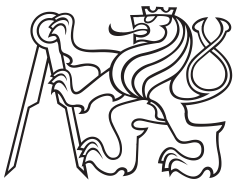
Czech Technical University in Prague
Faculty of Electrical Engineering

Doctoral Thesis

December 2019

Karel Horák



Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Science and Engineering

Doctoral Thesis

Karel Horák

Scalable Algorithms for Solving Stochastic Games with Limited Partial Observability

Ph.D. programme & Branch of study:

(P2612) Electrical Engineering and Information Technology
(2612V025) Information Science and Computer Engineering

Supervisor:

Mgr. Branislav Bošanský, Ph.D.

Supervisor-Specialist:

prof. Dr. Michal Pěchouček, M.Sc.

December 2019

Dedicated to my parents. Thank you for your support and patience.

Acknowledgments

I would like to thank all my colleagues and co-authors who made this thesis possible. Special thanks belong to my supervisor Branislav Bošanský and supervisor specialist prof. Michal Pěchouček for their support and guidance throughout the years I spent as a Ph.D. student. Furthermore, I would like to thank prof. Krishnendu Chatterjee and prof. Quanyan Zhu for the inspiring discussions during the time of my visits.

The research presented in this thesis was supported by the Czech Science Foundation (grant no. 15-23235S and 19-24384Y), by the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16 019/0000765 "Research Center for Informatics", by the Grant Agency of the Czech Technical University in Prague (grant no. SGS16/235/OHK3/3T/13) and by the U.S. Army Combat Capabilities Development Command Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Combat Capabilities Development Command Army Research Laboratory or the U.S. Government.

Abstract

Partially observable stochastic games (POSGs) represent a very general class of models that can be used to reason about sequential decision making in the presence of adversaries. In POSGs, each agent is uncertain about the state of the environment, as well as about the actions of other agents and the information they have. The cost for this generality is, however, the intractability of solving POSGs. This intractability can be attributed to the problem of *nested beliefs* as the players have to reason not only about their beliefs, but also about the beliefs other players have about the environment, about beliefs of other agents, and so on.

In this thesis, we study subclasses of POSGs where this type of nested reasoning is not necessary. First, inspired by security applications, we study a class of *one-sided POSGs*. We model a competition between two players that lasts over an infinite period of time. One of the players (typically the defender) has imperfect information, while the adversary (typically the attacker) is perfectly informed. In the zero-sum case, this kind of reasoning allows us to obtain robust strategies for the defender and provide guarantees even for settings where the attacker is less informed. We provide a scalable algorithm for solving one-sided POSGs that is inspired by techniques from the domain of POMDPs, and that provably computes an ε-approximation of the solution of the game. In some classes of games, however, the only way to win is to reason about the uncertainty of the adversary. To this end, we provide a model of POSGs with public observations that allows for reasoning about games where both players are imperfectly informed, but no player can use his private information to refine his belief. We establish structural properties of the solution of this class of games, and we also extend the algorithm for solving one-sided POSGs to this setting.

We also study practical applications of the proposed methods. Namely, we apply the model of one-sided POSGs to reason about a variant of lateral movement problem known in cybersecurity. The attacker sequentially acquires resources in the network in an attempt to reach his goal. On the other hand, the defender aims to prevent (or at least harden) the progress of the attacker by deploying honeypots into the network. In our model, the attacker can control an arbitrary subset of resources. Hence the state space of the game is exponential, and the exact method of solving one-sided POSGs suffers from the *curse of dimensionality*. To address this issue, we present a method to represent the information of the players compactly, and we show that the algorithmic results for one-sided POSGs can be extended to use this representation. While by using the compact representation we lose theoretical guarantees for finding near-optimal solutions, the compact representation allows us to scale to significantly larger instances.

Finally, we also discuss the problem of applying the heuristic search value iteration algorithm to Goal-POMDPs. We illuminate the key challenges of applying the algorithm in the undiscounted setting, and we design a novel algorithm, Goal-HSVI, that provably converges to the ε-optimal solution, and outperforms the prior approach RTDP-Bel.

Keywords algorithmic game theory, partially observable stochastic games, cybersecurity

Abstrakt

Částečně pozorovatelné stochastické hry jsou velmi obecným modelem uvažování o sekvenčním rozhodování agentů v adversariálních doménách. Každý z agentů má nejistotu ohledně stavu prostředí a také ohledně akcí ostatních agentů a informací jimi získanými. Obecnost tohoto modelu má za následek vysokou výpočetní složitost, kterou můžeme přisuzovat tomu, že agenti musí uvažovat nejen o možném stavu světa, ale také o představách ostatních agentů o světě, jejich představách o představách ostatních, atd.

V práci se zaměřujeme na podtřídy částečně pozorovatelných stochastických her, kde takovýto typ uvažování není potřebný. Nejprve na základě inspirace z domén spojených s bezpečností studujeme třídu jednostranně pozorovatelných stochastických her. Tato třída modeluje soupeření mezi dvěma hráči, které trvá po nekonečně dlouhou dobu. Jeden z hráčů (typicky obránce) má neúplné informace o průběhu hry, zatímco protihráč (typicky útočník) hru pozoruje perfektně. Ve hrách s nulovým součtem nám tento způsob uvažování umožňuje získat robustní strategie pro obránce, které mu poskytují garance i v situaci, kdy je útočník méně informovaný. Pro tuto třídu her navrhujeme škálovatelný algoritmus, který je inspirovaný technikami pro řešení částečně pozorovatelných Markovských rozhodovacích procesů a který poskytuje garance nalezení ε-optimálního řešení. Některé hry nicméně můžeme vyhrát pouze tehdy, pokud uvažujeme o nejistotě našeho soupeře. Proto třídu jednostranně pozorovatelných stochastických her dále zobecňujeme na třídu her, kde mají oba hráči neúplnou informaci, ale nemohou využít své soukromé znalosti pro zpřesnění svých představ. V práci se věnujeme jak strukturálním vlastnostem řešení této třídy her, ale také ukazujeme, že se algoritmus navržený pro jednostranně pozorovatelné hry dá zobecnit i na řešení této třídy her.

Práce se také věnuje praktickým aplikacím navrhovaných metod. Konkrétně využíváme model jednostranně pozorovatelných stochastických her pro analýzu problému, který je v síťové bezpečnosti známý pod pojmem *lateral movement*. Poté, co útočník získá přístup do sítě, se snaží dosáhnout svého cíle tím, že sekvenčně získává přístup k dalším zdrojům v síti. Obránce se mu snaží postup ztížit tím, že do sítě přidává tzv. *honeypoty*, které dokáží postup útočníka detekovat a zpomalit. V našem modelu může v každém okamžiku útočník kontrolovat libovolnou podmnožinu síťových zdrojů. Stavový prostor hry je proto exponenciální ve velikosti sítě, a exaktní metoda řešení jednostranně pozorovatelných stochastických her proto špatně škáluje. V práci navrhujeme tento problém vyřešit pomocí kompaktní reprezentace informace, kterou má obránce k dispozici, a ukazujeme, že tento přístup může výrazně vylepšit škálovatelnost algoritmu.

Dále se také věnujeme aplikaci algoritmu HSVI v Goal-POMDP. Na příkladech demonstrujeme základní problémy tohoto algoritmu v doménách, kde odměny nejsou diskontované. Na základě těchto poznatků navrhujeme algoritmus Goal-HSVI, který garantovaně konverguje k ε-optimálnímu řešení a překonává předchozí přístup RTDP-Bel.

Klíčová slova algoritmická teorie her, částečně pozorovatelné stochastické hry, síťová bezpečnost

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 the actions of other decision-makers, the optimality of the decisions we make becomes critical. This is especially true in applications related to security, where the attackers will try to identify weaknesses of our decisions and aim at exploiting these weaknesses in their favor. If we want to avoid falling victim to an attack performed by an advanced adversary, it is important to understand strategic interactions in the environment and to anticipate future actions of our opponent.

Understanding strategic interactions between adversarial parties in complex domains is, however, challenging. To this end, we use the mathematical framework of game theory to quantify the risks related, e.g., 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 many fields including biology [Sandholm, 2015], social sciences [Shubik, 1984] and of course also above mentioned 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 [Vaněk et al., 2012; Durkota et al., 2015; Kiekintveld et al., 2015; Cai et al., 2016; Nguyen et al., 2017], wildlife protection [Yang et al., 2014; Fang et al., 2015, 2016; Wang et al., 2019] or patrolling [Basilico et al., 2009a; 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 scalable algorithms for the computation of optimal or near-optimal strategies.

To understand the class of games we are interested in, let us consider a cyber-security scenario where so-called *advanced persistent threats* (or APTs) [Virvilis and Gritzalis,

2013; Mandiant Intelligence Center, 2013] are involved. APTs are highly sophisticated attackers that usually have access to considerable financial resources, and often they are even state-sponsored. Their activity is often long-lived and targeted. Motivated by this setting we can characterize the class of the games to consider. First, we need to reason about games with a long or even *infinite* horizon that involve *competition* between the attackers and the defenders. Second, players have only *imperfect information* about the course of the game (e.g., in cases of successful APT attacks, the defending side had been unaware of the infection for several years [Virvilis and Gritzalis, 2013, Table 1]). Third, we need to consider models that are *dynamic* in the sense that the state of the game can evolve (e.g., the attacker acquires additional privilege) and the players can react, e.g., to the progress of the attack or defense measures used.

Partially observable stochastic games (POSGs) [Hansen et al., 2004] represent a very general model that satisfies the above-mentioned requirements. Here, the game is played on a set of states, while the players use their actions to influence the transitions between the states. Each transition yields an individual reward for each of the players, and each of the players is aiming to maximize their rewards. Importantly, the players do not directly observe neither the state of the game, nor how other players are acting in the game, and what observations of the game they make.

While a single-agent version of POSGs with perfect information (i.e., Markov decision processes) can be solved in polynomial time [Papadimitriou and Tsitsiklis, 1987], the introduction of multiple competing agents and the imperfect information increases the complexity significantly. Goldsmith and Mundhenk [2008] have shown that the decision problem associated with the existence of a "good" strategy in partially observable stochastic games is $\mathrm{NEXP}^{\mathrm{NP}}$-complete, and several problems associated to POSGs are even undecidable [Madani et al., 1999].

