Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering

Doctoral thesis proposal

Algorithms for Solving Partially Observable Stochastic Games

Karel Horák

Supervisors: Prof. Dr. Michal Pěchouček MSc.
Mgr. Branislav Bošanský, Ph.D.

Study programme: Electrotechnics and Information Science
Field of study: Information Science and Computer Engineering

June 5, 2017

Contents

CHAPTER 1

Introduction

Sequential decision making plays a significant role in our daily lives. Every day we make sequences of decisions to accomplish our goals and, if possible, we aim to achieve them in an optimal, most convenient way for us, e.g. with respect to associated costs, time efficiency or execution safety. Specifically, in an environment where the outcomes also depend on actions of other decision makers, the optimality of the decisions we make becomes critical. This is especially true in mission critical scenarios, such as those arising in security related applications. Malicious users, such as attackers and other threats for the defended system, will try to identify weaknesses of our decisions – and subsequently, exploit these in their favor. In order to avoid falling victim to a successful attack performed by an advanced adversary, we need to understand the strategic interactions in the environment and anticipate future actions of our opponent.

Understanding strategic interactions between adversarial parties in complex domains is, however, challenging, and a computational support to drive our decisions is thus needed. To this end, we use the mathematical framework of game theory to quantify the risks related to security operations and to derive powerful defensive strategies that cannot be easily exploited by the adversary. Game theory has its roots in the economy, but since then it has found numerous applications in nearly all the fields including biology (Sandholm, 2015), social sciences (Shubik, 1984) and of course also security. The applications of game theory in security involve, to name a few, the protection of critical infrastructures (Pita et al., 2008; Kiekintveld et al., 2009; Shieh et al., 2012), computer networks (Vanek et al., 2012), wildlife protection (Fang et al., 2015, 2016) or patrolling (Basilico et al., 2009; Vorobeychik et al., 2014; Basilico et al., 2016).

Game theory provides us with mathematical concepts to study, understand and represent the rational behavior of self-interested agents. In order to apply these theoretical concepts to drive the decision making in practice, we need to study game-theoretic models that closely correspond to the reality. Furthermore, we need to devise efficient and scalable algorithms for the computation of optimal, or near-optimal strategies. A standard model for representing sequential decision making in adversarial domains are game trees, or extensive-form games (Koller et al., 1996; Shoham and Leyton-Brown, 2008), which represent the sequential interaction in the form of a tree where the internal nodes correspond to the points where an agent has to decide. While we have seen major progress in playing selected large-scale extensive-form games on a professional level recently (Silver et al., 2016; Moravčík et al.,

2017), solving these games, in general, remains hard.

From the standpoint of security applications, another issue of extensive-form games is that the trees are assumed to be finite, and the players get their rewards once the game reaches one of its terminal states only. This means that they can only be used to represent interactions that last for a well-defined finite horizon (and even this finite horizon cannot be large since the size of the representation grows exponentially with the horizon). This is not convenient for security applications. In security problems, we typically cannot restrict ourselves to a finite horizon since the length of the planning horizon is either unknown or even impossible to define. As an example, consider a task of protecting a power plant. In this setting, we have to take security measures to ensure safe operation of the plant for its entire lifetime. It is impossible to decide a finite horizon, say two years, and take actions accordingly since, in such a finite-horizon setting, our actions would have ensured security for the first two years of the operation, but the power plant would become vulnerable after this period is over.

We therefore need to study models that do not suffer from these issues and can be applied to reason about problems with infinite horizon arising in security. We focus on the model of partially observable stochastic games which allows us to reason about strategies in infinite-horizon setting while accounting for the uncertainty of the decision makers which is inherent to security problems. Finding optimal strategies in such dynamic games with imperfect information is, however, often computationally challenging. The complexity of partially observable stochastic games have been shown to be NEXP^{NP}-complete (Goldsmith and Mundhenk, 2008) and many problems related to partially observable stochastic games are even undecidable (Madani et al., 1999). These complexity results are, however, oftentimes based on a structure of the game which is overly generic and may not be required to model real-world security scenarios.

Apart from negative complexity-theoretical results on partially observable stochastic games, these games also pose significant challenges in practice. We illustrate these challenges by comparing partially observable stochastic games with extensive-form games. Consider a simple game with 2 states, 2 actions of each of the players and 2 observations. Even this trivial game has up to $(2 \times 2 \times 2 \times 2)^{15}/2 \approx 6 \times 10^{17}$ unique histories when considering horizon 15. This is more than the number of histories in the largest two-player zero-sum extensive-form game with imperfect information that has been solved by finding (near-)optimal strategies in the whole game – heads-up limit hold'em poker (Bowling et al., 2015) – which required 900 core-years of computing time to get solved. Moreover, note that if we consider discounting with discount factor $\gamma = 0.95$, the weight of rewards received in the first 15 stages of the game is approximately the same as the weight of rewards received from round 16 onwards and hence this finite horizon approximation of the game may be highly inaccurate. This illustrates the need for specialized algorithms for solving two-player zero-sum partially observable stochastic games.

We aim to understand the way, how the structure of the game (especially the structure of information the players have) influence the complexity of solving the game and the possibility to use efficient algorithms to compute optimal strategies. Currently, we have focused on the class of two-player zero-sum stochastic games where one of the players is perfectly informed – we term these as one-sided partially observable stochastic games. Our results (see Chapter 3) show that we can apply an efficient algorithm based on the principles of point-based value iteration algorithms seen in POMDPs (Smith and Simmons, 2012) to solve games from this class. Our algorithm provably converges to ϵ-optimal strategies (for any $\epsilon > 0$). More-

over, the experimental results show that our domain-independent algorithm can closely approximate solutions of selected games with up to thousands of states. We argue that this class of games is of great relevance to the security community since it allows us to devise robust defensive strategies with safety guarantees. Nevertheless, some problems require a more general structure of information, and we thus aim to understand what structures of information still allow us to apply efficient solution techniques.

1.1 Structure of the Thesis Proposal

In Chapter 2, we provide an overview of basic concepts we are using and the related work on (partially observable) stochastic games. We also briefly summarize our contributions and comment on papers we published. In Chapter 3, a detailed description of our algorithm for solving partially observable stochastic games with one-sided information is given. We also describe our novel application of this model in the field of network security. Finally, we outline future directions of our research and discuss their importance in Chapter 4.

CHAPTER 2

Technical Background and State of the Art

2.1 Game Theory

Game theory is a mathematical framework that allows us to study cooperation and conflict between rational decision makers, termed players. In the most general setting, we need to define four basic components that form the game – *players* who play the game, *actions* they can use, their *payoffs* for possible outcomes of the game and the *information* they have about the game (Rasmusen and Blackwell, 1994).

We will be mainly focusing on stochastic games (see Section 2.2), but in order to demonstrate the concepts of game theory, we present a simpler model first. *Normal form games* (often also termed as *strategic form*) are well-suited for representing one-shot interactions (Shoham and Leyton-Brown, 2008). Formally a N-player normal-form game is a tuple $\langle \{\mathcal{A}_i\}_{i=1}^N, \{u_i\}_{i=1}^N \rangle$, where $\mathcal{A}_i$ is the finite action set of Player i and $u_i : (\bigtimes_{i=1}^N \mathcal{A}_i) \to \mathbb{R}$ is the utility function which declares the payoff $u_i(\overline{\mathbf{a}})$ of Player i when players jointly opt to play actions $\overline{\mathbf{a}} \in \bigtimes_{i=1}^N \mathcal{A}_i$.

Each of the players selects a *strategy* σ_i to play the game. This strategy assigns a distribution over actions to each possible information state the player can attain about the game. We refer to the combination of strategies of all players as a *strategy profile* $\sigma = (\sigma_1, \ldots, \sigma_N)$, where N is the number of players. We will denote the expected payoff Player i gets when strategy profile $(\sigma_1, \ldots, \sigma_N)$ is applied by $u_i(\sigma_1, \ldots, \sigma_N)$.

We illustrate the concept of strategies and payoffs on the example of normal-form games. Since the normal-form games do not involve any sequential interaction of the players (the game is played just once), players do not get any information in the course of the game. For this reason, strategies in normal-form games are simple – $\sigma_i \in \Delta(\mathcal{A}_i)$ for Player i (where $\Delta(\mathcal{A}_i)$ is the set of all distributions over Player i's actions). The utility of strategy profile $\sigma = (\sigma_1, \ldots, \sigma_N)$ is in the case of normal-form games an expectation over all possible outcomes of the game $\overline{\mathbf{a}} \in \bigtimes_{i=1}^N \mathcal{A}_i$:

$$u_i(\sigma) = \sum_{\overline{\mathbf{a}} \in \bigtimes_{i=1}^N \mathcal{A}_i} u_i(\overline{\mathbf{a}}) \cdot \prod_{j=1}^{N} \sigma_j(a_j) \,. \tag{2.1}$$

Players typically choose their strategies in a way to maximize their individual expected payoff $u_i(\sigma_1, \ldots, \sigma_N)$ when facing other rational decision makers. The most famous solution concept with respect to this objective is the Nash equilibrium (Nash, 1951; Shoham and Leyton-Brown, 2008). Formally, a Nash equilibrium is a strategy profile $(\sigma_1, \ldots, \sigma_N)$ such that

$$\forall i \in [1..N]\ \forall \sigma_i' \in \Sigma_i : \\ u_i(\sigma_1, \ldots, \sigma_{i-1}, \sigma_i', \sigma_{i+1}, \ldots, \sigma_N) \leq u_i(\sigma_1, \ldots, \sigma_{i-1}, \sigma_i, \sigma_{i+1}, \ldots, \sigma_N)\ . \quad (2.2)$$

This means that no player can unilaterally change his strategy and improve his payoff. No player thus wants to deviate, and the strategy profile is stable. Unless stated otherwise, we will be focusing on the algorithmic computation of Nash equilibrium in games, or its ϵ-approximation (ϵ-Nash equilibrium) where no player can improve his payoff by more than ϵ by deviating from the equilibrium point:

$$\forall i \in [1..N]\ \forall \sigma_i' \in \Sigma_i : \\ u_i(\sigma_1, \ldots, \sigma_{i-1}, \sigma_i', \sigma_{i+1}, \ldots, \sigma_N) \leq u_i(\sigma_1, \ldots, \sigma_{i-1}, \sigma_i, \sigma_{i+1}, \ldots, \sigma_N) + \epsilon\ . \quad (2.3)$$

2.2 Stochastic Games

We consider stochastic games that are played on a finite graph. Vertices of this graph are formed by a finite set of states of the game $\mathcal{S}$ and the transitions between these states are controlled jointly by N decision makers, termed players. Denote $\Delta(X)$ a set of all probability distribution on a finite set X. Formally, an N-player stochastic game is a a tuple $\langle \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^N, \mathcal{T}, b^0 \rangle$, where $\mathcal{A}_i$ is the set of actions (or control inputs) Player i can use to control the game, $\mathcal{T} : \mathcal{S} \times (\bigtimes_{i=1}^N \mathcal{A}_i) \to \Delta(\mathcal{S})$ is a probabilistic transition function depending on the current state of the game and the joint control inputs of the players and $b^0 \in \Delta(\mathcal{S})$ is the distribution over initial states of the game.

A play in a stochastic game proceeds as follows. First an initial state s^0 is sampled from b^0. Then, at each stage t of the game, players simultaneously decide their actions $\overline{\mathbf{a}}^t = (a_1^t, \cdots, a_N^t)$ and the game moves to a new state s^{t+1} with probability $\mathcal{T}(s^t, \overline{\mathbf{a}}^t)(s^{t+1})$. We will denote a sequence of states $s_0 s_1 \ldots$ encountered during a play in a stochastic game as a *trace*. A play can either last for a finite, previously fixed number of stages $T \in \mathbb{N}$ (*finite horizon*), or the number of stages can be unbounded (*infinite* or *indefinite horizon*).

When $N = 1$, stochastic games correspond to Markov decision processes (MDPs) (Filar and Vrieze, 1997), or partially observable Markov decision processes (POMDPs) in the case of a partially observable environment (Smallwood and Sondik, 1973; Monahan, 1982; Smith and Simmons, 2012). Stochastic games can thus be seen as a generalization of these models.

Complete and Partial Observability

Players can be either able to observe every aspect of the game perfectly (completely observable games), or they may have limited information about the environment (partial observability). In the former case, the players base their decisions on the past history of the play $h \in \mathcal{S}(\mathcal{A}_1 \cdots \mathcal{A}_N \mathcal{S})^*$ (where we treat $\mathcal{S}(\mathcal{A}_1 \cdots \mathcal{A}_N \mathcal{S})^*$ as a regular expression), i.e. the states that were entered in the course of the game and sequences of actions all players have

played. A strategy σ_i of Player i in the completely observable case is thus a mapping from histories to distributions of actions Player i can use, i.e. $\sigma_i : \mathcal{S}(\mathcal{A}_1 \cdots \mathcal{A}_N \mathcal{S})^* \to \Delta(\mathcal{A}_i)$.

If a player cannot observe the game perfectly, he does not get access directly to the history h of the play. Instead, he receives an observation $o_i^t \in \mathcal{O}_i$ in each round. Let us extend the model of a stochastic game with a set of observation $\mathcal{O}_i$ for each of the players and modify the transition function $\mathcal{T}$ to characterize the observations the players get, i.e.

$$\mathcal{T} : \mathcal{S} \times (\bigtimes_{i=1}^{N} \mathcal{A}_i) \to \Delta\left((\bigtimes_{i=1}^{N} \mathcal{O}_i) \times \mathcal{S}\right) . \tag{2.4}$$

We term such a model with partially observable environment as *partially observable stochastic games* (POSGs). We consider the perfect recall, where players remember the actions they played and observation they received from the game environment; thus they can base their decisions on their action-observation histories. We will denote action-observation history of Player i by $\omega \in (\mathcal{A}_i \mathcal{O}_i)^*$. The strategy of Player i in the partially observable setting is thus a mapping $\sigma_i : (\mathcal{A}_i \mathcal{O}_i)^* \to \Delta(\mathcal{A}_i)$.

