt ervoor dat er strategieën worden geconstrueerd die rekening houden met zowel voorgaande acties als de verwachte toekomstige risicolevels. In het laatste hoofdstuk bekijken we Stackelberg-spellen met een groot aantal zuivere strategieën. Om implementeerbare strategieën te construeren, beperken we het aantal zuivere strategieën dat is toegestaan in de optimale gemengde strategie van de beveiliger. We onderzoeken de kosten van deze beperkingen door de *price of usability* te introduceren en we ontwikkelen een algoritmische methode om zulke strategieën efficiënt te berekenen.

About the author

Corine Laan was born in Amsterdam, the Netherlands, on June 11, 1990. After finishing high school at Sint Nicolaas Lyceum in Amsterdam in 2008, she moved to Enschede to study Technical Medicine.

After finishing her first year, Corine switched to study Applied Mathematics in 2009 for which she obtained her bachelor's degree in 2012. She continued in the master program of Applied Mathematics with a specialization in Stochastic Operations Research. During her master, she did a research internship at Dalhousie University, Halifax, Canada, where she studied the offload zone at the emergency department of an hospital. Her master project was at the Jeroen Bosch Hospital in Den Bosch, the Netherlands, where she developed models to optimally schedule patients at an outpatient surgical clinic.

In November 2014, Corine started her PhD research at the University of Twente on a joined project with the Netherlands Defense Academy and TNO. This research was about the optimal deployment of security forces using game theory and stochastic modeling and resulted in this PhD thesis. During this PhD project she visited the University of Southern California in Los Angeles, USA, to work on a joint project with the Teamcore research group.

Currently, Corine is working as a researcher at the Urban Analytics group of the Amsterdam University of Applied Sciences.