Given these negative complexity results, the absence of *scalable* algorithms to solve the general class of partially observable stochastic games is not surprising. Hansen et al. [2004] propose a dynamic programming algorithm to incrementally generate non-dominated deterministic policies to play the game. Upon generating the set of required policies for the game, a normal-form representation of the game is formed and solved to obtain optimal policies. The limiting factors of such an approach are apparent as the number of policies grows doubly-exponentially in the horizon of the game (each policy considered assigns an action to each of the exponentially many observation histories of a player). Clearly, such an approach cannot be used to solve games with infinite horizon that we need and can be applied only to games with very short horizon in practice (≈ 4 for a game of a trivial size). Kumar and Zilberstein [2009] improved upon the scalability of the algorithm of Hansen et al. by employing a more aggressive pruning of considered policies—however, at the cost of optimality. Moreover, although these algorithms have been framed in the POSG setting, both of these approaches have only been experimentally evaluated in a setting where all the agents optimize a single *shared* reward function (a framework commonly known as *decentralized partially observable Markov decision processes*, Dec-POMDPs [Oliehoek et al., 2016]).

The formalism of Dec-POMDPs has been extensively studied in the literature, and numerous algorithms have been proposed to solve these cooperative models. Apart of the algorithm of Hansen et al. [2004], the optimal algorithms for solving *finite* horizon Dec-POMDPs include the search in the space of possible partial joint policies [Szer et al., 2005] or value-based methods over the space of probability distributions over states *and* individual histories of the agents [Dibangoye et al., 2016]. The number of joint policies grows doubly-exponentially with the horizon, while the number of possible histories of an agent is exponential in horizon. Neither of these methods can, therefore, be easily applied in the setting with infinite horizon we are interested in. The techniques for solving Dec-POMDPs with infinite horizon have also been developed and involve the use of *finite-state controllers* (FSC). While the repeated use of an exhaustive-backup procedure can yield ε-optimal solutions [Bernstein et al., 2009], practical algorithms focus on optimizing the performance of controllers of a fixed size. Methods such as iterative improvement of the performance of an FSC (Dec-BPI [Bernstein et al., 2005]) or direct computation of the optimal controller via non-linear programming [Amato et al., 2010] have been proposed to accomplish this goal. Bounding the size of the controller, however, leads to approximate solutions only.

Although Dec-POMDPs are similar to partially observable stochastic games, the transferability of results from Dec-POMDPs to the competitive game-theoretic setting is limited. The conceptual differences between Dec-POMDPs and POSGs can be illustrated on the class of policies that are required to represent the optimal solution. While the optimal solution of Dec-POMDPs can be found within deterministic policies [Oliehoek et al., 2008, Proposition 2.1], this does not hold for the game-theoretic setting where the randomization is needed.[1] As a result, e.g., the multi-agent A* algorithm of Szer et al. [2005] relying on a search in a finite space of deterministic partial joint policies clearly cannot be translated to POSGs.

Our work is inspired by the positive results on partially observable Markov decision processes (POMDPs) [Astrom, 1965; Sondik, 1978; Pineau et al., 2003; Smith and Simmons, 2004, 2005; Spaan and Vlassis, 2005; Bonet and Geffner, 2009; Somani et al., 2013]. Although still intractable from the theoretical perspective (the decision problem associated with the existence of a "good" policy is PSPACE-complete [Papadimitriou and Tsitsiklis, 1987]), the absence of the multi-agent aspect in POMDPs allowed for the design of scalable algorithms capable of solving many practical problems with infinite horizon. From the perspective of this thesis, the class of so called *point-based value iteration* methods [Pineau et al., 2003; Smith and Simmons, 2004, 2005; Spaan and Vlassis, 2005] plays a pivotal role. These methods improve upon the standard value-iteration methods for solving POMDPs [Sondik, 1978; Littman, 1996; Zhang and Zhang, 2001] by avoiding the exhaustive backup operation. Instead, each backup operation is focused on improving the utility of the agent in a *single belief* as opposed to the exhaustive backup operation that targets improving the utility of the agent everywhere. We believe that

[1] See Section 3.7 for an example of a game where Nash equilibrium exists in randomized strategies only.

the key characteristics of POMDPs that allow for the design of these efficient methods is the ability to represent the information state of the agent concisely as his beliefs (i.e., objective probability distributions over possible states) and, importantly, the ability to identify the beliefs that are about to be reached. In POMDPs, the agent is the single decision-maker who influences the transitions between the states—hence given the known probabilistic characterization of the environment, he is able to infer the future beliefs by applying a Bayesian rule. This reasoning is more complicated in multi-agent settings where the decisions made by other agents have to be taken into account.

Given the positive results on point-based methods for solving POMDPs, it is natural to ask whether these methods can be extended to adversarial POSGs. Wiggers et al. [2016] suggest that in the case of finite-horizon two-player zero-sum POSGs, the value function can be defined using joint beliefs over private histories of the players. Wiggers et al. further suggest that this representation may allow for design of point-based methods. However, the representation using private histories of the players is similar to the *sequence-form* representation of behavioral strategies in the *sequence-form linear program* formulation [Koller et al., 1996; Shoham and Leyton-Brown, 2008] that can be used to solve finite zero-sum extensive-form games. Given the scalability issues of the sequence form LP formulation (recall that the number of private histories corresponding to sequences is exponential in horizon), we believe that a coarse approximation of the proposed belief space would have been required to design a scalable algorithm even for small finite horizon—thus sacrificing optimality.

Unfortunately, it seems that the representation from [Wiggers et al., 2016] is the most compact representation of information states of the players in zero-sum POSGs with general information structure. The key challenge in reasoning about information in POSGs lies in the asymmetry of the information the players have. Due to different private observations of the environment, the players can achieve different beliefs [Hansen et al., 2004]. The reasoning about optimal decisions then requires to reason not only the private belief, but also about the possible beliefs of the adversary. This results in the problem of so-called *nested beliefs* [MacDermed, 2013] where the players need to reason about possible beliefs of the adversary, possible beliefs of the adversary about others' beliefs, and so on. The actual distribution over possible private histories proposed in [Wiggers et al., 2016] is able to capture this nested reasoning implicitly.

Since it is likely impossible to come up with a compact representation of beliefs in general POSGs (and thus design near-optimal algorithms for solving POSGs with long or even infinite horizon), we have to ask whether the POSG model is not unnecessarily general to represent practical problems. This question has been discussed in [MacDermed, 2013]. In [MacDermed, 2013], the author proposes a subclass of POSGs called *Markov games of incomplete information* (MaGII) where restrictions are imposed on the transitions between states and the private observations the players can make. Namely, each player has to be able to infer the probability distribution over possible current hidden states of the environment *and* current private observations other agents observed solely from the publicly known information and current private observation of the player. Although

MacDermed provides an algorithm for solving MaGIIs, this algorithm addresses the cooperative case (i.e., Dec-POMDPs) only.

We follow the direction of MacDermed [2013], and we focus on subclasses of POSGs that are practical from the perspective of real-world applications and allow for efficient algorithmic treatment. However, compared to this work, we focus primarily on problems where competition is involved. First, inspired by the motivational example from the domain of cybersecurity, we study a class of *one-sided partially observable stochastic games* (OS-POSGs). These are two-player zero-sum games (modeling, e.g., the competition between the defender and the attacker) where one side of the interaction (typically the attacker) is assumed to be perfectly informed. We considered that the game is played over an infinite horizon, and the players are optimizing the discounted sum of rewards. While the restriction on information structure may seem overly restrictive, we argue that it is well-suited for security applications. The exact information that is revealed to the attacker is usually not known, and the assumption that the attacker is able to observe everything is therefore sensible in critical scenarios and is made in several previous works, e.g., on patrolling [Basilico et al., 2009a; Vorobeychik et al., 2014] or pursuit evasion [Isler et al., 2005; Isler and Karnad, 2008; Amigoni and Basilico, 2012]. The theoretical aspects of the class of OS-POSGs have been studied in [Sorin, 2003] where similarities with the POMDP models have been shown. Namely, the beliefs in OS-POSGs are over states only (i.e., does not involve private histories of the adversary), and the value function defined over such beliefs is convex and can be characterized by a recursive formula. These similarities allow us to use inspiration from POMDP techniques, and to design a scalable algorithm to solve OS-POSGs. Second, inspired by works of Cole and Kocherlakota [2001] and MacDermed [2013], we generalize the class of one-sided POSGs towards games where *both* players have imperfect information, but the individual beliefs of the players are conditioned on public information only. We therefore call such a class of games *partially observable stochastic games with public observations* (PO-POSGs). For both of these classes of games, we design scalable algorithms [Horák et al., 2017a; Horák and Bošanský, 2019] that are inspired by the heuristic search value iteration approach for solving POMDPs [Smith and Simmons, 2004, 2005]. Our proposed algorithms provably converge to ε-optimal solutions of both OS-POSGs and PO-POSGs.

We also focus on applications of the proposed techniques in cybersecurity. Namely, we apply the one-sided POSG model to study active deception techniques [Horák et al., 2017b] and to design strategies against lateral movement in cyber-physical environments [Horák et al., 2019a,b]. In the second example, the state space of the game is prohibitively large (exponential in the size of considered computer networks). To this end, we propose a technique to represent the information of the imperfectly informed defender in a compact manner, and we show that the use of compact representation can significantly improve the scalability of the algorithm.

Finally, we also focus on the problem of Goal-POMDPs. These are single-agent problems where the agent aims at minimizing the total cost until the goal is reached. Unlike in the previous models considered in the thesis, the objective is not a discounted sum of rewards, but a total sum without discounting. We study the possibility of using

heuristic search value iteration algorithm to solve Goal-POMDPs. We show key challenges of applying this algorithm in undiscounted problems on examples, and based on these insights we design a novel *Goal-HSVI* algorithm that provably converges to ε-optimal solutions of Goal-POMDPs, and outperforms the prior heuristic approach RTDP-Bel.