Objectives in Stochastic Games

We are assuming rational players that take their actions to accomplish their well-defined objectives. We will now provide an overview of the most significant objectives that are applied to describe player's intents in interactions over a graph. We can divide objectives into two major categories – optimization objectives and objectives based on ω-regular languages. We will cover both of these classes of objectives now.

Optimization Objectives

In the case of optimization objectives, Player i is receiving rewards for each of the transitions performed in the system. These rewards are described by the Player i's reward function $\mathcal{R}_i : \mathcal{S} \times (\bigtimes_{i=1}^{N} \mathcal{A}_i) \to \mathbb{R}$ where $\mathcal{R}_i(s, \overline{\mathbf{a}})$ expresses the reward Player i gets when players jointly use actions $\overline{\mathbf{a}}$ in state s. We will denote the reward received by Player i at time t by $r_{t,i}$.

The most common optimization objective in infinite-horizon problems is the *discounted-sum* objective Disc_γ^i (Filar and Vrieze, 1997). In this setting, the player weighs reward $r_{t,i}$ received at time t with a weight of γ^{t-1} for some fixed constant $0 < \gamma < 1$ termed *discount factor*.

$$\mathsf{Disc}_\gamma^i = \sum_{t=1}^{\infty} \gamma^{t-1} \cdot r_{t,i} . \tag{2.5}$$

The use of discounting ensures that the infinite sum converges and the optimization problem from the player's perspective is well-defined.

The discounted-sum objective accounts for the impatience of the player and puts less weight on rewards received at a later time. In some cases, however, this unequal treatment of rewards is undesirable. Assume that we are taking actions to shut down a reactor (the example is taken from (Kolobov et al., 2011)). Failing to perform a successful shutdown is equally undesirable regardless of the time this failed attempt occurs. Leaving out the discounting term, however, need not result in a well-posed optimization problem as the infinite sum of rewards need not converge. We will cover two of the settings where the undiscounted sum of rewards is applicable.

First, and the most straightforward, is to limit the horizon of the game to a finite number of stages T. In such a setting, we get a finite-horizon undiscounted sum objective Undisc_T^i (e.g. (Filar and Vrieze, 1997; Hansen et al., 2004)) defined as

$$\mathsf{Undisc}_T^i = \sum_{t=1}^{T} r_{t,i} \,. \tag{2.6}$$

While the finite horizon ensures that the sum from Equation (2.6) is finite, the restriction to a finite horizon is often undesirable. As an example, assume that we are designing a defensive strategy to protect a power plant against external threats. In such a setting, there is no finite T which would be sufficient to provide safety guarantees, and we need to accept infinite duration of the game.

If we want to guarantee that the infinite-horizon undiscounted sum objective Undisc^i

$$\mathsf{Undisc}^i = \sum_{t=1}^{\infty} r_{t,i} \tag{2.7}$$

is well-defined, we need to impose restrictions on the structure of the game. Examples of models where the Undisc^i objective is used are single-player Goal-MDPs and Goal-POMDPs (Bonet and Geffner, 2009). In these models, the agent is trying to reach a set of target states $T \subseteq \mathcal{S}$ with the minimal expected cost. The costs for each step before reaching the goal are positive – and hence an agent minimizing the total undiscounted cost will aim to reach the goal quickly with high probability (otherwise his Undisc^i cost would have been infinite). Note that the authors of (Bonet and Geffner, 2009) have shown that every POMDP with the discounted-sum objective is equivalent to a Goal-POMDP, hence the Goal-POMDP model is more general.

Apart from optimizing the discounted or undiscounted sum of the rewards, we can focus on optimizing their average. This yields an objective known as limit-average or mean-payoff (Filar and Vrieze, 1997; Degorre et al., 2010), $\underline{\mathsf{MP}}^i$ or $\overline{\mathsf{MP}}^i$, defined as

$$\underline{\mathsf{MP}}^i = \liminf_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} r_{t,i} \qquad \overline{\mathsf{MP}}^i = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} r_{t,i} \quad . \tag{2.8}$$

ω-Regular Objectives

ω-regular objectives are commonly applied in verification community for model checking. These objectives are based on ω-regular languages which extend regular languages to account for infinite words (i.e. infinite plays of the game) (Baier et al., 2008; Ibsen-Jensen, 2013). The goal of the player is in the case of ω-regular objectives defined using an ω-regular language $\mathcal{L}$. The player aims to ensure that the trace, i.e. a sequence of states encountered in the course of the game, is an infinite word in the language $\mathcal{L}$ (in such a case the player wins). In this section, we cover the standard ω-regular objectives and their application to stochastic games.

The reachability objective $\mathsf{Reach}(T)$ (for some $T \subseteq \mathcal{S}$) is satisfied for trace $s_0 s_1 \ldots$ if for some $t \geq 1$, $s_t \in T$ – that means if some of the states in T is eventually reached. A complementary objective to reachability is the safety objective $\mathsf{Safe}(\mathcal{S} \setminus T)$. This objective is satisfied for the given trace if for every $t \geq 1$, $s_t \notin T$ – that means states in T are never reached. Both of these objectives have considerable implications for security related applications. We can use reachability objective to reason about reaching a desirable state (e.g.

capturing an attacker) and safety objective to reason about our ability to keep the defended system in safe states (e.g. no attack is occurring, or it has been successfully bounced). Other common ω-regular objectives are $\mathsf{Buchi}(T)$ and $\mathsf{coBuchi}(T)$ objectives. In the former case, the agent aims to reach the states in T infinitely many times, in the latter case he wants to achieve that the play eventually reaches states in T and stays there forever.

From the verification standpoint, a lot of attention has been paid to solving stochastic games qualitatively (see e.g. (De Alfaro et al., 2007)). A qualitative solution of the game aims to establish whether there exists a strategy that achieves a winning trace (i.e. a trace which is an accepting run with respect to an ω-regular language $\mathcal{L}$) surely, almost-surely or limit-surely. If neither of these options hold, we can solve the game quantitatively and establish the value of the game (i.e. the maximum probability of achieving a winning trace against any strategy of the opponent). In the two-player case, this means to solve the following optimization problem.

$$\underline{V} = \sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} \Pr_{\sigma_1,\sigma_2}[s_1 s_2 \ldots \in \mathcal{L}] \tag{2.9}$$

Combining Objectives

Optimization and ω-regular objectives are not incompatible, and it is possible to use them at the same time. A prominent example of a combination of these types of objectives is the optimal-cost almost-sure reachability used in POMDPs (Chatterjee et al., 2016). In this setting, we use the infinite-horizon undiscounted-sum objective to express agent's preferences amongst the possible plays, but we restrict ourselves to policies that are guaranteed to reach one of the target states with probability 1.

Note that one has to be careful when extending these objectives to a multi-player setting. Strategies achieving the almost-sure reachability need not exist even if the value of the game, i.e. the probability of reaching the target in the limit, is 1 (Hansen et al., 2009; De Alfaro et al., 2007). This is caused by the fact that one has to play unsafe actions with positive probability in some settings, even though this probability can be arbitrarily close to zero. In fact, even strategies that achieves probability of winning of at least $1 - \epsilon$ for some $0 < \epsilon < 1/2$ may be impractical to compute and to use since the probabilities of playing unsafe actions (playing which is necessary to attempt to win the game) might be as low as $(\epsilon^2/(1-\epsilon))^{2^{|\mathcal{S}|-2}}$, i.e. we need a doubly-exponential patience to win the game (Hansen et al., 2009).

Zero-Sum and Cooperative Objectives

In multi-player games ($N \geq 2$), we establish either competition or cooperation among the agents by choice of reward functions $\mathcal{R}_i$. If $\mathcal{R}_i(s, \overline{\mathbf{a}}) = \mathcal{R}_j(s, \overline{\mathbf{a}})$ for every $i, j \in [1..N]$, $s \in \mathcal{S}$ and $\overline{\mathbf{a}} \in \bigtimes_{i=1}^{N} \mathcal{A}_i$, we obtain cooperative behavior of the agents since they all jointly optimize a shared utility function. Models with cooperative agents are commonly known as Dec-MDPs (or Dec-POMDPs for the partially observable case) (Bernstein et al., 2002).

We will be, however, mainly focusing on the competition between the agents. In two-player games (i.e. $N = 2$), we get a strict competition among the agents when $\mathcal{R}_1(s, a_1, a_2) = -\mathcal{R}_2(s, a_1, a_2)$ for every $s \in \mathcal{S}$, $a_1 \in \mathcal{A}_1$ and $a_2 \in \mathcal{A}_2$. In such a case, $\mathcal{R}_1(s, a_1, a_2) + \mathcal{R}_2(s, a_1, a_2) = 0$ and hence we call these games as zero-sum.

Note that we can achieve both the cooperation and strict competition even under the ω-regular objectives. The players tend to cooperate if they share the same objective, while they

are competing when their objectives are mutually exclusive (an example of such a pair of objectives are $\mathsf{Reach}(T)$ and $\mathsf{Safe}(\mathcal{S} \setminus T)$).

2.3 Partially Observable Markov Decision Processes

Partially observable Markov decision processes (POMDPs) can be seen as a special case of partially observable stochastic games, where only a single player interacts with the environment (i.e. $N = 1$). POMDPs are traditional models for planning under uncertainty in an environment with a single agent. Due to the absence of other decision makers, the player can keep track of the *belief* $b \in \Delta(\mathcal{S})$, i.e. the distribution over the states of the system, over time. Every time he plays an action $a \in \mathcal{A}_1$ and receives an observation $o \in \mathcal{O}_1$, he updates his belief $b \in \Delta(\mathcal{S})$ to $b' \in \Delta(\mathcal{S})$ using the Bayes formula (Smallwood and Sondik, 1973)

$$b'(s') = \eta \sum_{s \in \mathcal{S}} b(s) \cdot \mathcal{T}(s, a)(o, s') \,. \tag{2.10}$$

where η is the normalizing constant. We denote $b' = \tau(b, a, o)$. The ability of the player to keep track of the beliefs allows us to convert a POMDP to a belief-state MDP and apply a value iteration algorithm.

The value iteration algorithm aims to find an optimal value function $v^* : \Delta(\mathcal{S}) \to \mathbb{R}$ that maps belief states to expected utility of the decision maker. We can then use this value function to control the POMDP optimally. The algorithm starts with an initial, approximate value function v^0 and subsequently construct a sequence of value functions $\{v^t\}_{t=0}^{\infty}$ such that $v^{t+1} = Hv^t$. In the discounted-sum case, this operator maximizes the expectation over the immediate rewards and the discounted payoff in the remainder of the play:

$$Hv^t(b) = \max_{a \in \mathcal{A}_1} \left[\sum_{s \in \mathcal{S}} b(s) \cdot \mathcal{R}_1(s, a) + \gamma \sum_{o \in \mathcal{O}_1} \Pr[o|b, a] \cdot v^t(\tau(b, a, o)) \right] . \tag{2.11}$$

For every finite $t \geq 0$, the value function v^t is piecewise linear and convex and we can thus represent it as a point-wise maximum over a finite set of linear functions, termed α-vectors (Smallwood and Sondik, 1973). The authors have also presented an algorithm that constructs α-vectors of v^{t+1} by projecting those representing v^t in an exhaustive manner. The number of vectors needed to represent value function v^t can be, however, prohibitively large which prevents the algorithm from scaling up. Despite numerous attempts to reduce the number of α-vectors constructed in the course of the algorithm (Monahan, 1982; Littman, 1994; Zhang and Liu, 1996), the exact value iteration algorithm for POMDPs can solve problems of toy size only.

To avoid this problem, often referred to as curse of history, practical approaches focus on approximate versions of the algorithm (Pineau et al., 2003; Smith and Simmons, 2004, 2012). The approximate versions of value iterations received great attention and allowed to push the limits of scalability of POMDP models significantly. The key idea behind these algorithms is to generate only a limited number of vectors (corresponding to a finite set of belief points) in each iteration, instead of computing an optimal, exhaustive value backup. The main difference between various versions of approximate value iteration algorithm lies in the way the beliefs are being selected.

2.4 Partially Observable Stochastic Games and Dec-POMDPs

Solving Partially Observable Stochastic Games and Dec-POMDPs

Partially observable stochastic games (POSGs) generalize POMDPs by making multiple agents interact with the environment at the same time. Throughout the course of the play, each of the players receives her private observations and remembers her own actions – she does, however, have only limited information about actions and observations of other players and the states of the game. Due to this limitation, players often attain differing beliefs (Hansen et al., 2004) and thus they need to reason about current beliefs of other players in order to play optimally.

Most of the approaches resolve this by explicitly reasoning about histories of the play and aim to derive strategies that explicitly map action-observation histories to distributions over actions (similarly to behavioral strategies in extensive form games). In (Hansen et al., 2004), a bottom-up dynamic programming approach for constructing relevant finite-horizon policy trees for individual players is adopted. The main idea behind this approach is that we can safely ignore policy trees (and therefore avoid considering them in later stages of dynamic programming) that are dominated in the whole belief space. Strategies generated by this algorithm are then used to form a matrix form of the POSG which can then be solved by standard techniques. The number of such policies grows quickly for practical problems, and this approach hence allows us to handle only problems with a small horizon (≈ 4). (Kumar and Zilberstein, 2009) improve the scalability of this algorithm by applying a more aggressive pruning based on ϵ-dominance, however, at the cost of sacrificing optimality. The drawback of both of these algorithms is that the size of such history-dependent strategies is exponential in the planning horizon which makes the adoption of algorithms relying on construction of policy trees impractical in the infinite-horizon setting.

A common approach in Dec-POMDPs – a subclass of POSGs where players optimize a single shared utility function – is to represent infinite-horizon policies approximately using finite state controllers (FSCs) with a bounded size. Techniques for construction of such controllers involve, to name a few, iterative improvement methods (such as DEC-BPI, (Bernstein et al., 2009)), nonlinear programming (Amato et al., 2010) or likelihood maximization (Kumar et al., 2015). Many of these approaches rely on specific reward structure seen in Dec-POMDPs – due to the cooperative nature of the players, deterministic policies are sufficient which does not hold in the general case (see Figure 2.1). Significantly less attention has been paid to adversarial POSGs, and to the best of our knowledge, there have been no attempts on transferring FSC approaches to the noncooperative setting of POSGs with arbitrary (or even only zero-sum) reward structures.