1.1 Goals of the Thesis

Design scalable algorithms for solving practical subclasses of POSGs Solving general POSGs is a challenging problem from the computational perspective [Goldsmith and Mundhenk, 2008; Hansen et al., 2004]. To this end, we focus on subclasses of POSGs that are easier to solve, while they are still relevant for practical applications.

Inspired by applications in security, we study a subclass of *one-sided partially observable stochastic games*. In these two-player zero-sum games that last over infinite time period, one player (typically the defender) is imperfectly informed about the course of the game, while the adversary (typically the attacker) is able to observe the course of the game perfectly. This one-sided imperfect information is well-aligned with previous works on security problems (e.g., a standard assumption in patrolling [Basilico et al., 2009a] is that the attacker can observe movements of the defender), and it allows us to obtain robust strategies for the defender.

From the perspective of security domains, we want to design scalable algorithms that provide guarantees on the quality of the computed solution. We want to achieve both guarantees that the computed strategies will perform no worse than expected, as well as guarantees that the computed solution is near-optimal (i.e., we aim at computing ε-Nash equilibrium of the zero-sum game). We achieve this by providing an algorithm [Horák and Bošanský, 2016; Horák et al., 2017a] inspired by the heuristic search value iteration for POMDPs [Smith and Simmons, 2004, 2005]. We prove that the proposed algorithm converges to the value of the game, and the representation of the solution allows us to reconstruct strategies that achieve near-optimal performance.

To further emphasize the goal of scalability, we provide an approximate version of the proposed algorithm that represents the uncertainty of the defender (i.e., his belief) in a compact manner [Horák et al., 2019a,b] to significantly improve the scalability of the algorithm. Although this sacrifices the guarantees on the near-optimality of the computed solution, we still retain the guarantee that the computed value is a lower bound on the value of the considered one-sided POSG (hence, we still get worst-case type of guarantees for the defender).

As a last step, we also provide a generalization of the one-sided partially observable stochastic games model and we introduce *partially observable stochastic games with public observations* (PO-POSGs). Here, both players are allowed to be imperfectly informed, however, they have to be able to infer the belief of the adversary. We provide the analysis of structural properties of the solution of PO-POSGs, and we extend the heuristic search value iteration algorithm from one-sided POSGs to PO-POSGs [Horák and Bošanský, 2019].

Demonstrate the applicability of algorithmic results on real-world problems To ensure that the classes of games that we study in this thesis are indeed relevant for practical applications, we apply our algorithmic results to real-world problems. To this end, we apply our algorithms in the context of network security.

First, we use the model of one-sided partially observable stochastic games to reason about active deception in computer networks [Horák et al., 2017b]. Here, we provide a case study that aims at providing evidence that game-theoretic incident response strategies (i.e., the methods of how to react after we detect attacker's presence in the computer network) are needed. These strategies have to take the belief of the adversary into account, and actively try to confuse the attacker.

Second, we study the problem of *lateral movement* in cyber-physical topologies. Here, the attacker has already established his presence within the system, and he now aims at extending his presence in attempt to accomplish his goal. We provide a game-theoretic model based on one-sided POSGs [Horák et al., 2019a,b] that aims at studying the ways the defender should allocate honeypots within the network to detect and possibly delay the attacker. Importantly, however, our model considers the sequential nature of the problem and allows the defender to reallocate the honeypots based on the information the defender learns about the progress of the attacker.

1.2 Thesis Outline

In Chapter 2, we provide an overview of relevant concepts in game theory, and we provide a brief introduction to partially observable Markov decision processes. In Chapter 3, we provide a detailed discussion of one-sided partially observable stochastic games and we present the first scalable algorithm to solve this class of games. This algorithm is inspired by the heuristic search value iteration algorithm for solving POMDPs [Smith and Simmons, 2004, 2005] and is guaranteed to find ε-optimal solution of one-sided POSGs. We also provide an application of the one-sided POSGs model to study active deception in computer networks in Section 3.9. In Chapter 4, we study the compact representation of the beliefs in one-sided POSGs and we extend the heuristic search value iteration algorithm to use this representation. Furthermore, we apply these ideas to the problem of lateral movement known in cybersecurity, and we demonstrate that this compact representation can significantly improve the scalability of the algorithm. Next, in Chapter 5, we provide a generalization of the one-sided POSG model where both players are imperfectly informed about the course of the game. We provide analysis of the structural properties of the solution and we extend the heuristic search value iteration algorithm to this class of games. In Chapter 6, we study the problem of Goal-POMDPs and the applicability of the heuristic search value iteration to this class of partially observable Markov decision processes. Finally, in Chapter 7, we provide an overview of the thesis and we suggest directions for future work.

CHAPTER 2

Technical Background

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].

The behavior of the players during the game is described by *strategies*. These (behavioral) strategies map each possible information state of the player to a (possibly randomized) action the player is about to use at the given information state. Throughout this thesis, we denote such decision rules σ_i where i denotes the player that uses the strategy. A rational player i is aiming to choose a strategy such that his utility $u_i(\sigma_i, \sigma_{-i})$ is maximal when strategies σ_{-i} of other players[1] are considered.

To describe the behavior of rational players, equilibrium solution concepts have been established. Equilibrium points correspond to *strategy profiles* (i.e., vectors of strategies for each of the players) that are stable in the sense that the players are not incentivized to deviate from their strategy. One of the most famous solution concepts for characterizing rational behavior in games is the *Nash equilibrium* [Nash et al., 1950]. A strategy profile (σ_i, σ_{-i}) is in Nash equilibrium if no player i is incentivized to unilaterally deviate from his strategy σ_i, i.e., for every player i and every strategy σ_i' of player i we have $u_i(\sigma_i, \sigma_{-i}) \geq u_i(\sigma_i', \sigma_{-i})$.

From the perspective of this thesis, we do not aim to compute the exact Nash equilibrium. Instead we focus on the concept of ε-Nash equilibrium, where the improvement of utility the player can achieve by deviating from the equilibrial strategy profile is bounded by ε, i.e., $u_i(\sigma_i, \sigma_{-i}) \geq u_i(\sigma_i', \sigma_{-i}) - \varepsilon$ holds for every player i and every his strategy σ_i'.

[1] We use $-i$ to refer to all players in the game except for the player i.

2.1.1 Two-Player Zero-Sum Games

The focus of this thesis is on two-player zero-sum games (and we assume only such games in the remainder of the thesis). Such games are played by two players (denoted player 1 and player 2) where the utilities u_1 and u_2 sum to zero, i.e., $u_1(\sigma_1, \sigma_2) = -u_2(\sigma_1, \sigma_2)$ for every strategy profile (σ_1, σ_2). Observe that in this setting, each player i not only maximizes his own utility $u_i(\sigma_1, \sigma_2)$, but at the same time he minimizes the utility $u_{-i}(\sigma_1, \sigma_2)$ of the adversary. To this end, a single utility function u is sufficient to represent the goals of both players. We use u to denote the utility u_1 of the player 1, and we assume that player 1 aims at maximizing u while player 2 aims at minimizing u.

Definition 2.1 (Zero-sum game)**.** A tuple (Σ_1, Σ_2, u) where Σ_1 and Σ_2 are the sets of strategies of player 1 and player 2, respectively, and $u : \Sigma_1 \times \Sigma_2 \to \mathbb{R}$ is the utility function of player 1 is called a *zero-sum game*.

In the context of zero-sum games, the Nash equilibrium strategy profile $(\sigma_1, \sigma_2) \in \Sigma_1 \times \Sigma_2$ can be characterized by

$$u(\sigma_1, \sigma_2) \geq u(\sigma_1', \sigma_2) \quad \text{and} \quad u(\sigma_1, \sigma_2) \leq u(\sigma_1, \sigma_2') \qquad \text{for all } \sigma_1' \in \Sigma_1,\ \sigma_2' \in \Sigma_2. \tag{2.1}$$

An important characterization of Nash equilibrium in zero-sum games is provided by von Neumann's minimax theorem [von Neumann, 1928; Nikaido, 1953].

Theorem 2.1 (Minimax theorem [von Neumann, 1928] as stated by Nikaido [1953])**.** *Let Σ_1 and Σ_2 be convex compact sets of strategies, and let utility function u be continuous, quasi-concave in Σ_1 and quasi-convex in Σ_2. Then*

$$V = \max_{\sigma_1 \in \Sigma_1} \min_{\sigma_2 \in \Sigma_2} u(\sigma_1, \sigma_2) = \min_{\sigma_2 \in \Sigma_2} \max_{\sigma_1 \in \Sigma_1} u(\sigma_1, \sigma_2)\ . \tag{2.2}$$

Observe that the direct consequence of Theorem 2.1 is that a strategy profile (σ_1^*, σ_2^*) where σ_i^* are minimizers/maximizers from Equation (2.2) is a Nash equilibrium of the game (Σ_1, Σ_2, u). The utility of player 1 when strategy profile (σ_1^*, σ_2^*) is used is $u(\sigma_1^*, \sigma_2^*)$. Since (σ_1^*, σ_2^*) are maximizers/minimizers from Equation (2.2), we have

$$\begin{aligned} u(\sigma_1^*, \sigma_2^*) \leq \max_{\sigma_1 \in \Sigma_1} u(\sigma_1, \sigma_2^*) = \min_{\sigma_2 \in \Sigma_2} \max_{\sigma_1 \in \Sigma_1} u(\sigma_1, \sigma_2) = \\ = \max_{\sigma_1 \in \Sigma_1} \min_{\sigma_2 \in \Sigma_2} u(\sigma_1, \sigma_2) = \min_{\sigma_2 \in \Sigma_2} u(\sigma_1^*, \sigma_2) \leq u(\sigma_1^*, \sigma_2^*) \end{aligned} \tag{2.3}$$

and therefore we have

$$u(\sigma_1', \sigma_2^*) \leq \max_{\sigma_1 \in \Sigma_1} u(\sigma_1, \sigma_2^*) \leq u(\sigma_1^*, \sigma_2^*) \text{ and } u(\sigma_1^*, \sigma_2') \geq \min_{\sigma_2 \in \Sigma_2} u(\sigma_1^*, \sigma_2) \geq u(\sigma_1^*, \sigma_2^*) \tag{2.4}$$

which corresponds to Nash equilibrium characterization from Equation (2.1). Since the utility $V = u(\sigma_1^*, \sigma_2^*)$ of a Nash equilibrium is uniquely defined, we call it *value of the game*.

Several generalizations of von Neumann's theorem have been proposed in the literature. We provide a formulation of Sion [1958] where the set of strategies of one of the players is allowed not to be compact.

Theorem 2.2 (Sion's minimax theorem [Sion, 1958] as stated by Komiya [1988])**.** *Let Σ_1 and Σ_2 be convex sets of strategies where Σ_2 is compact, and let utility function u be*

- *upper semicontinuous and quasi-concave on Σ_1*
- *lower semicontinuous and quasi-convex on Σ_2.*

Then

$$V = \sup_{\sigma_1 \in \Sigma_1} \min_{\sigma_2 \in \Sigma_2} u(\sigma_1, \sigma_2) = \min_{\sigma_2 \in \Sigma_2} \sup_{\sigma_1 \in \Sigma_1} u(\sigma_1, \sigma_2) \ . \tag{2.5}$$

Throughout the thesis we will apply these theorems in a simplified setting where the utility function u is concave in Σ_1, convex in Σ_2 and continuous. Note that in such a case, u is also quasi-concave and upper semicontinuous on Σ_1 and quasi-convex and lower semicontinuous on Σ_2.

2.1.2 One-Shot Games

In this section, we will cover games where there is no dynamic interaction. Instead, the players act only once in the game. Such games are called *normal-form* (or *matrix*) games (NFGs) [Shoham and Leyton-Brown, 2008]. In these games, players choose from a finite set of actions, and the utility of the players is derived based on the joint action selected by the players.

Definition 2.2 (Zero-sum normal-form game)**.** A tuple (A_1, A_2, u) where A_1, A_2 are finite sets of actions of player 1 and player 2 and $u : A_1 \times A_2 \to \mathbb{R}$ is a utility function assigning payoff of the maximizing player 1 to each joint action $(a_1, a_2) \in A_1 \times A_2$ is called a *zero-sum normal form game*.

In normal form games, the players simultaneously decide upon their actions $(a_1, a_2) \in A_1 \times A_2$. In general, the equilibrium in NFGs exists only in mixed strategies. Hence the decision of the players is characterized by randomized strategies $\sigma_i \in \Delta(A_i)$, termed *mixed strategies*. The expected utility $u'(\sigma_1, \sigma_2)$ of playing a strategy profile (σ_1, σ_2) satisfies

$$u'(\sigma_1, \sigma_2) = \mathbb{E}_{a_1 \sim \sigma_1, a_2 \sim \sigma_2}[u(a_1, a_2)] = \sum_{(a_1, a_2) \in A_1 \times A_2} \sigma_1(a_1) \cdot \sigma_2(a_2) \cdot u(a_1, a_2) \ . \tag{2.6}$$

Since the strategy sets $\Sigma_i = \Delta(A_i)$ are convex compact sets, and the utility function u' is linear in both Σ_1 and Σ_2, von Neumann's minimax theorem (Theorem 2.1) applies and we can compute equilibrial strategies using the following linear program (here shown from the perspective of the maximizing player 1, but the formulation for player 2 is analogous).

$$\max_{\sigma_1, V} V \tag{2.7a}$$

$$\text{s.t. } V \leq \sum_{a_1 \in A_1} \sigma_1(a_1) \cdot u(a_1, a_2) \qquad \forall a_2 \in A_2 \tag{2.7b}$$

$$\sum_{a_1 \in A_2} \sigma_1(a_1) = 1 \tag{2.7c}$$

$$\sigma_1(a_1) \geq 0 \qquad \forall a_1 \in A_1 \tag{2.7d}$$

Here, the player 1 seeks a strategy σ_1 that maximizes the utility against *any* best response of player 2. The utility against individual best responses of player 2 is characterized by constraints (2.7b).

Bayesian games generalize normal-form games and introduce incomplete information about payoffs of the game. The payoffs of individual action profiles in the game are conditioned on joint *types* of the players. While each player knows his own private type, the type of the adversary is hidden.

Definition 2.3 (Zero-sum Bayesian game)**.** A tuple $(A_1, A_2, \Theta_1, \Theta_2, u, p)$ where

- A_1, A_2 are finite sets of actions of player 1 and player 2, respectively,
- Θ_1, Θ_2 are finite sets of private types of the players,
- $u(a_1, a_2|\theta_1, \theta_2)$ is the payoff of action profile (a_1, a_2) when the joint type of players is (θ_1, θ_2),
- $p \in \Delta(\Theta_1 \times \Theta_2)$ is the commonly known prior over joint types

is called a *zero-sum Bayesian game*.

Unlike in normal-form games, the strategy of the player i has to be conditioned on the private type $\theta_i \in \Theta_i$ that gets revealed to him at the beginning of the game. Hence the strategy in the Bayesian game is a mapping $\sigma_i : \Theta_i \to \Delta(A_i)$. Under these strategies, we can formulate the expected utility $u'(\sigma_1, \sigma_2)$ of a strategy profile (σ_1, σ_2) as an expectation over joint types and joint actions.

$$u'(\sigma_1, \sigma_2) = \sum_{(\theta_1, \theta_2) \in \Theta_1 \times \Theta_2} p(\theta_1, \theta_2) \sum_{(a_1, a_2) \in A_1 \times A_2} \sigma_1(\theta_1, a_1) \cdot \sigma_2(\theta_2, a_2) \cdot u(a_1, a_2 \mid \theta_1, \theta_2) \tag{2.8}$$

Any randomized strategy $\sigma_i : \Theta_i \to \Delta(A_i)$ in zero-sum Bayesian games can be represented as a mixture over pure strategies $\sigma_i^{\text{pure}} : \Theta_i \to A_i$. Hence, any Bayesian game can be converted to a normal-form game where the actions A_i of player i correspond to the (finite) set of his pure strategies σ_i^{pure}.

2.1.3 Stochastic Games

Stochastic games generalize the concept of Markov decision processes by introducing multiple players that jointly influence the environment. We focus on zero-sum stochastic games where competition is involved (hence also called competitive Markov decision processes in [Filar and Vrieze, 1997]). Unlike in the one-shot models discussed in the previous section, the game is played over multiple stages and the players accumulate rewards over time.

Definition 2.4 (Zero-sum stochastic game)**.** *Zero-sum stochastic game* is a tuple (S, A_1, A_2, T, R) where

- S is a finite set of states,
- A_1, A_2 are finite sets of actions of player 1 and player 2, respectively,
- $T(s' \mid s, a_1, a_2)$ is the probability to transition from s to s' when actions (a_1, a_2) are taken by the players, and
- $R(s, a_1, a_2)$ is the reward player 1 receives when actions (a_1, a_2) are applied in state s.[a]

[a] Due to the zero-sum property, $-R(s, a_1, a_2)$ is the reward of player 2.

The play in a stochastic game with initial state $s^{(1)}$ proceeds as follows. In each stage, the players recall the entire history of the game (including all past states and actions of both players) and they decide independently upon the actions to play at the current stage $t = 1, 2, \ldots$. When actions $(a_1^{(t)}, a_2^{(t)})$ are chosen at stage t, player 1 receives $r_t = R(s^{(t)}, a_1^{(t)}, a_2^{(t)})$ and the game moves to state $s^{(t+1)}$ with probability $T(s^{(t+1)} \mid s^{(t)}, a_1^{(t)}, a_2^{(t)})$. Unless specified otherwise, we assume that this process is repeated for an infinitely long period of time and the goal of the players is to optimize the infinite discounted sum of rewards Disc^γ, where

$$\mathsf{Disc}^\gamma = \sum_{t=1}^{\infty} \gamma^{t-1} r_t \tag{2.9}$$

and $\gamma \in (0, 1)$ is a constant *discount factor*. The discounted stochastic games are also known as *Shapley games* [Shapley, 1953; Filar and Vrieze, 1997].

Although the players have access to entire histories $(s^{(i)}, a_1^{(i)}, a_2^{(i)})_{i=1}^{t} s^{(t+1)}$ at the begining of the stage $t + 1$, it has been shown that the equilibrium exists in stationary strategies that depend on the current state only, i.e., $\sigma_i : S \to \Delta(A_i)$. The value of the discounted stochastic game exists [Shapley, 1953] and can be characterized using a vector $V^* \in \mathbb{R}^S$ where V_s^* is the value of the game starting from state $s \in S$. Furthermore, vector V^* of optimal values can be characterized using a fixed point equation where the effective utility is the sum of reward of player 1 in the first stage of the game and the discounted utility player 1 is able to achieve in the rest of the game.

$$V_s^* = [HV^*]_s = \max_{\sigma_1(s)\in\Delta(A_1)} \min_{\sigma_2(s)\in\Delta(A_2)} \Big[\mathbb{E}_{a_1\sim\sigma_1(s),a_2\sim\sigma_2(s)}[R(s,a_1,a_2) + \\ +\gamma \textstyle\sum_{s'\in S} T(s' \mid s,a_1,a_2)\cdot V_{s'}^*]\Big] \tag{2.10}$$

The solution of Equation (2.10) corresponds to the value of normal-form game (A_1, A_2, u_s) where $u_s(a_1,a_2) = R(s,a_1,a_2) + \gamma \sum_{s'\in S} T(s' \mid s,a_1,a_2)\cdot V_{s'}^*$. If the vector V^* of optimal values is known, we can use the linear program (2.7) to obtain strategies $(\sigma_1(s), \sigma_2(s))$. These can then be used to form equilibrial stationary strategies in discounted stochastic games.

Since the value backup operator $H : \mathbb{R}^s \to \mathbb{R}^s$ defined by Equation (2.10) is a contraction, one can approximate the vector V^* of optimal values using value iteration algorithm [Shapley, 1953; Bowling and Veloso, 2000] that provably converges to V^*. This algorithm starts with arbitrary value vector $V_0 \in \mathbb{R}^S$ and generates a sequence of vectors $\{V_i\}_{i=1}^{\infty}$ such that $V_{i+1} = HV_i$. Other methods for solving stochastic games include policy iteration methods [Rao et al., 1973; Filar and Vrieze, 1997] or reinforcement learning methods [Littman, 1994], however, their description is beyond the scope of this thesis.