Complexity of Partially Observable Stochastic Games and Dec-POMDPs

In computational complexity theory, we are interested in determining a complexity class of yes/no problems. Solving partially observable stochastic games is inherently an optimization task (players aim to optimize their utility values) and in order to discuss the complexity of solving POSGs a way to convert these optimization problems to yes/no problems is needed. A common approach is to consider problems of the following form: "Is there a strategy (a strategy profile) that guarantees an expected value of at least K?" This decision problem has been shown to be NEXP-complete for the class of cooperative Dec-POMDPs with finite horizon (Bernstein et al., 2002).

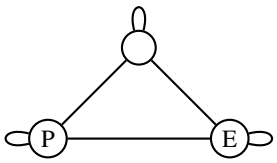

Figure 2.1: A pursuit-evasion game played on a three-vertex graph (position of the pursuer is denoted by P, a position of the evader by E). Players choose the vertex they will move next simultaneously. If the evader knows the next vertex of the pursuer (since the pursuer uses a deterministic strategy), he can always avoid entering the same vertex as the pursuer. By randomizing the actions of the pursuer we ensure that the evader is eventually captured.

The introduction of competition amongst the players makes the problem no easier. The decision problem for unrestricted partially observable stochastic games (i.e. with arbitrary number of players and an arbitrary reward structure) has been shown to be NEXP^{NP}-complete in (Goldsmith and Mundhenk, 2008). The proof of this result, however, involves a group of cooperative agents and gives us no information about the complexity of two-player competitive POSG (which is to the best of our knowledge unknown at the moment). We can at least establish PSPACE-hardness of two-player zero-sum POSGs (both in the two-sided and one-sided partially observable settings) since they extend POMDPs (where the problem of deciding whether a policy with expected reward of at least K in a specific finite-horizon problem is known to be PSPACE-complete, (Papadimitriou and Tsitsiklis, 1987)).

Other Models with One-Sided Partial Observability

In many domains, the complex model of partially observable stochastic games with arbitrary structure of information need not be necessary. In this section, we provide an overview of models that assume one-sided partial observability – that means that only one of the players is uncertain about the course of the game. These models are relevant for the security, since they allow us to model the worst-case type of the attacker who has a perfect information about the game.

In (McEneaney, 2004), a model with one-sided partial observability was proposed, however the author considers only finite-horizon problems where the perfectly informed player can always anticipate what action is her opponent about to use in the current time step. Due to this structure of information, the game has a turn-based character and pure strategies are therefore considered. This highlights the limitations of this model since in the simultaneous setting, optimal strategies may require randomization (see Figure 2.1).

In (Chatterjee and Doyen, 2014b), stochastic game models with one-sided partial observability were considered, however with reachability and safety objectives (a player either wants to reach a set of target states or she wants to avoid leaving a set of safe states). These objectives are fundamentally different from the optimization of an infinite sum of discounted rewards and techniques outlined in this paper are therefore not applicable in our setting.

2.5 Current Status of Our Research

We focus on two-player partially observable stochastic games that can be used to model problems arising in security applications and to reason about optimal (or near-optimal) strategies of the players – the defender and the attacker. Such strategic interactions typically involve a high degree of uncertainty of the players, however, in order to provide security guarantees, we assume that the attacker is more informed (and thus has an advantage over the defender).

In a sequence of our papers (Horak and Bosansky, 2016, 2017; Horák et al., 2017a), we have shown that two-player partially observable stochastic games with discounted-sum objective and simultaneous moves where the attacker is *perfectly* informed about the course of the game can be solved by a value iteration algorithm. This result is based on the fact that the belief over the state space $b \in \Delta(\mathcal{S})$ is a sufficient statistic to characterize future dynamics of the game. Based on this result we developed an efficient algorithm, inspired by point-based versions of value iteration (Smith and Simmons, 2012), which is able to approximate selected games with up to $\approx$ 2000 states within 2 hours. The existence of an efficient algorithm opens an avenue for applying game-theoretic strategies in a large number of scenarios. Our preliminary results use this model and algorithm to reason about defense strategies in network security (Horák et al., 2017b).

List of Publications

- Horak, K.; Bosansky, B.; and Pechoucek, M. 2017. Heuristic Search Value Iteration for One-Sided Partially Observable Stochastic Games. In Proceedings of the 31st Conference on Artificial Intelligence (AAAI-17), 558–564.

  In this paper, we provided our domain-independent algorithm for solving stochastic games with one-sided partial observability that builds upon the results from the papers listed below. We combined theoretical results (on the structure of solution and desirable properties of the algorithm) with an experimental evaluation that has shown that our algorithm is able to solve games of non-trivial sizes.

- Horak, K. and Bosansky, B. 2017. Dynamic Programming for One-Sided Partially Observable Pursuit-Evasion Games. In Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART-17), 503–510.

  We focused on a subclass of one-sided partially observable stochastic games that models pursuit-evasion scenarios on a finite graph. We studied the structure of the solution and provided an exhaustive value-iteration algorithm for this class of games. The derivation of this algorithm presented several challenges that we needed to address, mainly the adversarial nature of the problem and the need for randomization. The algorithm presented in this paper is not a practical one, however, the results have been of great significance for subsequent research.

- Horak, K. and Bosansky, B. 2016. A Point-Based Approximate Algorithm for One-Sided Partially Observable Pursuit-Evasion Games. In Proceedings of the Conference on Decision and Game Theory for Security (GameSec-16), 435–454.

The algorithm presented in this paper is a preliminary version of the algorithm from (Horák et al., 2017a), in this case only for pursuit-evasion scenarios. We have derived necessary framework to adopt the heuristic search value iteration algorithm (Smith and Simmons, 2012) in the game-theoretic setting. We haven't provided theoretical guarantees for our algorithm in this paper (which was fixed in (Horák et al., 2017a)) and focused on experimental evaluation.

- Horak, K; Zhu, Q; and Bosansky, B. 2017. Manipulating Adversary's Belief: A Dynamic Game Approach to Deception by Design for Proactive Network Security. Submitted to European Symposium on Research in Computer Security (ESORICS-17).

  In this paper, we applied our algorithm from (Horák et al., 2017a) to analyze effective defensive strategies for networked computer systems. We have studied the case where the attacker is uncertain about the state of the attack (e.g. whether he is detected or not or where the most valuable assets are located). We focused on the way we can make use of this uncertainty of the attacker to derive strong defensive strategies.

- Bosansky, B.; Cermak, J.; Horak, K.; and Pechoucek, M. 2017. Computing Maxmin Strategies in Extensive-form Zero-sum Games with Imperfect Recall. In Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART-17), 63–74.

CHAPTER 3

One-Sided Partially Observable Stochastic Games

3.1 Two-Player One-Sided POSGs

A *one-sided partially observable stochastic game* G is a tuple $G = \langle \mathcal{S}, \mathcal{A}_1, \mathcal{A}_2, \mathcal{O}, \mathcal{T}, \mathcal{R} \rangle$. The game is played for an infinite number of *stages*. At each stage, the game is in one of the states $s \in \mathcal{S}$ and players choose their actions $a \in \mathcal{A}_1$ and $a' \in \mathcal{A}_2$ simultaneously. An initial state of the game is drawn from a probability distribution $b^0 \in \Delta(\mathcal{S})$, which we treat as a parameter of the game and term the *initial belief*.

The choice of actions determines the outcome of the current stage: Player 1 gets an *observation* $o \in \mathcal{O}$ and the game moves to a state $s' \in \mathcal{S}$ with probability $\mathcal{T}_{s,a,a'}(o, s')$, where s is the current state. Furthermore he gets a reward $\mathcal{R}(s, a, a')$ for this transition. We assume the zero-sum case, hence player 2 receives $-\mathcal{R}(s, a, a')$, and we assume that the rewards are discounted over time with discount factor $\gamma < 1$. Players do not observe their rewards during the game.

We assume perfect recall, hence both players remember their respective histories. A history of the first player is formed by actions he played and observations he received, i.e. $(\mathcal{A}_1\mathcal{O})^*$. The second player has complete observation, hence $\mathcal{S}(\mathcal{A}_1\mathcal{A}_2\mathcal{O}\mathcal{S})^*$ is a set of her histories. The strategies σ_1, σ_2 of the players map each of their histories to a distribution over their actions.

Value of a Strategy and Value of the Game

In this section, we show that the value of a strategy (the expected reward of the first player playing σ_1 when the opponent plays her best response) has a linear dependence on the belief.

The value of the game G is the value of the best strategy available for each of the initial beliefs $b^0 \in \Delta(\mathcal{S})$. We represent the value of a game (based on the initial belief) as a *value function*. This function is a pointwise maximum taken over values of all strategies of the first player, which, since the value of every strategy is linear, forms a convex function.

In the convergence proof of our algorithm, we exploit that the rate of change in the value function is bounded in terms of minimum and maximum rewards of G, i.e. the value function

is K-Lipschitz continuous. We say that a function f is K-Lipschitz if it satisfies $|f(x) - f(y)| \leq K \cdot \|x - y\|_2$.

Definition 1 (Value functions)**.** *The value of a strategy σ_1 of the first player is a function $v_{\sigma_1} : \Delta(\mathcal{S}) \to \mathbb{R}$ which assigns the expected utility $v_{\sigma_1}(b^0)$ of the player 1 in the game with initial belief b^0 when the first player follows σ_1 and the second player best-responds. The value function of the game G is a function $v^* : \Delta(\mathcal{S}) \to \mathbb{R}$ that assigns the value $v^*(b^0)$ of the best strategy of the first player for each of the beliefs, i.e. $v^*(b^0) = \sup_{\sigma_1} v_{\sigma_1}(b^0)$.*

Lemma 1. *The value v_{σ_1} of a fixed strategy σ_1 of the first player is linear in the initial belief.*

Proof. The second player knows the initial state of the game as well as the strategy σ_1 she is best-responding. She therefore chooses her best-response strategy for each of the initial states of the game. The initial belief serves as coefficients of the convex combination of values of individual best responses, which is a linear function. □

We will now turn our attention to deriving the Lipschitz constant of the value function v_{σ_1} for any strategy σ_1 of the first player. The key observation to derive this constant is that the Player 1 can never get a discounted-sum payoff greater than U and smaller than L, where

$$L = \min_{(s,a,a')} \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s,a,a') \,, \quad U = \max_{(s,a,a')} \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s,a,a') \,. \tag{3.1}$$

The proof of the following lemma then relies on defining the value of the strategy by assigning these extreme values to the vertices of the belief simplex and identifying the configuration with the largest rate of change.

Lemma 2. *Value function v_{σ_1} of a fixed strategy σ_1 of player 1 is K-Lipschitz for $K = (U-L)\sqrt{|\mathcal{S}|/4}$.*

Proof sketch. We define the linear value of the strategy by assigning values v_s from the interval $[L, U]$ to the extreme beliefs $s \in \mathcal{S}$ of the simplex. Let $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{S}$ be non-empty sets that partitions $\mathcal{S}$, i.e. $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$, $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$ and $\mathcal{S}_i \neq \emptyset$. We observe that the Lipschitz constant of the value function v_{σ_1} is maximized if $||\mathcal{S}_1| - |\mathcal{S}_2|| \leq 1$ and we assign $v_s = U$ for all $s \in \mathcal{S}_1$ and $v_s = L$ for all $s \in \mathcal{S}_2$. Take two belief points $x_1, x_2 \in \Delta(\mathcal{S})$ such that

$$x_i(s) = \begin{cases} 1/|\mathcal{S}_i| & s \in \mathcal{S}_i \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

Clearly $v(x_1) - v(x_2) = U - L$. We will now turn our attention to the Euclidean distance of x_1 and x_2. It holds

$$\begin{aligned} \|x_1 - x_2\|_2 &= \sqrt{\sum_{s \in \mathcal{S}} [x_1(s) - x_2(s)]^2} = \sqrt{\sum_{s \in \mathcal{S}_1} x_1(s)^2 + \sum_{s \in \mathcal{S}_2} x_2(s)^2} \\ &= \sqrt{|\mathcal{S}_1| \cdot \frac{1}{|\mathcal{S}_1|^2} + |\mathcal{S}_2| \cdot \frac{1}{|\mathcal{S}_2|^2}} = \sqrt{\frac{1}{|\mathcal{S}_1|} + \frac{1}{|\mathcal{S}_2|}} \\ &\geq \sqrt{\frac{4}{|\mathcal{S}_1| + |\mathcal{S}_2|}} = \sqrt{4/|\mathcal{S}|} \qquad \left(\text{since } \frac{1}{a} + \frac{1}{b} \geq \frac{4}{a+b} \text{ for } a, b > 0\right) \end{aligned}$$

Therefore

$$K = \frac{|v(x_1) - v(x_2)|}{\|x_1 - x_2\|_2} \leq \frac{U - L}{\sqrt{4/|\mathcal{S}|}} = (U-L)\sqrt{|\mathcal{S}|/4}\,. \tag{3.3}$$

□

Theorem 1. *Value function v^* of the game G is convex in the initial belief and K-Lipschitz continuous for $K = (U-L)\sqrt{|\mathcal{S}|/4}$.*

Proof. The value function v^* is the supremum taken over a set of K-Lipschitz functions corresponding to values of strategies available to Player 1 (Definition 1, Lemma 2). Supremum taken over a family of bounded K-Lipschitz continuous functions is K-Lipschitz continuous. Moreover since these functions are linear (Lemma 1), the resulting value function is convex. □

3.2 Value Backup

Now we present a value iteration algorithm for solving one-sided POSGs. The algorithm approximates the value function v^* of the infinite horizon game G by considering value functions of the game with a restricted horizon. Each iteration of the algorithm improves the approximation by increasing the horizon by one step using the *value backup operator* (denoted H). Applying this operator means that players choose their Nash equilibrium strategies in the current step while assuming that the value of the subsequent stage is represented by the value function from the previous iteration.

The algorithm constructs a sequence $\{v^t\}_{t=0}^{\infty}$, starting with a value function v^0 of a game where only immediate rewards are considered. First, we discuss application of the operator in a single stage. Afterward we show the convergence when the operator is applied repeatedly.

Value Backup Operator

The value backup operator H evaluated at belief b —$[Hv](b)$— corresponds to solving a *stage game* where players choose their Nash equilibrium strategies for one stage of the game (in latter text we use $[Hv](b)$ to refer to this game as well). We denote strategies for one stage $\pi_1 \in \Delta(\mathcal{A}_1)$ for the first player and $\pi_2 : \mathcal{S} \to \Delta(\mathcal{A}_2)$ for the player 2. The utilities in $[Hv](b)$ depend both on the immediate rewards $\mathcal{R}$ and the discounted value of the subsequent game represented by value function v. The immediate rewards part depends solely on the actions played by the players:

$$R^{\text{imm}}_{\pi_1,\pi_2} = \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}_1} \sum_{a'\in\mathcal{A}_2} b(s) \cdot \pi_1(a) \cdot \pi_2(s, a') \cdot \mathcal{R}(s, a, a') \tag{3.4}$$

The first player both knows the action a he played and observes observation o. He can use this information to derive his belief for the subsequent game:

$$b^{a,o}_{\pi_2}(s') = \frac{1}{\Pr[o|a, \pi_2]} \sum_{s\in\mathcal{S}} \sum_{a'\in\mathcal{A}_2} \mathcal{T}_{s,a,a'}(o, s') \cdot b(s) \cdot \pi_2(s, a') \tag{3.5}$$

The value of the subsequent game is then the expectation taken over individual action-observation pairs (a, o) of the first player from the values of a game starting in belief $b^{a,o}_{\pi_2}$:

$$R^{\text{subs}}_{\pi_1,\pi_2}(v) = \sum_{a\in\mathcal{A}_1}\sum_{o\in\mathcal{O}} \pi_1(a)\cdot \Pr[o|a,\pi_2]\cdot v(b^{a,o}_{\pi_2})\,. \tag{3.6}$$

Since the value function is convex, utility of playing strategy profile (π_1, π_2) is convex when π_1 is fixed and linear when we fix π_2. The minimax theorem (von Neumann, 1928; Nikaido, 1954) applies and the Nash equilibrium strategy is solved by maximin/minimax:

$$[Hv](b) = \min_{\pi_2}\max_{\pi_1}\left(R^{\text{imm}}_{\pi_1,\pi_2} + \gamma\cdot R^{\text{subs}}_{\pi_1,\pi_2}(v)\right)\,. \tag{3.7}$$

Computation of Value Backup Operator

Finally, we present the way of computing $[Hv](b)$. When the value function v is piecewise linear and convex (PWLC), it can be represented by a set Γ of α-vectors and the value backup $[Hv](b)$ can be evaluated by means of linear programming. Each α-vector $\alpha \in \Gamma$ is an $|\mathcal{S}|$-tuple representing the affine value function v_{σ_1} of a fixed strategy σ_1 by specifying its values in each of the pure beliefs ($\alpha(s)$ for each $s \in \mathcal{S}$). We focus on the problem of solving the problem from the perspective of the second player first, who has to choose her strategy π_2 such that the utility V of the best responding player 1 (who chooses his pure best response $a \in \mathcal{A}_1$) is minimized.

The value of playing strategy π_2 against action $a \in \mathcal{A}_1$ equals $R^{\text{imm}}_{a,\pi_2} + \gamma R^{\text{succ}}_{a,\pi_2}(v)$, which allows us to construct a set of best-response constraints (one for each action a)

$$V \geq \sum_{s\in\mathcal{S}}\sum_{a'\in\mathcal{A}_2} b(s)\cdot\pi_2(s,a')\cdot\mathcal{R}(s,a,a') + \gamma\sum_{o\in\mathcal{O}}\Pr[o|a,\pi_2]\cdot v(b^{a,o}_{\pi_2})\,. \tag{3.8}$$

Assuming that the value function v is represented by a set Γ of α-vectors, such that $v(b) = \max_{\alpha\in\Gamma}\langle\alpha, b\rangle$ ($\langle\cdot,\cdot\rangle$ denotes an inner product), its value can be rewritten by a set of inequalities

$$v(b^{a,o}_{\pi_2}) \geq \sum_{s'\in\mathcal{S}} \alpha(s')\cdot b^{a,o}_{\pi_2}(s') \qquad \forall\alpha\in\Gamma \tag{3.9}$$

where $b^{a,o}_{\pi_2}(s')$ is represented by linear constraints corresponding to Eq. (3.5). The term $\Pr[o|a,\pi_2]$ occuring in Eqs. (3.5) and (3.8) cancels out to form the resulting linear program.

Strategy of the First Player

One way to approximate the value function by a PWLC function is to use a finite subset of strategies of the first player. Value functions of these strategies are linear (Lemma 1), and the pointwise maximum from these linear functions gives us the desired PWLC approximation. In such a case, each of the vectors in Γ corresponds to the value function of one of the strategies. The dual linear program is used to find the optimal control strategy of the first player, when duals of Eq. (3.8) correspond to the strategy to play in the first stage (when the history of the first player is empty) and duals of Eq. (3.9) prescribes what strategy to follow when (a, o) was observed in the first stage.

Convergence of the Value Backup Operator

In this section we show that a repetitive application of the value backup operator H converges to the *same* value function v^* of the infinite horizon game regardless of what value function it is applied on. We show this by demonstrating that the operator H is a contraction mapping with a factor $\gamma < 1$.

Lemma 3. *Let v, v' be value functions, $b \in \Delta(\mathcal{S})$ be a belief and π_1, π_2 (resp. π_1', π_2') be equilibrial strategies in $[Hv](b)$ (resp. $[Hv'](b)$). Assume that for every action-observation pair (a, o) of the first player, $|v(b^{a,o}_{\pi_2}) - v'(b^{a,o}_{\pi_2})| \leq \mu$. Then $|[Hv](b) - [Hv'](b)| \leq \gamma\mu$.*

Proof. Without loss of generality, assume that $[Hv](b) \leq [Hv'](b)$. If the first player plays π_1' instead of π_1 in $[Hv](b)$ (but the second player keeps his strategy π_2) he will receive utility $\underline{V} \leq [Hv](b)$ (since he does not want to deviate from his equilibrial strategy π_1). The similar holds for the second player; if she plays π_2 instead of π_2' in $[Hv'](b)$, she will receive payoff $\overline{V} \geq [Hv'](b)$.

After these changes, players use the same strategies π_1', π_2 in both games $[Hv](b)$ and $[Hv'](b)$. The expected immediate reward of playing these strategies (Eq. (3.4)) does not depend neither on v or v' and thus it is the same. Hence

$$\begin{aligned}
\overline{V} - \underline{V} &= \gamma \cdot R^{\text{subs}}_{\pi_1',\pi_2}(v') - \gamma \cdot R^{\text{subs}}_{\pi_1',\pi_2}(v) && (3.10)\\
&= \gamma \sum_{a\in\mathcal{A}_1} \sum_{o\in\mathcal{O}} \Pr[o|a,\pi_f] \cdot \pi_1'(a) \cdot [v'(b^{a,o}_{\pi_2}) - v(b^{a,o}_{\pi_2})] && (3.11)\\
&\leq \gamma \sum_{a\in\mathcal{A}_1} \sum_{o\in\mathcal{O}} \Pr[o|a,\pi_2] \cdot \pi_1'(a) \cdot |v'(b^{a,o}_{\pi_2}) - v(b^{a,o}_{\pi_2})| && (3.12)\\
&\leq \gamma \sum_{a\in\mathcal{A}_1} \sum_{o\in\mathcal{O}} \Pr[o|a,\pi_2] \cdot \pi_1'(a) \cdot \mu = \gamma\mu\,. && (3.13)
\end{aligned}$$

As $\overline{V} \geq [Hv'](b) \geq [Hv](b) \geq \underline{V}$ and $\overline{V} - \underline{V} \leq \gamma\mu$, $[Hv'](b) - [Hv](b) \leq \gamma\mu$, which completes the proof. □

Theorem 2. *The operator H is a contraction mapping under the norm $\|v - v'\| = \max_{b\in\Delta(\mathcal{S})} |v(b) - v'(b)|$. It thus has a unique fixpoint – the value function of the infinite horizon game.*

Proof. Let $\|v - v'\| \leq \mu$. Then for every $b^{a,o}_{\pi_2}$ from Lemma 3 $|v(b^{a,o}_{\pi_2}) - v'(b^{a,o}_{\pi_2})| \leq \mu$ and for every belief b, $|[Hv](b) - [Hv'](b)| \leq \gamma\mu$. The uniqueness of the fixpoint and the convergence properties follow from the Banach's fixed point theorem (Ciesielski, 2007). □

3.3 Heuristic Search Value Iteration for One-Sided POSGs

Similarly to POMDPs, the exact value iteration algorithm cannot scale for practical problems. We thus present a point-based algorithm (Algorithm 1) that by sampling the belief space bounds and approximates the true value function v^* of the game by a pair of PWLC functions $\underline{v}$ (*lower bound*), represented by a set of α-vectors Γ, and $\overline{v}$ (*upper bound*) represented as a lower convex envelope of a set of points Υ. We refer to these functions jointly as $\hat{v}$. The goal of the algorithm is to ensure that the *gap* in the initial belief b^0 of the game induced by the approximation defined as $\text{gap}(\hat{v}(b)) = \overline{v}(b) - \underline{v}(b)$ is no higher than the required precision ϵ.

Data: Game $\langle \mathcal{S}, \mathcal{A}_1, \mathcal{A}_2, \mathcal{O}, \mathcal{T}, \mathcal{R} \rangle$, initial belief b^0, discount factor γ, desired precision $\epsilon > 0$, neighborhood parameter R
Result: Approximate value function $\hat{v}$

1 Initialize $\hat{v}$
2 **while** $\mathrm{gap}(\hat{v}(b^0)) > \epsilon$ **do**
3 `Explore` $(b^0, \epsilon, R, 0)$
4 **return** $\hat{v}$
5 **procedure** `Explore` (b, ϵ, R, t)
6 $\pi_2 \leftarrow$ optimal strategy of player 2 in $[H\underline{v}](b)$
7 $(a, o) \leftarrow$ select according to forward exploration heuristic
8 **if** $\mathrm{excess}(\hat{v}(b^{a,o}_{\pi_2}), t+1) > 0$ **then**
9 `Explore` $(b^{a,o}_{\pi_2}, \epsilon, R, t+1)$
10 $\Gamma \leftarrow \Gamma \cup \{L\Gamma(b)\}$
11 $\Upsilon \leftarrow \Upsilon \cup \{U\Upsilon(b)\}$ and make $\overline{v}$ K-Lipschitz ($K = (U-L)\sqrt{|\mathcal{S}|/4}$, see Lemma 2)

Algorithm 1: HSVI algorithm for one-sided POSGs

Functions $\hat{v}$ are refined by adding new elements to their sets. These new elements result from *point-based updates* of operator H at a single belief point b.

The algorithm is initialized with $\underline{v}$ (and Γ) corresponding to the value of a uniform strategy of the first player and the upper bound $\overline{v}$ (and Υ) results from solving a perfect information refinement of the game. In every iteration, a finite set of beliefs is updated by *forward exploration* (lines 6-9). Beliefs selected by this process contribute to the fact that the gap at b^0 is not sufficiently small, and hence the approximation in these beliefs needs to be improved by applying point-based updates (lines 10 and 11). We now describe how the updates are performed, followed by the description of the forward exploration search.

Point-Based Updates

A point-based update at belief point b updates the lower and upper bound functions $\underline{v}$ and $\overline{v}$ using the optimal strategies in games $[H\underline{v}](b)$ and $[H\overline{v}](b)$. In order to prove the convergence, we require that the functions $\underline{v}$ and $\overline{v}$ are K-Lipschitz ($K = (U-L)\sqrt{|\mathcal{S}|/4}$); hence, the update has to preserve this property.

The update of $\underline{v}$ adds an α-vector corresponding to the value of a Nash equilibrium strategy of the first player in $[H\underline{v}](b)$ (denoted $L\Gamma(b)$) computed from duals of the linear program (Eqs. (3.8)-(3.9)). The value of such strategy is linear and K-Lipschitz (Lemma 2), hence the expansion of Γ by $L\Gamma(b)$ preserves K-Lipschitz continuity of $\underline{v}$.

The upper bound function $\overline{v}$ is represented by a set of points Υ. Update of upper bound adds one point, $U\Upsilon(b) = b \rightarrow [H\overline{v}](b)$, that corresponds to the evaluation of the value backup at belief b. We cannot use the linear program outlined in Eqs. (3.8)-(3.9) to compute $[H\overline{v}](b)$ directly since the function $\overline{v}$ is not represented using α-vectors. We, therefore, use a transformation presented in (Horak and Bosansky, 2016) which performs projections of beliefs to the lower envelope of $\overline{v}$ while preserving linearity of the constraints.

Adding a point to Υ can break the K-Lipschitz continuity of $\overline{v}$. We can fix this by con-

structing a piecewise linear approximation of a lower K-Lipschitz envelope:

$$\overline{v}(b) \coloneqq \inf_{b' \in \Delta(\mathcal{S})} \left\{ \overline{v}(b') + K \cdot \|b - b'\|_2 \right\} . \tag{3.14}$$

The resulting function is cK-Lipschitz when $c \geq 1$ depends on the accuracy of the approximation and can be arbitrarily close to 1.

Forward Exploration

The value backup operator H expresses the value in belief b in terms of values of subsequent beliefs $b^{a,o}_{\pi_2}$. When applied to value functions $\hat{v}$, it also propagates the approximation error. In order to minimize the gap in the initial belief b^0, we need to achieve sufficient accuracy also in beliefs encountered at a later *time*.