2.1.4 Partially Observable Stochastic Games

Partially observable stochastic games [Hansen et al., 2004] generalize stochastic games by introducing imperfect information. Unlike in perfect-information stochastic games, the players can only observe their *private* observations and recall the actions they made. They need not, however, have access to information about the states of the game, or actions and observations of the adversary.

Definition 2.5 (Zero-sum partially observable stochastic game)**.** A *zero-sum partially observable stochastic game* is a tuple $(S, A_1, A_2, O_1, O_2, T, R, b^{\text{init}})$ where

- S is a finite set of states,
- A_1, A_2 are finite sets of actions of player 1 and player 2,
- O_1, O_2 are finite sets of observations of player 1 and player 2,
- $T(o_1, o_2, s' \mid s, a_1, a_2)$ is a probability to transition to state s' from s while generating observations (o_1, o_2) when (a_1, a_2) was played,
- $R(s, a_1, a_2)$ is the reward of player 1 when actions (a_1, a_2) are used in state s, and
- $b^{\text{init}} \in \Delta(S)$ is the initial belief.

The play in a zero-sum POSG proceeds as follows. First, the initial state $s^{(1)}$ is sampled from the probability distribution b^{init} which is commonly known by both players. In stage t, players choose independently actions $a_1^{(t)}$, $a_2^{(t)}$. As a result, player 1 receives reward $R(s^{(t)}, a_1^{(t)}, a_2^{(t)})$ and player 2 receives $-R(s^{(t)}, a_1^{(t)}, a_2^{(t)})$ due to the zero-sum property.

The rewards are assumed to be unobservable by the players during the play.[2] With probability $T(o_1^{(t)}, o_2^{(t)}, s^{(t+1)} \mid s^{(t)}, a_1^{(t)}, a_2^{(t)})$ the game moves to a state $s^{(t+1)}$, player 1 receives observation $o_1^{(t)}$, and player 2 receives observation $o_2^{(t)}$. The only information available to player i when deciding upon action in stage t is the history of his past private observations and actions, $(a_i^{(j)}, o_i^{(j)})_{j=1}^{t-1}$.

Unlike in stochastic games of perfect information, it seems that the players have to condition their decisions on the entire history of private observations and actions in POSGs. The reason for this distinction lies in the fact that the players have a different perception of the game, and hence attain different beliefs [Hansen et al., 2004]. A behavioral strategy $\sigma_i : (A_i O_i)^* \to \Delta(A_i)$ of player i in a POSG is therefore a mapping from private histories to a distribution over actions.

Structure of value function in finite-horizon zero-sum POSGs The focus of this thesis is on designing scalable value methods for solving partially observable stochastic games. To this end, it is necessary to establish a value function of POSGs. Wiggers et al. [2016] study finite-horizon version of zero-sum POSGs where the objective is to optimize undiscounted sum of rewards obtained over a finite period H, i.e., $\sum_{t=1}^{H} r_t$. Inspired by previous results on Dec-POMDPs [Oliehoek, 2013], the authors propose to establish value functions of POSGs based on so-called *plan-time sufficient statistics*.

For t-th stage of the game, a plan-time sufficient statistic is defined as a probability distribution over possible joint action-observation histories of the players at t-th stage of the game, i.e., $p \in \Delta((A_1 O_1)^{t-1} \times (A_2 O_2)^{t-1})$. Based on this statistic, Wiggers et al. define value function V_t^* of the t-th stage of the game. For $p \in \Delta((A_1 O_1)^{t-1} \times (A_2 O_2)^{t-1})$, $V_t^*(p)$ is the utility $\sum_{i=t}^{H} r_t$ player 1 can achieve in the remaining stages when the distribution over possible joint private action-observation histories of the players is p. The authors show that the value function V_H^* for the last stage game corresponds to the value function of a *family of Bayesian games*. Here, types $\Theta_i^H = (A_i O_i)^{H-1}$ of player i correspond to possible private action-observation histories player i can have at the beginning of the last stage H of the game, and the utility function involves only single reward, $u(a_1, a_2 \mid \theta_1, \theta_2) = \sum_{s^{(H)} \in S} \mathbb{P}[s^{(H)} | \theta_1, \theta_2] \cdot R(s^{(H)}, a_1, a_2)$. The authors show that V_H^* is concave in the marginal probabilities over action-observation histories $(A_1 O_1)^{H-1}$ of the maximizing player 1, and convex in the marginal probabilities over $(A_2 O_2)^{H-1}$ of the minimizing player 2. Furthermore, they show that the value function of previous stages $t < H$ can be characterized inductively while preserving the structural properties, i.e., V_t^* is concave in marginals $\Delta((A_1 O_1)^t)$ and convex in $\Delta((A_2 O_2)^t)$.

Although Wiggers et al. [2016] provide a useful theoretical characterization of value in finite-horizon zero-sum POSGs, the number of possible action-observations histories and therefore also the dimension of the value functions still grow exponentially in horizon. Furthermore, the results are based on an inductive construction that starts from the value function of the last stage of the game and they cannot therefore be applied to the

[2] This does not restrict generality of the model. The rewards can be made observable by including them in observations the players get.

infinite-horizon setting that we are interested in. To this end, we focus on games where this growth in the dimension does not occur. We will now provide an overview of some classes of such games that are relevant to this thesis.

Sorin [2003] studies zero-sum stochastic games with one-sided incomplete information and both finite and infinite horizon (with discounted rewards). Here, one player is perfectly informed about the states, actions, and observations of the other player (hence having perfect information about the game), while his adversary is less informed. In the case of the discounted problem, the value function $V^* : \Delta(S) \to \mathbb{R}$ is defined over beliefs over states and is characterized by a recursive formula

$$V^*(b) = \min_{\pi_2(\cdot|s)\in\Delta(A_2)} \max_{a_1\in A_1} \Big[\lambda \sum_{s,a_2} b(s)\pi_2(a_2 \mid s)R(s,a_1,a_2) + \\ + (1-\lambda)\mathbb{E}_{b,a_1,\pi_2}[V^*(\tau(b,a_1,\pi_2,o)]\Big] \tag{2.11}$$

where τ denotes the Bayesian update of the belief of the imperfectly-informed player.[3] Furthermore, V^* is a convex function which makes it structurally similar to the optimal value function in POMDPs (see Section 2.2 for more details). Similar models where one player is perfectly informed has also appeared, e.g., in [Chatterjee and Doyen, 2014] as *semi-perfect information games* where the game is, however, considered to be turn-based and a different class of objectives is used, or [Krausz and Rieder, 1997], where the actions of the players are, however, assumed to be observable which is not convenient for security applications we are interested in[4].

The characterization of value function of one-sided models (Equation (2.11)) is possible since the players can infer and agree upon each other's beliefs when the strategy π_2 is known—i.e., the beliefs are Markov. Cole and Kocherlakota [2001] aim to identify a class of games where beliefs are Markov. They use a factorization of the state space $s = (s_1, s_2)$, where player i observes s_i but not s_{-i}, and observations $o_i = (o_{\text{pub}}, o_i')$ consist of a commonly known public observation o_{pub} and a private observation o_i' of player i. To this end, they study deterministic *Markov-private strategies* $\sigma_i : (O_{\text{pub}})^* S_i \to A_i$ that depend only on the public information available to the players and their *current* private state, and they suggest that the beliefs are Markov when the belief of player i at the t-th stage depends only on the public information $(o_{\text{pub}})_{j=1}^{t-1}$ and the current private state $s_i^{(t)}$ of player i. Cole and Kocherlakota [2001] provides an algorithmic characterization of achievable payoffs, however, the approach does not translate to an efficient algorithm and applies only to the setting where the strategies are deterministic.

MacDermed [2013] studies a model called *Markov games of incomplete information* (MaGII) which closely resembles the model of Cole and Kocherlakota [2001]. Similarly, the state space is factored $s = (s', s_1, s_2)$ where the private states s_i act as private

[3] In [Sorin, 2003], the role of the players is reversed, and the maximizing player is assumed to be perfectly informed. To ensure consistency with the rest of the thesis, we reversed the roles of the players in the formula.

[4] E.g., in patrolling problems, the attacker would have been required to announce that he attacks a particular target if the actions were observable.

observations, and the model requires that the distribution over states $s^{(t)}$ is independent of private signal $s_i^{(t-1)}$ of the player i given a history of public observations and the *current* private signal s_i^t of player i. MacDermed [2013] provides an algorithmic approach for solving common-payoff case of MaGIIs (which corresponds to a subclass of Dec-POMDPs and hence is conceptually different from zero-sum POSGs). MacDermed [2013] further suggest that MaGIIs with arbitrary payoffs (i.e., including the zero-sum case) can be converted to a belief-stochastic game, however, no algorithm for solving such game is provided.[5] Furthermore, the strategies from this belief-stochastic game can only be used when the players are required to publicly announce their past strategies (the subjective belief of the player then corresponds to the true distribution over possible situations in the game). This cannot be expected to happen in the domains we consider (e.g., the actual strategy the attacker uses is not advertised) and the belief typically cannot be used as a sufficient statistic in imperfect information games [Burch et al., 2014; Burch, 2018]. We demonstrate the insufficiency of belief in imperfect-information games (specifically one-sided POSGs) in Section 3.7.

2.1.5 Extensive-form Games

Much of the success in designing scalable algorithms for solving dynamic games with imperfect information has been achieved in the domain of extensive-form games (or EFGs) [Koller et al., 1996; Bošanský et al., 2014; Moravčík et al., 2017; Brown and Sandholm, 2018]. A two-player zero-sum extensive-form game is traditionally thought of as a finite game tree $\mathcal{H}$, where each node of the game tree represents a history $h \in \mathcal{H}$ of play. Unless the history h represents a leaf node of the tree, a player $\mathcal{P}(h) \in \{1, 2\} \cup \{c\}$ (including the chance player c simulating stochastic actions of the environment) takes an action $a \in \mathcal{A}(h)$—and thus extends the current history h to ha. Histories $\mathcal{Z} \subset \mathcal{H}$ which are the leafs of the tree are called *terminal* and mark the end of the game. Upon reaching a terminal history $h \in \mathcal{Z}$, player 1 and player 2 collect utility $u(h)$ and $-u(h)$, respectively. The limited information of the players is captured by the notion of *information sets* (or *infosets* for short). Non-terminal histories $\mathcal{H}_i = \{h \in \mathcal{H} \setminus \mathcal{Z} \mid \mathcal{P}(h) = i\}$ where player $i \in \{1, 2\}$ is to play are partitioned into disjoint sets $\mathcal{I}_i = \{I_i^{(1)}, \dots I_i^{(k)}\}$. An infoset $I_i \in \mathcal{I}_i$ denotes a set of histories that the player i is unable to tell apart (i.e., also $\mathcal{A}(h)$ is identical for all $h \in I_i$ and we denote that $\mathcal{A}(I_i)$ for simplicity). Since the histories in $I_i \in \mathcal{I}_i$ are indistinguishable, player i has to behave identically in all of these histories. To this end, a (behavioral) strategy σ_i of player i maps information sets $I_i \in \mathcal{I}_i$ to randomized decisions, $\sigma_i(I_i) \to \Delta(\mathcal{A}(I_i))$. We denote the information set where history h is contained $\mathcal{I}^{-1}(h)$. In the remainder of this section, we will focus solely on extensive-form games with *perfect recall*, i.e., the players remember their past actions and past information they acquired.

[5] MacDermed [2013] suggests that the resulting stochastic game can be discretized and finite-state stochastic games algorithms can be applied to obtain an approximate solution. The actual construction of the game is not covered, and to the best of our knowledge the construction is not straightforward.

Definition 2.6 (Perfect-recall in EFGs)**.** An EFG is of perfect recall iff:

1. For every player $i \in \{1,2\}$, every two distinct infosets $I_i, I_i' \in \mathcal{I}_i$ ($I_i \neq I_i'$), every two histories $h \in I_i$ and $h' \in I_i'$, and every two extensions of these histories $\overline{h} \sqsupseteq h$ and $\overline{h}' \sqsupseteq h'$ where player i is to act, $\mathcal{I}^{-1}(\overline{h}) \neq \mathcal{I}^{-1}(\overline{h}')$.
2. For every player $i \in \{1,2\}$, every history h of player i (i.e., $\mathcal{P}(h) = i$), every two distinct actions $a, b \in \mathcal{A}(h)$ and every two extensions $h_a \sqsupseteq ha$ and $h_b \sqsupseteq hb$ where player i is to play (i.e., $\mathcal{P}(h_a) = \mathcal{P}(h_b) = i$), $\mathcal{I}^{-1}(h_a) \neq \mathcal{I}^{-1}(h_b)$.

The restriction on perfect-recall games does not significantly reduce the applicability of the EFG model, however, it allows for design of scalable algorithms to compute Nash equilibrium strategies in EFGs with perfect recall. The examples of such algorithms include sequence-form linear programming [Koller et al., 1996] or counterfactual regret minimization [Zinkevich et al., 2008].

Viewing POSG as extensive-form game Every POSG with a finite horizon H can be thought of as an extensive-form game. Here, histories correspond to partial plays in a POSG, $\mathcal{H} = \mathcal{H}_1 \cup \mathcal{H}_2 \cup \mathcal{H}_c$ where $\mathcal{H}_1 = \bigcup_{t=1}^{H} (SA_1A_2O_1O_2)^{t-1}S$, $\mathcal{H}_2 = \bigcup_{t=1}^{H} (SA_1A_2O_1O_2)^{t-1}SA_1$ and $\mathcal{H}_c = \bigcup_{t=1}^{H} (SA_1A_2O_1O_2)^{t-1}SA_1A_2$. At the beginning of stage t, the history of the EFG is $h = (s^{(j)}a_1^{(j)}a_2^{(j)}o_1^{(j)}o_2^{(j)})_{j=1}^{t-1}s^{(t)} \in \mathcal{H}_1$. The players then consecutively choose their action: Player 1 chooses $a_1^{(t)} \in A_1$ and extends the history to $ha_1^{(t)} \in \mathcal{H}_2$, then player 2 chooses $a_2^{(t)} \in A_2$ and extends the history to $ha_1^{(t)}a_2^{(t)} \in \mathcal{H}_c$. Finally, in history $ha_1^{(t)}a_2^{(t)} \in \mathcal{H}_c$ the chance player applies a fixed strategy simulating the transition probability $T(o_1^{(t)}, o_2^{(t)}, s^{(t+1)} \mid s^{(t)}, a_1^{(t)}a_2^{(t)})$ and the history is extended to $ha_1^{(t)}a_2^{(t)}o_1^{(t)}o_2^{(t)}s^{(t+1)}$. After H stages elapse, a terminal history $h_z = (s^{(j)}a_1^{(j)}a_2^{(j)}o_1^{(j)}o_2^{(j)})_{j=1}^{H}s^{(H+1)} \in \mathcal{Z}$ is reached and player 1 receives utility $\sum_{j=1}^{H} R(s^{(j)}, a_1^{(j)}, a_2^{(j)})$ or $\sum_{j=1}^{H} \gamma^{j-1} R(s^{(j)}, a_1^{(j)}, a_2^{(j)})$ when the discounting is employed.

Although the players decide sequentially in this EFG, the structure of information sets ensures that they cannot use information they are not supposed to know in the POSG. Namely, we have an information set $I_i^{\omega} \in \mathcal{I}_i$ for every action-observation history $\omega \in \bigcup_{t=1}^{H} (A_iO_i)^{t-1}$ of player i, and I_i^{ω} consists of all histories $h \in \mathcal{H}_i$ where the action observation history of player i is ω. Observe that in such a case, the behavioral strategy σ_i of player i coincides with the behavioral strategy in the POSG with horizon H.

Based on this formulation, we provide a simple argument showing the existence of the value of infinite-horizon partially observable stochastic games when the discounted sum of rewards $\sum_{t=1}^{\infty} \gamma^{t-1} r_t$ is used as the objective. The proof relies on constructing EFG representations of a finite-horizon version G_H of the given POSG. This game has horizon H and the leaf nodes are assigned utility $\sum_{t=1}^{H} \gamma^{t-1} r_t$ (i.e., partial sum of the infinite discounted sum of rewards).

Theorem 2.3. *Any infinite-horizon zero-sum POSG with* Disc^{γ} *objective has a value.*

Proof. Denote V_H the value of EFG G_H constructed for the POSG with horizon H, and let (σ_1^H, σ_2^H) be Nash equilibrium strategy profile of G_H. Since σ_1^H is a Nash equilibrium of G_H, playing σ_1^H in the infinite-horizon POSG player 1 ensures that the expected partial sum $\mathbb{E}[\sum_{t=1}^{H} \gamma^{t-1} r_t]$ of rewards in the first H stages is at least V_H. The least payoff the player 1 achieves in the rest of the game (when employing arbitrary strategy) is $\sum_{t=H+1}^{\infty} \gamma^{t-1} r_t \geq \gamma^H \sum_{t=1}^{\infty} \gamma^{t-1} \underline{r} = \gamma^H \underline{r}/(1-\gamma)$ where $\underline{r}$ is the least reward in the game. Hence by playing σ_1^H in the infinite horizon POSG for first H stages, player 1 is able to ensure utility of at least $\underline{v}_H = V_H + \gamma^H \underline{r}/(1-\gamma)$. Analogous reasoning can be employed for player 2 to obtain that σ_2^H guarantees that the utility will be at most $\overline{v}_H = V_H + \gamma^H \overline{r}/(1-\gamma)$ where $\overline{r}$ is the maximum reward in the POSG. As $H \to \infty$, the utilities $\underline{v}_H$ and $\overline{v}_H$ the players are able to guarantee in the infinite-horizon game by playing σ_1^H and σ_2^H converge, and $\lim_{H\to\infty} V_H$ is the value of the infinite-horizon POSG. □

Solving extensive-form games Traditional approaches to solving extensive-form games, such as *sequence-form linear program* [Koller et al., 1996] or *counterfactual regret minimization* [Zinkevich et al., 2008] reason about the entire extensive-form game prior to play. The limitations of such an approach for solving finite-horizon POSGs with non-trivial horizon H is apparent as the size of the game tree is exponential in the horizon. To the best of our knowledge, heads-up limit Texas hold'em poker (HULHE) is the largest game that has been solved [Bowling et al., 2015] by approaches that operate on the entire game tree. This game consists of 10^{14} information sets (i.e., action-observation sequences in POSG terminology) and required 900 core-years to be weakly solved. To put this in perspective, the number of information sets in HULHE is comparable with the number of sequences of player i in a POSGs of trivial size where player i has only $|A_i| = 2$ actions and $|O_i| = 2$ observations and the considered horizon is $H = 24$. Techniques for solving larger extensive-form games therefore usually rely on *abstractions* where a smaller version of the game is formed and solved, and the resulting strategy is translated back to the original game. However, such an approach to solving EFGs is only approximate and typically leads to exploitable strategies that can be defeated [Wood, 2015].

Continual resolving & Deepstack A recent approach of Deepstack [Moravčík et al., 2017] revolutionized the landscape of scalable algorithms for solving large extensive-form games and allowed for the design of the first poker bot that achieved expert-level performance in the game of heads-up no-limit poker (HUNL). Compared to HULHE, the no-limit version of the game is significantly larger with 10^{160} decision points. The approach relies on the idea of *continual-resolving* [Burch et al., 2014] that allows for decomposition of the game based on the public information the players have. Namely, it allows us to reconstruct a strategy for a *subgame* (i.e., subtree where the players start with the same shared public information) without remembering the strategy for the rest of the game that is not reached. Instead, one needs only to remember the (counterfactual) values the subgame strategy has to achieve in the top-level information

sets of the adversary (i.e., the value of the resolved strategy given each possible private information the adversary could have at the beginning of the subgame).