The forward exploration simulates a play between the players while assuming that the second player follows a strategy obtained from the application of H on the lower bound $\underline{v}$ (i.e. she is overly optimistic with her strategy). When a belief b is encountered at time t (we term such a pair (b, t) a *timed belief*) and its approximation $\hat{v}(b)$ is not sufficiently accurate, we say that it has positive *excess gap*.

Definition 2 (Excess gap)**.** *Let ϵ be the desired precision and $R > 0$ be a neighborhood parameter. Let*

$$\rho(t) = \epsilon\gamma^{-t} - \sum_{i=1}^{t} 2RK\gamma^{-i} . \tag{3.15}$$

We define the excess gap of a timed belief (b, t) as

$$\operatorname{excess}(b, t) = \operatorname{gap}(\hat{v}(b)) - \rho(t) . \tag{3.16}$$

Later we show that if all subsequent timed beliefs $(b^{a,o}_{\pi_2}, t+1)$ have negative excess gap, a point-based update at (b, t) makes the excess gap $\operatorname{excess}(b, t)$ negative as well (in fact, $\operatorname{excess}(b, t) \leq -2RK$; we then term (b, t) as *closed*). If this does not hold for the belief (b, t) currently explored, the forward exploration process selects one of the subsequent beliefs $(b^{a,o}_{\pi_2}, t+1)$ with a positive excess gap for further exploration and the process is repeated with the timed belief $(b^{a,o}_{\pi_2}, t+1)$. If all subsequent beliefs have a negative excess gap, the forward exploration process terminates. The termination is guaranteed if the neighborhood parameter R is chosen so that the sequence $\rho(t)$ is monotonically increasing in t and unbounded.

Forward Exploration Heuristic

A positive excess gap of a belief contributes to the approximation error in the initial belief. If there are multiple subsequent timed beliefs $(b^{a,o}_{\pi_2}, t+1)$ with a positive excess gap, we select the one with the highest *weighted* excess gap which is similar to the weighted excess heuristic used in (Smith and Simmons, 2004). The excess gap is weighted by both the observation probability *and* the probability that the first player plays a given action when using the strategy obtained from the upper bound value function $\overline{v}$ (i.e. according to the strategy π_1 from the game $[H\overline{v}](b)$). The action observation pair (a, o) selected in timed belief (b, t) for the further exploration maximizes $\pi_1(a) \cdot \Pr[o|a, \pi_2] \cdot \operatorname{excess}(b^{a,o}_{\pi_2}, t+1)$.

Convergence of the Algorithm

The goal of the HSVI algorithm is to make the excess gap negative in all reachable timed beliefs and thus sufficiently decrease the gap in the initial belief. Contrary to POMDPs, reachable beliefs in POSGs are influenced by the strategy of the second player – she can change her strategy to reach a belief (b', t) with a positive excess gap instead of a closed belief (b, t), while b' stays arbitrarily close to b.

We avoid this by ensuring that if (b', t) with a positive excess gap is reached by the forward exploration, it lies sufficiently far from all previously closed beliefs at time t – the minimum distance between the beliefs being controlled by the neighborhood parameter $R > 0$ from the definition of the excess gap. Unlike in POMDPs, our modified definition of the excess gap ensures that not only a closed belief itself gets a negative excess gap: all beliefs within its R-neighborhood get a negative excess gap as well (Lemma 4). The convergence of the algorithm follows since there is only a finite number of such R-separated belief points.

Lemma 4. *Let (b, t) be a timed belief and π_2 be the optimal strategy of the second player in $[H\underline{v}](b)$. If $\mathrm{excess}(b^{a,o}_{\pi_2}, t+1) \leq 0$ for all action-observation pairs (a, o) of the first player, then after performing a point-based update at b it holds that (i) $\mathrm{excess}(b, t) \leq -2RK$ and (ii) all belief points b' in the R-neighborhood of b (i.e. $\|b - b'\|_2 \leq R$) have a negative excess gap $\mathrm{excess}(b', t)$.*

Proof. We know that all beliefs $(b^{a,o}_{\pi_2}, t+1)$ reached from (b, t) when the second player plays π_2 have negative excess gap, hence for every (a, o)

$$\mathrm{gap}(\hat{v}(b^{a,o}_{\pi_2})) \leq \epsilon\gamma^{-(t+1)} - \sum_{i=1}^{t+1} 2RK\gamma^{-i} = \rho(t+1)\,. \tag{3.17}$$

From Lemma 3 we get that after performing the point-based update at b

$$[H\overline{v}](b) - [H\underline{v}](b) \leq \epsilon\gamma^{-t} - \sum_{i=1}^{t} 2RK\gamma^{-i} - 2RK \tag{3.18}$$

and hence $\mathrm{excess}(b, t) \leq -2RK$. Both updated functions $\underline{v}$ and $\overline{v}$ are K-Lipschitz, hence $\overline{v} - \underline{v}$ is $2K$-Lipschitz, which concludes the proof. □

Definition 3. *Let t be time. The set of all beliefs with negative excess gap at time t is denoted Ψ_t;*

$$\Psi_t = \{b \in \Delta(S) \mid \mathrm{gap}(\hat{v}(b)) \leq \rho(t)\}\,. \tag{3.19}$$

Theorem 3. *HSVI algorithm converges to the precision ϵ.*

Proof. In each iteration, the algorithm performs a forward exploration until it encounters a timed belief (b, t) such that all subsequent timed beliefs $(b^{a,o}_{\pi_f}, t+1)$ have a negative excess gap. Since $\mathrm{gap}(b^{a,o}_{\pi_f})$ is bounded by $U - L$, this happens after at most $t_{\max}$ steps, where

$$t_{\max} = \left\lceil \log_{1/\gamma}\left(\frac{U-L}{\epsilon} \cdot \left[1 + 2R\frac{1-\gamma^t}{\gamma^t(1-\gamma)}\right]\right)\right\rceil\,. \tag{3.20}$$

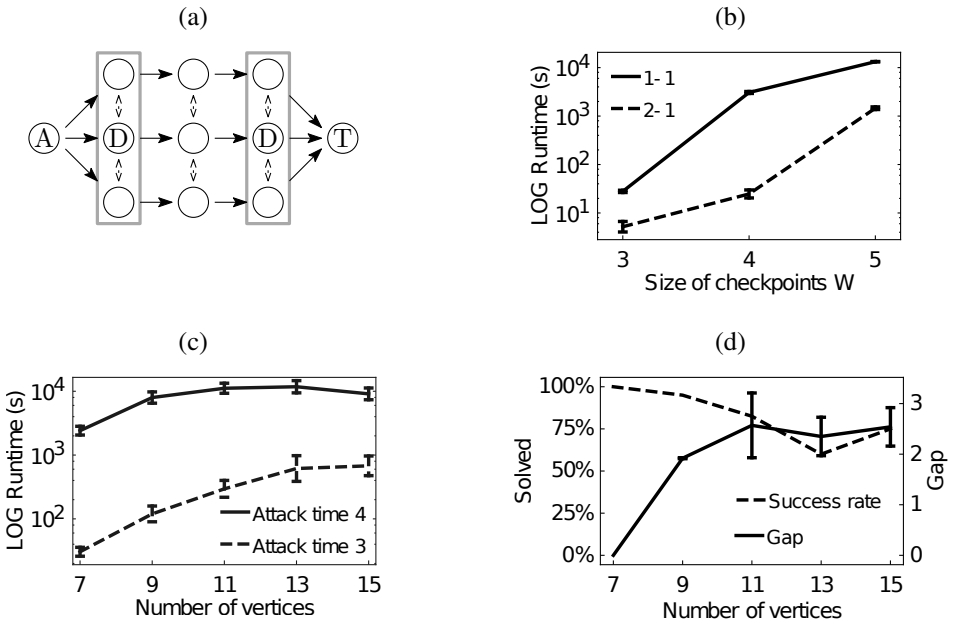

Figure 3.1: (a) Intrusion-search game: $W = 3$, configuration 1-1: A denotes initial position of the attacker, D initial positions of defender's units, T is attacker's target (b) Intrusion-search games with 2 zones, each with W vertices: Time to reach $\mathrm{gap}(\hat{v}(b^0)) \leq 1$ (c) Patrolling games played on graphs generated from $ER(0.25)$: Time to reach $\mathrm{gap}(\hat{v}(b^0)) \leq 1$ (only successfully solved instances within 10 hours) (d) Patrolling games played on graphs generated from $ER(0.25)$: Percentage of successfully solved instances with $t_\times = 4$ and the gap on failed instances after 10 hours

When the terminal timed belief (b, t) is reached, then $b \notin \Psi_t$ and all subsequent timed beliefs have negative excess gap. After performing the point-based update at (b, t), the excess gap of (b, t), as well as of all timed beliefs in the R-neighborhood of (b, t), is negative (Lemma 4) and Ψ_t is expanded. We show that the expansion of the sets $\Psi_{t'}$ guarantees that eventually $\Psi_{t'} = \Delta(\mathcal{S})$ for all times $t' \leq t_{\max}$, unless the desired precision ϵ is achieved beforehand.

The distance of b from the nearest belief b' in Ψ_t previously closed by the algorithm is at least R, since all points in the R-neighborhood of b' have a negative excess gap and thus are in Ψ_t. In each iteration, Ψ_t is expanded by at least one belief and (at least) its R-neighborhood.

The number of such expansions of timed beliefs is finite. In fact, the problem of finding maximum set of R-separated beliefs can be seen as a *hypersphere packing* (a higher dimensional version of the sphere packing (Hales, 2011)) filling the belief simplex using non-overlapping hyperspheres of radius $R/2$, since the hyperspheres do not overlap exactly when the distance between their centers is at least R. When no hypersphere can be further inserted, it means that we cannot find any belief with a positive excess gap, hence we reached the desired precision in the whole belief space. □

3.4 Experiments

We demonstrate application possibilities and scalability of our algorithm on three types of games: pursuit-evasion games (e.g., evaluated in (Horak and Bosansky, 2016)), intrusion search games (e.g., see (Bosansky et al., 2014)), and patrolling games with a discount factor (e.g., see (Vorobeychik et al., 2014)). Each player is assigned a team of units (either one or multiple units) located in vertices of a graph and he or she controls their movement on the graph. A move consists of moving the units simultaneously to vertices adjacent to their current positions, or they can wait.

The utilities are scaled so that the values of the games lies in the interval $[0, 100]$ (or $[-100, 0]$, respectively). Unless stated otherwise the discount factor is $\gamma = 0.95$ and we ran the algorithm until $\mathrm{gap}(\hat{v}(b^0)) \leq 1$.

Algorithm Settings

We initialize the value functions by solving the perfect-information refinement of the game (for $\overline{v}$) and as a best response to a uniform strategy of player 1 (for $\underline{v}$). We use standard value iteration for stochastic games, or MDPs, respectively, and terminate the initialization when either change in valuations between iterations is lower than 0.025, or $20s$ time limit has expired. The initialization time is included in the computation times of the algorithm.

Similarly to (Smith and Simmons, 2004), we adjust ϵ in each iteration using formula $\epsilon = 0.25 + \eta(\mathrm{gap}(\hat{v}(b^0)) - 0.25)$ with $\eta = 0.9$. We set the neighborhood parameter R to the largest value satisfying $\rho(t) \geq 0.25\gamma^{-t}$ for all $t \leq t_{\max}$ from the proof of Theorem 3.

Finally, we remove dominated points and vectors from sets Γ and Υ whenever their size grows by 20% to reduce the size of the linear programs. Again, this is similar to POMDPs (Smith and Simmons, 2004).

Pursuit-Evasion Games (PEGs)

A team of centrally controlled pursuers aims to locate and capture the evader, and receive the utility of +100; the evader aims for the opposite. We consider $3 \times N$ grid graphs (we vary the number of columns N), two pursuing units start in top left positions, the evader starts in bottom right corner. Our algorithm achieves similar scalability as the existing algorithm designed specifically for one-sided PEGs (Horak and Bosansky, 2016) and displays exponential dependence of the runtime on the width of the grid N. The game with $N = 3$ was solved in 9s on average, the game with $N = 6$ took 3.5 hours to be solved to the gap 1. Sizes of the games range from 143 states and 2671 transitions to 1299 states and 34807 transitions.

Search Games

In search games that model intrusion, the defender patrols checkpoint zones (see Figure 3.1a, the zones are marked with box). The attacker aims to cross the graph, while not being captured by the defender. If the attacker crosses the graph unharmed, the defender receives a utility of -100. Whenever the attacker enters a node, she leaves a trace and the defender can later detect it. She can either wait for one move to conceal her presence (and clean up the trace), or move further.

We consider games with 2 checkpoint zones with varying sizes W (i.e. width of the graph) and 2 configurations of the defending forces – with one defender in each of the checkpoint zones (denoted 1-1), and 2 defenders in the first zone while just 1 defender being in the second one (denoted 2-1). The results are shown in Figure 3.1b (with 5 runs for each parameterization, the confidence intervals mark the standard error in our graphs). The largest game ($W = 5$ and 2 defenders in the first zone) has 4656 states and 121239 transitions and can be solved within 27 minutes. This case highlights that our algorithm can solve even large games. However, a much smaller game with the configuration 1-1 (964 states and 9633 transitions) is more challenging, since the coordination problem with just 1 defender in the first zone is harder, and is solved within 3.5 hours.

Patrolling Games

In patrolling games (Basilico et al., 2009; Vorobeychik et al., 2014) the *patroller* patrols vertices of a graph by moving over the graph. The attacker decides the vertex she will attack and the time she will do so. The patroller does not know if an attack has started, however, he has a limited time (termed *attack time*, denoted $t_\times$) to reach the vertex under the attack. Otherwise, the vertex is successfully attacked and the patroller receives a negative reward associated to that vertex.