To obtain the counterfactual values in order to start resolving the game, it is necessary to solve the entire game first—which renders this approach of limited practical applicability in solving very large games such as HUNL. Deepstack [Moravčík et al., 2017] avoids such reasoning by introducing a limited-horizon lookahead. This idea is common in perfect-information games, such as chess, but it has been unknown in the context of games with imperfect-information. Instead of solving the entire (sub-)game from the root to the very bottom of the tree, the tree is truncated at a predefined depth (thus lowering the number of decision nodes within the considered tree significantly). At the cutoff depth, the counterfactual values for the remainder of the game are estimated based on the belief over possible private cards of player i (termed *range*) player $-i$ would have at the given situation of the game when the current estimate of (partial) strategy of player i is considered. In Deepstack, these estimates are provided by a neural network that is trained by solving randomly generated poker situations.

Similarly to the recursive characterization of value in POSGs (e.g., as in Equation (2.11)), Deepstack represents the solution of the current situation in terms of values of later situation. However, unlike our approach that relies upon the recursive characterization of value, the value function in Deepstack is more complex (instead of a single number, it outputs a vector of counterfactual values for each possible private information), and the approach does not leverage the structural properties of the value function represented by a neural network, which in the case of Deepstack is treated as a black box. The distinction can also be seen in the procedure used to propagate the values. Deepstack uses an iterative procedure of CFR [Zinkevich et al., 2008] that iteratively approximates the solution of every lookahead tree, while our approach uses exact mathematical programming. However, the key difference is that we view the problem from the perspective of infinite-horizon POSGs, while Deepstack and continual resolving is framed within the context of finite-horizon extensive-form games.

2.2 Partially Observable MDPs

Partially observable Markov decision processes (POMDPs) [Astrom, 1965; Sondik, 1978; Pineau et al., 2003; Smith and Simmons, 2004, 2005; Spaan and Vlassis, 2005; Bonet and Geffner, 2009; Somani et al., 2013] are a standard tool for single-agent decision making in stochastic environment under uncertainty about the states. From the perspective of partially observable stochastic games, POMDPs can be seen as a variant of POSG that is played by a single player only.

Definition 2.7 (Partially observable Markov decision process)**.** A *partially observable Markov decision process* is a tuple (S, A, O, T, R) where

- S is a finite set of states,

- A is a finite set of actions the agent can use,
- O is a finite set of observations the agent can observe,
- $T(o, s' \mid s, a)$ is a probability to transition to s' while generating observation o when the current state is s and agent uses action a,
- $R(s, a)$ is the immediate reward of the agent when using action a in state s.

In POMDPs, the agent starts with a known belief $b^{\text{init}} \in \Delta(S)$ that characterizes the probability $b^{\text{init}}(s)$ that s is the initial state. The play proceeds similarly as in POSGs, except that there is only one decision-maker involved: The initial state $s^{(1)}$ is sampled from the distribution b^{init}. Then, in every stage t, the agent decides about the current action $a^{(t)}$ and receives reward $R(s^{(t)}, a^{(t)})$ based on the current state of the environment $s^{(t)}$. With probability $T(o^{(t)}, s^{(t+1)} \mid s^{(t)}, a^{(t)})$ the system transitions to $s^{(t+1)}$ and the agent receives observation $o^{(t)}$. The decision process is then repeated. Although many objectives have been studied in POMDPs, in this section we discuss only discounted POMDPs with infinite-horizon, i.e., the objective is to optimize $\sum_{t=1}^{\infty} \gamma^{t-1} r_t$ for a discount factor $\gamma \in (0, 1)$.

A strategy $\sigma : (A_1 O)^* \to A_1$ in POMDPs is traditionally called a *policy* and assigns a deterministic action to each observed history $\omega \in (A_1 O)^*$ of the agent. Since the agent is the only decision-maker within the environment, and the probabilistic characterization of the environment is known, the player is able to infer his belief $\mathbb{P}_{b^{\text{init}}}[s^{(t+1)} \mid (a^{(i)} o^{(i)})_{i=1}^{t}]$ (i.e., how likely it is to be in a particular state after a sequence of actions and observations $(a^{(i)} o^{(i)})_{i=1}^{t}$ has been used and observed). This belief can be defined recursively

$$\tau(b, a, o)(s') = \eta \sum_{s \in S} b(s) \cdot T(o, s' \mid s, a) \tag{2.12}$$

where η is a normalizing term, and $\tau(b, a, o) \in \Delta(S)$ is the updated belief of the agent when his current belief was b and he played and observed (a, o). Sondik [1971] has shown that the belief of the agent is a sufficient statistic, and POMDPs can therefore be translated into *belief-space MDP*. In theory, standard methods for solving MDPs can be applied, and POMDPs can be solved, e.g., by iterating

$$V^{t+1}(b) = [HV^t](b) = \max_{a \in A} \left[\sum_{s \in S} b(s) \cdot R(s, a) + \gamma \sum_{o \in O} \mathbb{P}_b[o \mid a] \cdot V^t(\tau(b, a, o)) \right] . \tag{2.13}$$

Since H is a contraction, the repeated application of Equation (2.13) converges to a unique convex value function $V^* : \Delta(S) \to \mathbb{R}$ of the POMDP. However, since the number of beliefs is infinite, it is impossible to apply this formula to approximate V^* directly.

Exact value iteration The value iteration can be, however, rewritten in terms of operations with so-called α-vectors [Sondik, 1978]. An α-vector can be seen as a linear function $\alpha : \Delta(S) \to \mathbb{R}$ characterized by its values $\alpha(s)$ in the vertices $s \in S$ of the belief simplex $\Delta(S)$. We thus have $\alpha(b) = \sum_{s \in S} b(s) \cdot \alpha(s)$.

Assume that V^t is a piecewise-linear and convex function where $V^t(b) = \max_{\alpha \in \Gamma^t} \alpha(b)$ for a finite set of α-vectors Γ^t. We can then form a new (finite) set Γ^{t+1} of α-vectors to represent V^{t+1} from Equation (2.13) by considering all possible combinations of α-vectors from the set Γ^t:

$$\Gamma^{t+1} = \left\{ \alpha : \alpha(s) = R(s,a) + \gamma \sum_{(o,s') \in O \times S} T(o, s' \mid s, a) \alpha^o(s') \mid \forall a \in A, (\alpha^o)_{o \in O} \text{ for } \alpha^o \in \Gamma^t \right\} . \tag{2.14}$$

As $|\Gamma^{t+1}| = |A| \cdot |\Gamma^t|^{|O|}$, this exact approach suffers from poor scalability. Several techniques have been proposed to reduce the size of sets Γ^t [Littman, 1996; Zhang and Zhang, 2001], however, this still does not translate to an efficient algorithm.

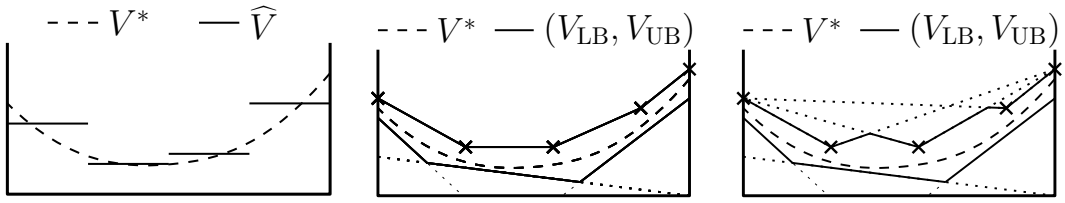
In the remainder of this section, we present two scalable algorithms for solving POMDPs that are relevant to this thesis. First, we present RTDP-Bel that uses discretized value function and applies Equation (2.13) directly. Second, we present heuristic search value iteration (HSVI) [Smith and Simmons, 2004, 2005] that inspires our methods for solving POSGs.

RTDP-Bel The RTDP-Bel algorithm [Bonet, 1998] is based on RTDP [Barto et al., 1995] and has been originally framed in the context of Goal-POMDPs. In Goal-POMDPs, the problem is undiscounted (i.e., $\gamma = 1$ in Equation (2.13)), however, the agent is incentivized to reach the goal state g as his reward for every transition before reaching the goal is negative (i.e., it represents the cost). The RTDP-Bel also applies to discounted POMDPs as discounting can be modeled within the Goal-POMDP framework as a fixed probability $1 - \gamma$ of reaching the goal state during every transition [Bonet and Geffner, 2009].

RTDP-Bel adapts RTDP to partially observable domains by using a grid-based approximation of V^* and using a hash-table to store the values, where $V^*(b) \sim \widehat{V}(\lfloor K \cdot b \rfloor)$ for some fixed parameter $K \in \mathbb{N}$. This approximation, however, loses the theoretical properties of RTDP. The algorithm need not converge as the values of the discretized value function may oscillate. Moreover, there is no guarantee that the values stored in the hash-table will form a bound on the values of V^* [Bonet and Geffner, 2009, p. 3, last paragraph of Section 3]. Despite the lack of theoretical properties, RTDP-Bel has been shown to perform well in practice. The RTDP-Bel algorithm performs a sequence of trials (see Algorithm 2.1) that updates the discretized value function $\widehat{V}$.

Algorithm 2.1: A single trial of the RTDP-Bel algorithm.

```
b ← b^init;  s ∼ b
while b(g) < 1 do
    Q(b,a) ← Σ_{s∈S} b(s)R(s,a) + Σ_{o∈O} ℙ_b[o | a] · V̂(⌊K · τ(b,a,o)⌋)
    a* ← arg max_{a∈A} Q(b,a)
    V̂(⌊K · b⌋) ← Q(b,a*)
    (o,s') ∼ T(o,s' | s,a*);   b ← τ(b,a*,o);   s ← s'
```



(a) RTDP-Bel **(b)** HSVI (PWLC) **(c)** HSVI2 (sawtooth)

Figure 2.1: Comparison of value function approximation schemes

Heuristic search value iteration (HSVI) Heuristic search value iteration [Smith and Simmons, 2004, 2005] is a representative of a class of point-based methods for solving POMDPs. Unlike RTDP-Bel, it approximates V^* using piecewise-linear functions. We illustrate the difference between a grid-based approximation used in RTDP-Bel and a piecewise-linear approximation in Figures 2.1a and 2.1b. Observe that unlike the grid-based approximation, a piecewise-linear approximation can yield a close approximation of V^* even in regions with rapid change of value.