Following the setting in (Vorobeychik et al., 2014), we focus on graphs generated from Erdos-Renyi model (Newman, 2010) with parameter $p = 0.25$ (denoted $ER(0.25)$) with attack times 3 and 4 and number of vertices $|\mathcal{V}|$ ranging from 7 to 15. Each instance with attack time $t_\times = 3$ was solved by our algorithm in less than 12 minutes (see Figure 3.1c). This result generally outperforms the computation times reported for tailored algorithm for solving discounted patrolling games (Vorobeychik et al., 2014). For attack time $t_\times = 4$, however, some number of instances failed to reach the precision $\mathrm{gap}(\hat{v}(b^0)) \leq 1$ within the time limit of 10 hours. For the most difficult setting, $|\mathcal{V}| = 13$, the algorithm reached desired precision in 60% of instances (see Figure 3.1d). For unsolved instances, mean $\mathrm{gap}(\hat{v}(b^0))$ after the cutoff after 10 hours is however reasonably small (also depicted in Figure 3.1d, see the solid line and right y-axes). The results include games with up to 856 states and 6409 transitions.

3.5 One-Sided POSGs in Network Security

In this section, we provide an example showing how one-sided partially observable stochastic games can be used to reason about optimal defense strategies in network security (Horák et al., 2017b). We focus on studying the concept of *active deception* (Underbrink, 2016) where the defender takes countermeasures against the progress of the attacker in an interactive manner.

The use of the active deception can significantly improve the security level of the network. In our approach, we use the deception to make the attacker think positive about his progress so far and make him proceed in the wrong manner. We show that we cannot, however, rely solely on the deceptive actions if we want to maximize the effectiveness of the deceptive operation. The deception is the most effective if it is *stealthy* and the attacker remains unaware that we are trying to deceive him, or, at least, if we make him uncertain about the state of deception.

As soon as the attacker realizes that we are trying to deceive him, his behavior changes significantly. He will attempt to take evasive actions in attempt to lose defender's attention

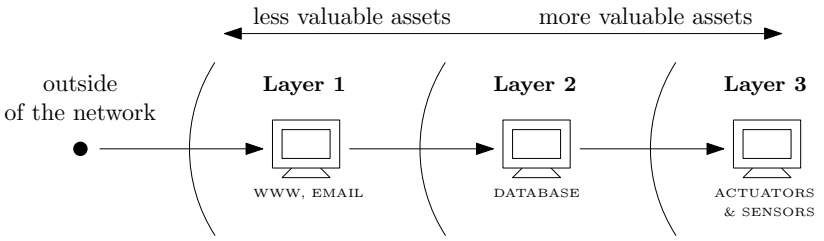

Figure 3.2: Network topology (attacker starts outside of the network and attempts to gain access to the most valuable assets in the network)

(e.g. by changing his identity), or, as a matter of last resort, he may opt to inflict severe damage based only on the information he collected so far. These decisions of the attacker make the defender's attempts to contain the attack substantially harder and should be averted.

In order to preserve the stealthy nature of the deception, it is crucial that the attacker thinks that the signals he is receiving are not *too good to be true*. The defender has to carefully manipulate attacker's belief about the deception state if he wants to make the attacker believe that no deceptive operation is taking place and keep him *engaged* in the network. We present a novel game-theoretic model of deception that allows for reasoning about attacker's beliefs and devising strong defensive strategies based upon ideas of active deception.

Network Topology and the Anatomy of an Attack

We study the concept of active deception using a network topology depicted in Fig. 3.2. We use it as an abstraction of a multilayer network which is commonly adopted in critical network operations, such as power plants or production facilities (Kuipers and Fabro, 2006). Our example network consists of three layers. The outermost layer of the network (Layer 1) is directly exposed to the Internet via demilitarized zones (DMZs) and provides less sensitive services that are used to communicate with the customers and business partners, such as web or email servers.

More critical assets are located in the deeper layers of the network. In our case, the second layer consists of data stores containing confidential data the loss of which may have a severe impact on the company. The third layer is the most critical one since it provides an access to physical devices, such as actuators and sensors, the integrity of which is absolutely essential for the secure operation of the facility. Breach of assets in the Layer 3 of the network may even pose a risk of physical damage, such as in the case of the Stuxnet attack (Falliere et al., 2011; Gostev and Soumenkov, 2011).

Attack Options We assume that an attack is initiated from a computer outside of the network. The attacker attempts to take control of a system in Layer 1 and then escalate his privileges to take control of the computers located deeper in the network by compromising them (hence we refer to this action of the attacker as `compromise`). At any point, the attacker can either `wait` or leverage the current access. Apart from compromising a host in the next layer, he has two options:

The first option is to cause significant immediate damage, such as eliminating a physical device in Layer 3 (having the attacker had access to it) – we refer to this action as `take down`. Such an action surely attracts the attention of the defender and will lead to the detection

of the attacker's presence. Therefore, the attacker is forced to quit the network and possibly repeat his attack later.

The second option is to cause smaller amount of damage while attempting not to attract defender's attention. The actions the attacker can use include e.g. a stealthy exfiltration of data or a manipulation of the records in the database – for simplicity we refer to them collectively using the `exfiltrate` action. Nevertheless, this option still runs into a small risk of being detected. Moreover it runs into the risk that the defender will avert the damage resulting from this action by means of active deception and possibly even use the fact that the attacker uses the `exfiltrate` action for his benefit. This makes it critical for the attacker to understand whether he is deceived or not.

Detection System An intrusion detection system (IDS) is deployed in the network and can identify malicious actions of the attacker. This detection is not reliable. We assume that the attacker's presence is detected with probability $d_{comp} = 0.2$, if he escalates his privileges and penetrates deeper in the network using the `compromise` action. If the attacker performs stealthy exfiltration of the data (`exfiltrate` action), we detect him with probability $d_{exf} = 0.1$. We have chosen these probabilities based on a discussion with an expert, however, the model is general enough to account for any choice of these parameters.

Active Deception We assume that the passive defensive systems, such as IDS and honeypots, are already in place and we focus on the way the defender can *actively* deceive the attacker when his presence has been detected. We take an abstracted view on defender's actions to focus on the main idea of deception, however, our model is general and these actions can be refined to account for *any* actions the defender can use. In our example, he can either use a stealthy deceptive action and attempt to `engage` the attacker in the network, or he can attempt to exclude the attacker from the network (non-deceptive `block` action). We assume that the `block` action really achieves its goal and all the privileges the attacker has gets revoked and the attacker thus has to start his attack from scratch. If it were not the case and the `block` action was less powerful, blocking the attacker would have been less tempting and hence the use of deception would have been even more desirable. By engaging the attacker we attempt to anticipate the action of the attacker and minimize (or even eliminate) the damage caused by his stealthy damaging action of `exfiltrate`. Note that both of these actions can be only used once the attacker got detected – otherwise, the defender has to rely on the infrastructure of passive defensive systems.

Game Model

We analyze the active deception in the context of the network presented in Section 3.5 using a game-theoretic model. We adopt the notion of partially observable stochastic games (POSGs) and capture the interaction between the defender and the attacker using a transition system depicted in Fig. 3.3. The state space is divided into two parts. In the upper half, the presence of the attacker in the network has not yet been revealed by the IDS, therefore, the defender cannot take active countermeasures yet. Triggering an IDS alert switches the game states into the bottom part and thus gives the defender an opportunity to apply the deception.

The arrows in the diagram represent individual transitions in the game. If the attacker uses `compromise` action, he penetrates deeper in the network. If he opts for `exfiltrate`, he stays in the current layer of the network while possibly gaining access to confidential

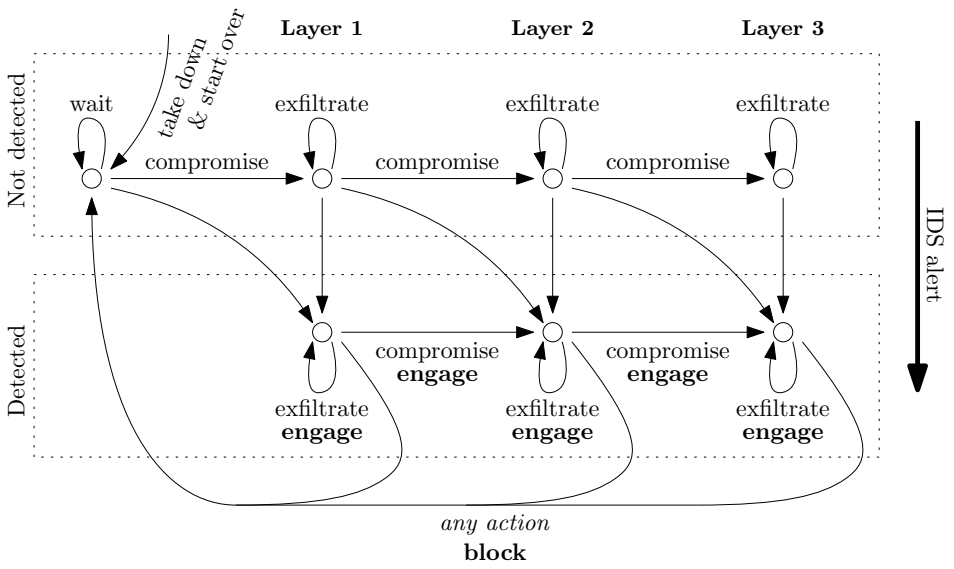

Figure 3.3: Transition system of a partially observable stochastic game representing attack on the network from Fig. 3.2. The attacker can use the `take down` action in every layer. The `wait` action of the attacker has been omitted for clarity and is always applicable.

information. And finally, he can decide to do the immediate damage by the `take down` action at any time. In such a case he gets detected and thus returns to the initial state, outside of the network. The defender can stop all this from happening by taking the `block` action (had he detected the attacker) when the defender is pushed out of the network as well. Due to the presence of the IDS in the network, there is a probability of transitioning from the upper states to lower states (d_{comp} if the attacker uses `compromise` action, d_{exf} if he opts to `exfiltrate`).

The attacker can identify the layer he has penetrated (i.e. he knows the "column" of the transition system where he is located), but he does not know whether he has been detected or not (i.e. whether the game is in the upper or lower half). The defender also does not have perfect information about the state of the attack – namely, he does not know anything about the attacker until the IDS generates an alert. After the alert is generated, however, he can get a close to perfect information about the attacker by studying the traces he has created in the system. Since the defender cannot make use of the information about the attacker in the upper states (he cannot take any active countermeasures), we can safely assume that the defender has a *perfect* information in the whole game. We therefore deal with a game with *one-sided* information – the defender is perfectly informed, while the attacker does not know whether he has been detected – and we can use techniques presented in Section 3.3 to solve the game.

Game Utilities

We associate a cost of the defender to each action the attacker performs. Since the attacker takes his actions sequentially, a sequence of costs $c_1, c_2, \ldots$ is generated, and we use discount-

		Action			
Attacker's position	Detected	Attacker	Defender	Defender's loss	
any	no	`compromise`	—	-2	$(= L_1)$
Layer i	no	`exfiltrate`	—	$15i$	$(= L_2^i)$
Layer i	no	`take down`	—	$25i$	$(= L_3^i)$
any	yes	`compromise`	`engage`	-4	$(= L_4)$
any	yes	`exfiltrate`	`engage`	-2	$(= L_5)$
Layer i	yes	`take down`	`engage`	$25i$	$(= L_6^i)$
any	yes	`compromise`	`block`	-2	$(= L_7)$
any	yes	`exfiltrate`	`block`	0	$(= L_8)$
any	yes	`take down`	`block`	0	$(= L_9)$

Table 3.1: Game rewards for the game represented in Fig. 3.3. In each time step, the players take their actions simultaneously and the loss of the defender in the current time step is determined according to their joint action.

ing to obtain the aggregated loss of the defender using the formula

$$\ell = \sum_{t=1}^{\infty} \gamma^{t-1} \cdot c_t \,. \tag{3.21}$$

The use of discounting (in our case, $\gamma = 0.95$) reflects the attacker's impatience during an attack as he does not want to wait forever to achieve his goals.

Each of the costs c_i depends on the current state of the attack (what layer the attacker has penetrated and whether he has been detected), the action the attacker performs and the counteraction of the defender (if applicable). This utility model is general enough to capture any kind of preferences of the defender. The costs we use in our case study are based on a discussion with an expert and are summarized in Table 3.1. Recall that the players take their action simultaneously and the costs thus depend on their joint action.

The `compromise` action does not cause any immediate harm to the defender and only leaks information to the defender (e.g. about an exploit used) so the loss of the defender is negative ($L_1 = L_7 = -2$). Moreover, if the defender is already aware of attacker's presence and engages him in the network, he can better understand the techniques used by the attacker and thus his loss is even lower ($L_4 = -4$).

The `exfiltrate` action is already harmful to the defender. If the defender does not take any active countermeasures, the attacker accesses confidential data which implies a significant damage to the defender. Since the assets located deeper in the network are more valuable, we account for this by defining the cost for the defender of $L_2^i = 15i$ for losing data located in the i-th layer.

If the defender realizes that he is dealing with a malicious user, he can eliminate the risk of losing sensitive data, e.g. by faking them, using the `engage` action. The attacker then receives useless data and only provides the defender with time to collect the forensic evidence. The loss of the defender is, therefore, negative ($L_5 = -2$) if the attacker exfiltrates data while being engaged. The defender can also prevent the data exfiltration by restricting attacker's access to the network (action `block`), however, by doing so, he loses the option to collect the evidence and hence the reward is $L_8 = 0$.

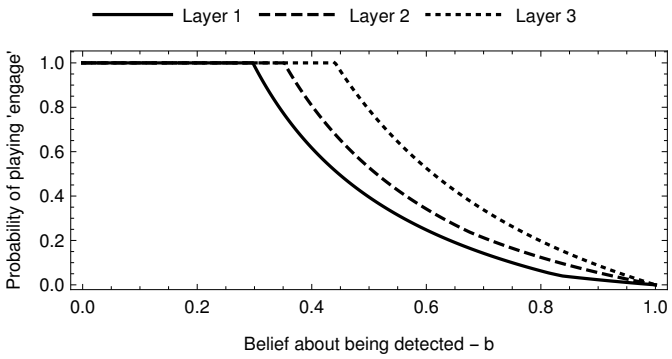

Figure 3.4: Optimal defense strategy σ for the network from Fig. 3.2. The optimal strategy of the defender is randomized and depends on the current position of the attacker (the layer he penetrated) and his belief about the detection state.

If the attacker decides to cause significant immediate damage by the `take down` action, the only option of the defender to prevent this from happening is to block the attacker (if applicable) when the loss is $L_9 = 0$. Otherwise, the cost of the defender is $L^i_3 = L^i_6 = 25i$ (when i represents the layer the attacker is in).

Optimal Defense Strategy

Once the defender succeeds in detecting the presence of the attacker, he can investigate log records to analyze past attacker's actions and estimate his belief about being detected. The defender can make use of this belief to reason about the defensive measures he should apply and design an optimal defense strategy. We are aware that in real world deployments, accurate tracking of attacker's belief need not be possible and we address this in Section 3.5.

The optimal defense strategy incurs expected long-term discounted loss of the defender of 282.154. This is a significant improvement over the common practice nowadays of attempting to block the attacker immediately after he is detected. The *always-block* strategy where the defender is restricted to play only `block` action leads to an expected loss of 429.375. It is also, however, not good to keep the attacker engaged in the deception forever (and consider only the `engage` action – we refer to this strategy as *always-engage*). Such an approach would not make the deception believable, and the attacker would rather cause the damage and forfeit his current attack attempt, than battle the deception.

We represent the optimal defensive strategy as a mapping from the current position of the attacker (i.e. the layer of the network he penetrated) and his *belief* about being detected (and thus being deceived). Since the defender has only two actions available, we express the probability of playing the `engage` action only (had he succeeded in detecting the attacker), $\sigma(i, b)$, where $i \in \{1, 2, 3\}$ is the current layer and $b \in [0, 1]$ is the attacker's belief about the detection state. The optimal defense strategy $\sigma(i, b)$ for each of the layers is depicted in Fig. 3.4.

The optimal defense strategy prescribes the defender to always keep the attacker in the network when the attacker is highly confident that he has not been detected yet. In such a situation, the attacker will opt for data exfiltration, which we can prevent, e.g. by providing him with fake data. At a certain point, the attacker starts being worried about being detected and starts considering to do immediate damage, incur a high loss to the defender and leave the network. The belief point where this starts to be the case differs based on the position of

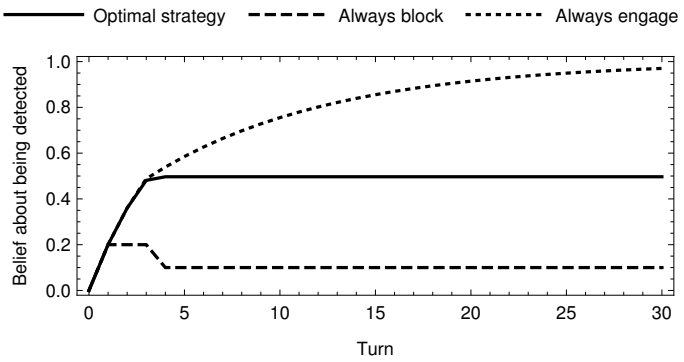

Figure 3.5: Evolution of attacker's belief over time. If we block the attacker immediately after detection, he remains highly confident that we cannot employ deceptive actions which allows him to perform long-term data exfiltration. If we always attempt to deceive the attacker by engaging him, he realizes that he likely faces a deception and decides to cause immediate damage – which is not prevented by the deceptive `engage` action.

the attacker. The deeper in the network he has penetrated, the more reluctant he is to leave the network since he regrets losing his current progress (he needed to put in more effort to get deeper into the network). After this critical belief point, the defender has to start considering excluding the attacker from the network. The more the attacker believes he is detected, the more he thinks that stealthy exfiltration of the data is no longer beneficial for him and the more likely he decides to cause immediate harm. The defender has to react to this development by decreasing the probability of keeping the attacker in the network.

To better understand the implications of the optimal defense strategy and the need for precise randomization between `engage` and `block` actions, we simulate an attack on the network and depict attacker's belief about being detected when applying the optimal, *always-block* and *always-engage* strategies.

After performing an action and getting feedback from the network, the attacker updates his belief about the detection state from b to b'. Assume that the attacker was in Layer i and he used action $a \in \{comp, exf\}$ in the last step (for `compromise`, resp. `exfiltrate`) and he didn't get blocked. In order to be detected at the current time step, the attacker could have either triggered an alert using his last action (which happens with probability $(1-b)d_a$) or the defender must have decided not to block the attacker (the probability of which is $b\sigma(i,b)$). The probability of not getting blocked equals to $1 - b(1 - \sigma(i,b))$. The updated belief of the attacker b' is then the probability of being detected given that the `block` action wasn't used:

$$b' = \frac{(1-b)d_a + b\sigma(i,b)}{1 - b(1-\sigma(i,b))} \,. \tag{3.22}$$

We assume that the the attacker conducts an attack that consists of penetrating to the deepest layer of the network using three consecutive `compromise` actions and then the exfiltration of the data. The comparison of the evolution of attacker's belief while using these three strategies is shown in Fig 3.5.

First of all, we explain why the current best practice in incident response represented by the *always-block* strategy is inferior. Whenever the attacker realizes that he has not been blocked and his access to the network has not been restricted (or limited), he *knows* that he cannot have been detected in the previous time step (since otherwise, the defender would have blocked him). His belief about being detected thus depends solely on the detection rate of IDS

– which in our experiments is $d_{comp} = 0.2$ when the attacker uses `compromise` action to penetrate deeper into the network (first 3 steps) and $d_{exf} = 0.1$ afterward. Since the attacker remains highly confident that he is not detected at each time step, he can cause a lot of harm by a long-term data exfiltration.

The *always-engage* strategy suffers from playing `engage` action even at times when the attacker becomes highly confident that he has been detected and thus realizes that the data he exfiltrates may be useless. At that point, the attacker deviates from the assumed attack plan and opts for causing immediate damage and leaving the network temporarily (before launching a new attack).

The optimal defense strategy, on the other hand, stabilizes attacker's belief about being detected at the value of $b = 0.4968$. This is the right belief where the attacker still thinks that it is worth attempting to cause a long term damage by data exfiltration, despite being vulnerable to defender's deceptive attempts.

This result draws one important conclusion about the use of deception to manipulate attacker's belief. Blocking the attacker (or even more importantly *not* blocking him) leaks a valuable piece of information to him that he can capitalize on to devise a powerful attack plan. We have to weigh the use of stealthy and non-stealthy defensive actions carefully not to alert the attacker to the use of deception. The optimal defensive strategy (unlike the *always-block* and *always-engage* strategies) achieves a belief point where no further information leaks to the attacker and the malicious effects of attacker's actions are minimized.

Engaging the Attacker

In the previous section, we have shown that the common practice in incident response deployments of blocking the attacker immediately after detection is susceptible to severe drawbacks. We proposed an alternative strategy, based on a game-theoretic model, that postpones the decision to block the attacker to minimize the long-term damage to the network. The key motivation for using this strategy is that by anticipating malicious actions of the attacker, we can minimize negative impacts of his actions and delay his progress. On the other hand, excluding the attacker from the network is only temporary. The attacker is potentially able to reenter the network and cause significant damage while he stays undetected.

Our strategy has, however, one more significant advantage since it can be leveraged to decrease false positive rates of the IDS. False detections can have a considerable negative impact on the network operations. By engaging a suspicious user in the network, we can make use of the extra time given by our deceptive strategy to identify the user, infer their objectives and take proper defense actions to reduce the impact of the network defense system on legitimate users. To this end, we can use various types of deceptive signals that do not influence legitimate users but make the progress of an attacker difficult. These signals are not explicitly captured in our example, but the model is general enough to account for them.

We conducted an experimental evaluation of our game-theoretic strategy to determine the average time between the first IDS alert and the time we decide to block the user. We evaluated our strategy against an advanced attacker who conducts the most damaging attack (a *best response* to our strategy) and we considered only the attacks where the attacker does not decide to quit the network himself. We found out that the average time between detection and access restriction is in our case 4.577 time steps. In this time window, the defender gets

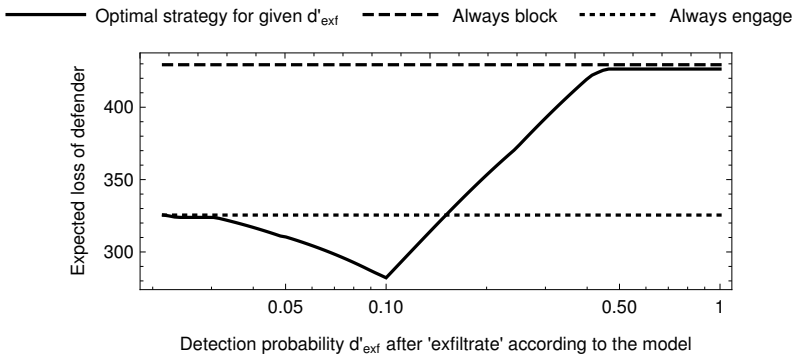

Figure 3.6: Expected loss of the defender when using a strategy originating from an inaccurate model. Strategy is computed while assuming detection probability d'_{exf} and this strategy is evaluated in a network with the detection probability of $d_{exf} = 0.1$.

additional alerts from the IDS which may help him decide about the credibility of the alert better.

Robustness of the Model

In real world setups, it need not be possible for the defender to keep track of attacker's belief accurately as a result of failing to reconstruct the exact history of the attacker and/or deficiencies in the model of the network. In this section, we focus on the impact of not knowing the exact IDS detection probabilities. We compute the optimal defensive strategy of the defender based on the model where the IDS detection rates do not match those of the real network. We manipulate the probability of triggering an IDS alert when the attacker uses `exfiltrate` action from $d_{exf} = 0.1$ to d'_{exf}, and evaluate the strategy using the original detection probability of $d_{exf} = 0.1$. Since the model is no longer accurate, the resulting strategies need not be optimal. The experimental evaluation of these strategies is shown in Fig. 3.6. The experimental results show that our strategy provides significant room for the error in the design of the model, especially if we are pessimistic about the detection rates.

CHAPTER 4

Plan Towards Dissertation

Our goal is to study partially observable stochastic games and to derive practical, scalable algorithms tailored to approximating solutions of these games. Our results on partially observable stochastic games with one-sided information (see Chapter 3 and (Horák et al., 2017a; Horak and Bosansky, 2016, 2017)) are promising and can be built upon. We have shown that the class of one-sided partially observable stochastic games with zero-sum rewards that are discounted over time is relevant for security related applications (Horák et al., 2017b). In some domains, however, these assumptions on the structure of information and the structure of rewards may be overly restrictive. To complement the message of the thesis, we plan to study models where some of the assumptions are weakened and we plan to devise efficient algorithms for these classes of models as well. Namely, we want to relax the assumption of perfectly-informed adversary (see Section 4.1), study different types of objectives (Section 4.2) and general-sum reward structures (Section 4.3). We also plan to study the applicability of our models and algorithms in practice, especially in network security (see Section 4.4).

4.1 Games with a Weaker Adversary

This work is to be done in collaboration with the group of Krishnendu Chatterjee at IST Austria

While the assumption of the perfectly informed attacker is convenient to provide safety guarantees, it might be overly conservative in some settings and prevent the defender from deriving effective defensive strategies. A perfectly informed adversary may possibly get access to system variables that cannot be observed by any of the players in reality, and make use of this extra information to render any strategy of the defender fail. We plan to overcome this deficiency of our model while keeping its safety guarantees by studying games where one player (typically the attacker) is more informed than the other one (the defender). In such a setting, the attacker no longer gets access to the full history of the game. Instead, he receives his private observation o_2^t which reveals him the action and observation (a_1^t, o_1^t) of the defender and gives him possibly additional information about the game. Similar classes of games have been studied in (Bertrand et al., 2009; Chatterjee and Doyen, 2014a) with ω-regular objectives from the perspective of qualitative determinacy.

We conjecture that the discounted-sum variant of these games can be reduced to one-sided partially observable stochastic games over the set of attacker's beliefs $\Delta(\mathcal{S})$, and the belief over attacker's beliefs $b \in \Delta(\Delta(\mathcal{S}))$ is thus a sufficient statistic to play the game optimally. We plan to study this class of games and establish this result about the sufficiency of belief over beliefs $b \in \Delta(\Delta(\mathcal{S}))$ formally. Note that our current solution techniques are, however, not sufficient to deal with continuous state spaces and novel algorithms thus need to be proposed. One of the options is to consider discretization of the continuous state space of $\Delta(\mathcal{S})$, such as in the case of grid-based POMDP solution techniques seen in (Bonet, 2002; Bonet and Geffner, 2009). The application of these techniques to the game-theoretic setting, however, introduces new challenges that need to be addressed.

4.2 Undiscounted and ω-Regular Objectives

This work is to be done in collaboration with the group of Krishnendu Chatterjee at IST Austria

Our algorithm presented in (Horák et al., 2017a) is well suited for solving one-sided partially observable stochastic games with the discounted-sum objective. The proof technique we used to show the correctness of our algorithm relies on the fact that the value backup operator used to improve the value functions over time is a γ-contraction (for $\gamma < 1$). As discussed in Section 2.2, the use of discounting is not well-suited for some of the problems arising in practice. The undiscounted-sum objectives, however, typically do not lead to contractive value backup operators and we cannot apply our algorithm in a straightforward manner.

Firstly, we want to study games with the finite-horizon undiscounted-sum objective. The point-based approximate algorithm is no longer applicable since the strategies of the players are not stationary in general (the players adapt to the number of stages remaining until the finite-horizon bound T). We observe that there is a straightforward reduction of finite-horizon undiscounted-sum problems to time-expanded problems with discounted-sum objective and infinite horizon – which allows us to apply the algorithms such as heuristic search value iteration (Smith and Simmons, 2012; Horák et al., 2017a). This reduction, however, increases the state space considerably and it is impractical for problems with large horizon T. We plan to focus on devising more practical algorithms to solve the problems with finite-horizon. The interest in the finite-horizon undiscounted-sum objective is also motivated by (Chatterjee et al., 2016) where the exact, exhaustive finite-horizon value iteration was used as a subroutine in the main algorithm proposed by the authors.

Secondly, we want to focus on designing practical algorithms for computation of values of games with reachability and safety objectives, which is important for the verification community. While we cannot hope for an efficient algorithm that can solve every instance of the game (see negative results in (Hansen et al., 2009)), our goal is to design an algorithm that would scale well to problems arising in practice.

4.3 General-Sum Games and Stackelberg Equilibrium

Currently, the algorithm we proposed assumes zero-sum rewards – that means that the objective of the attacker is to harm the defender as much as he can. While this accounts for the most

conservative scenario from the perspective of the defender, intentions of the attackers in the real world are typically different. They have their own, individual reward function $\mathcal{R}_2$ which can be even completely independent of the reward function $\mathcal{R}_1$ of the defender. We can use the preferences of the attacker to derive a more effective defensive strategy that capitalizes on the fact that the attacker will not opt for undesirable outcomes for the defender unless these outcomes are compatible with attacker's goals.

Such a reasoning is characterized using the game-theoretic concept of Stackelberg equilibrium (von Stackelberg et al., 1952). The defender acts as a leader who decides a strategy – we assume that this strategy is observable for the attacker. The attacker (the follower) follows by choosing a strategy that achieves the best utility from his perspective (i.e. according to attacker's reward function $\mathcal{R}_2$) against the strategy of the defender. A common assumption in the literature on Stackelberg equilibrium is the concept of Strong Stackelberg equilibrium (SSE) where the attacker breaks ties in favor of the defender if he has multiple strategies that achieve the same, optimal payoff (Leitmann, 1978; von Stengel and Zamir, 2004). Models based on the concept of the Stackelberg equilibrium are often used to reason about incentives of attackers in security related applications (Tambe, 2011; Fang et al., 2015; Durkota et al., 2015) and extending the concept towards partially observable stochastic games is thus of great relevance.

Our preliminary results for the case where the player with partial observability is the leader show, that the value of each strategy of the leader is linear in the belief (counterpart of Lemma 1 in Chapter 3). Therefore we can devise a convex value function representing the value of optimal strategy of the leader based on the belief b^0 about the initial state of the game. We plan to investigate whether this positive result about the structure of the solution allows us to devise an algorithm for solving general-sum games, similar to the one we proposed for zero-sum games (Horák et al., 2017a). The existence of such an algorithm will enlarge the scope of the applicability of our concepts and algorithms to a wider range of problems arising in security.

4.4 Applications in Network Security

This work is to be done in collaboration with Quanyan Zhu from New York University

Cyber deception presents a novel technique to defend computer networks. The essential component of a successful defensive strategy based on cyber deception is to understand and leverage the uncertainty the attacker is facing when conducting an attack. Based on that, we are able to manipulate his beliefs over time and prevent the attacker from accomplishing his goals. Most of the current work on cyber deception based on game-theoretic models rely on simple one-shot interactions where the defender is supposed to deploy a passive infrastructure to deter the attacker and this infrastructure remains fixed over the period of an attack. We argue that a more interactive approach where the defender reacts to the actions of the attacker can result in better defensive strategies. To this end, it is crucial to study the cyber deception in a sequential setting. Our model is well suited to analyzing this type of deception since it accounts both for the sequential nature of an attack as well as for uncertainties of one of the players. Our preliminary results (Horák et al., 2017b) show that applying interactive cyber

deception based on the model of one-sided partially observable stochastic games can increase the security level of the network significantly.

Our algorithm has, however, only a limited applicability at the moment as it is able to find near-optimal strategies to defend only networks of toy size. In order to apply our techniques in practice, we need to study the possibilities to improve the scalability of our algorithm. This includes the use of better heuristics (both domain-independent and domain-dependent) or factorization of the networks to smaller parts that are easier to solve and the solution of which can help us deriving a near-optimal solution for the original, unfactored problem.

Bibliography

Amato, C., Bernstein, D. S., and Zilberstein, S. (2010). Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Autonomous Agents and Multi-Agent Systems*, 21(3):293–320.

Baier, C., Katoen, J.-P., and Larsen, K. G. (2008). *Principles of model checking*. MIT press.

Basilico, N., Gatti, N., and Amigoni, F. (2009). Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 57–64.

Basilico, N., Nittis, G. D., and Gatti, N. (2016). A Security Game Combining Patrolling and Alarm–Triggered Responses Under Spatial and Detection Uncertainties. In *AAAI Conference on Artificial Intelligence*.

Bernstein, D. S., Amato, C., Hansen, E. A., and Zilberstein, S. (2009). Policy iteration for decentralized control of Markov decision processes. *Journal of Artificial Intelligence Research*, 34(1):89.

Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. (2002). The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840.

Bertrand, N., Genest, B., and Gimbert, H. (2009). Qualitative determinacy and decidability of stochastic games with signals. In *Logic In Computer Science, 2009. LICS'09. 24th Annual IEEE Symposium on*, pages 319–328. IEEE.

Bonet, B. (2002). An e-optimal grid-based algorithm for partially observable markov decision processes. In *Proc. of the 19th Int. Conf. on Machine Learning (ICML-02)*.

Bonet, B. and Geffner, H. (2009). Solving pomdps: Rtdp-bel vs. point-based algorithms. In *IJCAI*, pages 1641–1646. Pasadena CA.

Bosansky, B., Kiekintveld, C., Lisy, V., and Pechoucek, M. (2014). An Exact Double-Oracle Algorithm for Zero-Sum Extensive-Form Games with Imperfect Information. *Journal of Artificial Intelligence Research*, 51:829–866.

Bowling, M., Burch, N., Johanson, M., and Tammelin, O. (2015). Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149.

Chatterjee, K., Chmelík, M., Gupta, R., and Kanodia, A. (2016). Optimal cost almost-sure reachability in pomdps. *Artificial Intelligence*, 234:26–48.

Chatterjee, K. and Doyen, L. (2014a). Games with a weak adversary. In *International Colloquium on Automata, Languages, and Programming*, pages 110–121. Springer.

Chatterjee, K. and Doyen, L. (2014b). Partial-observation stochastic games: How to win when belief fails. *ACM Transactions on Computational Logic (TOCL)*, 15(2):16.

Ciesielski, K. (2007). On Stefan Banach and some of his results. *Banach Journal of Mathematical Analysis*, 1(1):1–10.

De Alfaro, L., Henzinger, T. A., and Kupferman, O. (2007). Concurrent reachability games. *Theoretical Computer Science*, 386(3):188–217.

Degorre, A., Doyen, L., Gentilini, R., Raskin, J.-F., and Toruńczyk, S. (2010). Energy and mean-payoff games with imperfect information. In *International Workshop on Computer Science Logic*, pages 260–274. Springer.

Durkota, K., Lisý, V., Bošanský, B., and Kiekintveld, C. (2015). *Approximate Solutions for Attack Graph Games with Imperfect Information*, pages 228–249. Springer International Publishing, Cham.

Falliere, N., Murchu, L. O., and Chien, E. (2011). W32. stuxnet dossier. *White paper, Symantec Corp., Security Response*, 5(6).

Fang, F., Nguyen, T. H., Pickles, R., Lam, W. Y., Clements, G. R., An, B., Singh, A., Tambe, M., and Lemieux, A. (2016). Deploying PAWS: Field optimization of the protection assistant for wildlife security. In *Proceedings of the Thirtieth Conference on Artificial Intelligence (AAAI)*, pages 3966–3973.

Fang, F., Stone, P., and Tambe, M. (2015). When Security Games Go Green: Designing Defender Strategies to Prevent Poaching and Illegal Fishing. In *In Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI)*.

Filar, J. and Vrieze, K. (1997). *Competitive Markov Decision Processes*. Springer-Verlag.

Goldsmith, J. and Mundhenk, M. (2008). Competition adds complexity. In *Advances in Neural Information Processing Systems*, pages 561–568.

Gostev, A. and Soumenkov, I. (2011). Stuxnet/Duqu: The evolution of drivers. *Online: http://www. securelist. com/en/analysis/204792208/Stuxnet Duqu The Evolution of Drivers*.

Hales, T. C. (2011). Historical overview of the Kepler conjecture. In *The Kepler Conjecture*, pages 65–82. Springer.

Hansen, E. A., Bernstein, D. S., and Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715.

Hansen, K. A., Koucky, M., and Miltersen, P. B. (2009). Winning concurrent reachability games requires doubly-exponential patience. In *Logic In Computer Science, 2009. LICS'09. 24th Annual IEEE Symposium on*, pages 332–341. IEEE.

Horak, K. and Bosansky, B. (2016). A Point-Based Approximate Algorithm for One-Sided Partially Observable Pursuit-Evasion Games. In *Decision and Game Theory for Security*.

Horak, K. and Bosansky, B. (2017). Dynamic Programming for One-Sided Partially Observable Pursuit-Evasion Games. In *Proceeding of the International Conference on Agents and Artificial Intelligence (ICAART-17)*.

Horák, K., Bošanský, B., and Pěchouček, M. (2017a). Heuristic search value iteration for one-sided partially observable stochastic games. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*.

Horák, K., Zhu, Q., and Bošanský (2017b). Manipulating Adversary's Belief: A Dynamic Game Approach to Deception by Design for Proactive Network Security.

Ibsen-Jensen, R. (2013). Strategy complexity of two-player, zero-sum games.

Kiekintveld, C., Jain, M., Tsai, J., Pita, J., Ordóñez, F., and Tambe, M. (2009). Computing optimal randomized resource allocations for massive security games. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 689–696.

Koller, D., Megiddo, N., and von Stengel, B. (1996). Efficient computation of equilibria for extensive two-person games. *Games and economic behavior*, 14(2):247–259.

Kolobov, A., Mausam, Weld, D. S., and Geffner, H. (2011). Heuristic Search for Generalized Stochastic Shortest Path MDPs.

Kuipers, D. and Fabro, M. (2006). *Control systems cyber security: Defense in depth strategies*. United States. Department of Energy.

Kumar, A. and Zilberstein, S. (2009). Dynamic programming approximations for partially observable stochastic games.

Kumar, A., Zilberstein, S., and Toussaint, M. (2015). Probabilistic Inference Techniques for Scalable Multiagent Decision Making. *J. Artif. Intell. Res.(JAIR)*, 53:223–270.

Leitmann, G. (1978). On generalized stackelberg strategies. *Journal of optimization theory and applications*, 26(4):637–643.

Littman, M. L. (1994). The witness algorithm: Solving partially observable markov decision processes. *Brown University, Providence, RI*.

Madani, O., Hanks, S., and Condon, A. (1999). On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *AAAI/IAAI*, pages 541–548.

McEneaney, W. M. (2004). Some classes of imperfect information finite state-space stochastic games with finite-dimensional solutions. *Applied Mathematics and Optimization*, 50(2):87–118.

Monahan, G. E. (1982). State of the art—a survey of partially observable markov decision processes: theory, models, and algorithms. *Management Science*, 28(1):1–16.

Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., and Bowling, M. (2017). Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*.

Nash, J. (1951). Non-cooperative games. *Annals of mathematics*, pages 286–295.

Newman, M. (2010). *Networks: an introduction*. Oxford university press.

Nikaido, H. (1954). On von Neumann's minimax theorem. *Pacific J. Math.*, 4(1):65–72.

Papadimitriou, C. H. and Tsitsiklis, J. N. (1987). The complexity of Markov decision processes. *Mathematics of operations research*, 12(3):441–450.

Pineau, J., Gordon, G., and Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, volume 3, pages 1025–1032.

Pita, J., Jain, M., Marecki, J., Ordóñez, F., Portway, C., Tambe, M., Western, C., Paruchuri, P., and Kraus, S. (2008). Deployed ARMOR protection: the application of a game theoretic model for security at the Los Angeles International Airport. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, pages 125–132.

Rasmusen, E. and Blackwell, B. (1994). Games and information. *Cambridge, MA*, 15.

Sandholm, T. (2015). Steering evolution strategically: Computational game theory and opponent exploitation for treatment planning, drug design, and synthetic biology. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., Direnzo, J., Meyer, G., Baldwin, C. W., Maule, B. J., and Meyer, G. R. (2012). PROTECT : A Deployed Game Theoretic System to Protect the Ports of the United States. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 13–20.

Shoham, Y. and Leyton-Brown, K. (2008). *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press.

Shubik, M. (1984). *Game theory in the social sciences*. JSTOR.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Smallwood, R. D. and Sondik, E. J. (1973). The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5):1071–1088.

Smith, T. and Simmons, R. (2004). Heuristic search value iteration for POMDPs. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 520–527. AUAI Press.

Smith, T. and Simmons, R. (2012). Point-based POMDP algorithms: Improved analysis and implementation. *arXiv preprint arXiv:1207.1412*.

Tambe, M. (2011). *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press.

Underbrink, A. (2016). Effective Cyber Deception. In *Cyber Deception*, pages 117–149. Springer.

Vanek, O., Yin, Z., Jain, M., Bosansky, B., Tambe, M., and Pechoucek, M. (2012). Game-theoretic Resource Allocation for Malicious Packet Detection in Computer Networks. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 902–915.

von Neumann, J. (1928). Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320.

von Stackelberg, H. et al. (1952). Theory of the market economy.

von Stengel, B. and Zamir, S. (2004). Leadership with commitment to mixed strategies.

Vorobeychik, Y., An, B., Tambe, M., and Singh, S. P. (2014). Computing Solutions in Infinite-Horizon Discounted Adversarial Patrolling Games. In *24th International Conference on Automated Planning and Scheduling (ICAPS)*.

Zhang, N. L. and Liu, W. (1996). Planning in stochastic domains: Problem characteristics and approximation. Technical report, Technical Report HKUST-CS96-31, Hong Kong University of Science and Technology.