In the original version of the *heuristic-search value iteration* algorithm (HSVI) [Smith and Simmons, 2004], the algorithm keeps two piecewise-linear and convex (PWLC) functions V_{LB}^{Γ} and $V_{\mathrm{UB}}^{\Upsilon}$ to approximate V^* (see Figure 2.1b) and refines them over time. The lower bound on the value is represented in the vector-set representation using a finite set of α-vectors Γ, while the upper bound is formed as a lower convex hull of a set of points $\Upsilon = \{(b_i, y_i) \,|\, i = 1, \ldots, m\}$ where $b_i \in \Delta(S)$ and $y_i \in \mathbb{R}$. We then have

$$V_{\mathrm{LB}}^{\Gamma}(b) = \max_{\alpha \in \Gamma} \sum_{s \in S} b(s) \cdot \alpha(s) \tag{2.15a}$$

$$V_{\mathrm{UB}}^{\Upsilon}(b) = \min\{\textstyle\sum_{i=1}^{m} \lambda_i y_i \mid \lambda \in \mathbb{R}_{\geq 0}^{m} : \sum_{i=1}^{m} \lambda_i b_i = b\} \; . \tag{2.15b}$$

Computing $V_{\mathrm{UB}}^{\Upsilon}(b)$ according to Equation (2.15b) requires solving a linear program. In the second version of the algorithm (HSVI2, [Smith and Simmons, 2005]), the PWLC representation of upper bound has been replaced by a sawtooth-shaped approximation [Hauskrecht, 2000] (see Figure 2.1c). While the sawtooth approximation is less tight with the same set of points, the computation of $V_{\mathrm{UB}}^{\Upsilon}(b)$ does not rely on the use of linear programming and can be done in linear time in the size of Υ.

HSVI2 initializes the value function V_{LB}^{Γ} by considering policies 'always play action a' and constructing one α-vector for each action $a \in A$ corresponding to the expected cost for playing such policy. For the initialization of the upper bound, the fast-informed bound is used [Hauskrecht, 2000].

The refinement of V_{LB}^{Γ} and $V_{\mathrm{UB}}^{\Upsilon}$ is done by adding new elements to the sets Γ and Υ. Each of these updates is meant to improve the approximation quality in a selected belief b as much as possible, hence termed *point-based update* (see Algorithm 2.2).

Similarly to RTDP-Bel, HSVI2 selects beliefs where the update should be performed based on the simulated play (selecting actions according to $V_{\mathrm{UB}}^{\Upsilon}$). Unlike RTDP-Bel, however, observations are not selected randomly. Instead, HSVI2 selects an observation

Algorithm 2.2: Point-based `update`(b) procedure of $(V_{\mathrm{LB}}^{\Gamma}, V_{\mathrm{UB}}^{\Upsilon})$.

$\alpha^{a,o} \leftarrow \arg\max_{\alpha\in\Gamma} \sum_{s'\in S} \tau(b,a,o)(s') \cdot \alpha(s')$ for all $a \in A, o \in O$
$\alpha^{a}(s) \leftarrow R(s,a) + \gamma \sum_{o,s'} T(o, s' \mid s, a) \cdot \alpha^{a,o}(s')$ for all $s \in S, a \in A$
$\Gamma \leftarrow \Gamma \cup \{\arg\max_{\alpha^a} \sum_{s\in S} b(s) \cdot \alpha^a(s)\}$
$\Upsilon \leftarrow \Upsilon \cup \{(b, \max_{a\in A} \left[\sum_{s\in S} b(s)R(s,a) + \gamma \sum_{o\in O} \mathbb{P}_b[o \mid a] \cdot V_{\mathrm{UB}}^{\Upsilon}(\tau(b,a,o))\right])\}$

Algorithm 2.3: HSVI2 for discounted POMDPs. The pseudocode follows the ZMDP implementation and includes `update` on line 6.

1 Initialize V_{LB}^{Γ} and $V_{\mathrm{UB}}^{\Upsilon}$
2 **while** $V_{\mathrm{UB}}^{\Upsilon}(b^{\mathrm{init}}) - V_{\mathrm{LB}}^{\Gamma}(b^{\mathrm{init}}) > \varepsilon$ **do** `explore`$(b^{\mathrm{init}}, \varepsilon, 0)$
3 **procedure** `explore`(b, ε, t)
4 **if** $V_{\mathrm{UB}}^{\Upsilon}(b) - V_{\mathrm{LB}}^{\Gamma}(b) \leq \varepsilon\gamma^{-t}$ **then return**
5 $a^* \leftarrow \arg\max_{a\in A} \left[\sum_s b(s) \cdot R(s,a) + \gamma \sum_{o\in O} \mathbb{P}_b[o \mid a] V_{\mathrm{UB}}^{\Upsilon}(\tau(b,a,o))\right]$
6 `update`(b)
7 $o^* \leftarrow \arg\max_{o\in O} \mathbb{P}_b[o \mid a] \cdot \mathrm{excess}_{t+1}(\tau(b,a^*,o))$
8 `explore`$(\tau(b,a^*,o^*), \varepsilon, t+1)$
9 `update`(b)

with the highest *weighted excess gap*, i.e. the excess approximation error

$$\mathrm{excess}_{t+1}(\tau(b,a^*,o)) = V_{\mathrm{UB}}^{\Upsilon}(\tau(b,a^*,o)) - V_{\mathrm{LB}}^{\Gamma}(\tau(b,a^*,o)) - \varepsilon\gamma^{-(t+1)} \tag{2.16}$$

in $\tau(b,a^*,o)$ weighted by the probability $\mathbb{P}_b[o \mid a^*]$. This heuristic choice attempts to target beliefs where the update will have the most significant impact on $V_{\mathrm{UB}}^{\Upsilon}(b^{\mathrm{init}}) - V_{\mathrm{LB}}^{\Gamma}(b^{\mathrm{init}})$.

The HSVI2 algorithm for discounted-sum POMDPs ($\gamma \in (0,1)$) is shown in Algorithm 2.3. This algorithm provably converges to an ε-approximation of $V^*(b^{\mathrm{init}})$ using values $V_{\mathrm{LB}}^{\Gamma}(b^{\mathrm{init}})$ and $V_{\mathrm{UB}}^{\Upsilon}(b^{\mathrm{init}})$, see [Smith and Simmons, 2004].

CHAPTER 3

One-Sided POSGs

In this chapter, we study a class of *one-sided partially observable stochastic games* where one side of the game has imperfect information about the course of the game, while the other player is perfectly informed. Scotland Yard board game [Wikipedia, 2019] is an example of such a game. Here one party, the criminal who is escaping the team of policemen, has access to all information about the game, while the other party, the team of policemen trying to track down the criminal, is lacking information about the game (specifically about the movement of the criminal).

Other examples of games that fit within this class of games can be found amongst security-related problems. Many existing game models studied over the years naturally belong to the class of one-sided POSGs, namely, e.g., patrolling games [Basilico et al., 2009a; Vorobeychik et al., 2014], or recently proposed model of non-cooperative reinforcement learning (N-CIRL) [Zhang et al., 2019]. In many security-related problems, the defender is protecting an area (or a computer network, for example) against the attacker that aims at intruding the area (network) or attacking some target. The defender does not have full information about the environment since he does not know which actions the

---

This chapter is based on a manuscript by K. Horák, V. Kovařík and B. Bošanský that is being prepared for submission, and on the following publications:

[Horák et al., 2017a] Horák, K., Bošanský, B., and Pěchouček, M. (2017a). Heuristic Search Value Iteration for One-Sided Partially Observable Stochastic Games. In *31st AAAI Conference on Artificial Intelligence*, pages 558–564 (50%, 11 citations)

[Horák et al., 2017b] Horák, K., Zhu, Q., and Bošanský, B. (2017b). Manipulating adversary's belief: A dynamic game approach to deception by design for proactive network security. In *International Conference on Decision and Game Theory for Security (GameSec)*, pages 273–294 (33%, 18 citations)

[Horák and Bošanský, 2017] Horák, K. and Bošanský, B. (2017). Dynamic Programming for One-sided Partially Observable Pursuit-evasion Games. In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART)*, pages 503–510 (50%)

[Horák and Bošanský, 2016] Horák, K. and Bošanský, B. (2016). A Point-Based Approximate Algorithm for One-Sided Partially Observable Pursuit-Evasion Games. In *International Conference on Decision and Game Theory for Security (GameSec)*, pages 435–454 (50%, 5 citations)

attacker performed (e.g., which hosts in the computer network have been compromised by the attacker). At the same time, it is difficult for the defender to know exactly what information the attacker has since the attacker can have infiltrated the system, or he might have access to insider information, and thus he can have substantial knowledge about the environment. Hence, as the worst-case assumption, the defender can assume that the attacker has full knowledge about the environment. From this perspective, one-sided POSGs can be used to compute robust defense strategies. We restrict to the strictly competitive (or zero-sum) setting and in this case the defender has guarantees on the expected outcome when using such robust strategies even against attackers with less information.

The class of one-sided POSGs has been studied previously in [Sorin, 2003] as *Level-1* stochastic games with incomplete information. Here, the author shows that the value of the game (parameterized by the belief of the imperfectly informed player) can be characterized by a recursive formula (see Equation (2.11)). Furthermore, the value function that assigns the value of a game to each belief is a convex function. Although Sorin [2003] provides a useful characterization of the value of one-sided POSGs, to the best of our knowledge, no previous practical algorithm for computing solutions of one-sided POSGs existed.

In this chapter, we focus on the algorithmic aspect of solving one-sided POSGs. The results from [Sorin, 2003] suggest that the solution of one-sided POSGs has a similar structure to value function of POMDPs (see Section 2.2). It is therefore natural to ask whether POMDP techniques can be extended to apply to one-sided POSGs. We answer this question affirmatively. First, we provide a detailed theoretical analysis of the one-sided POSG model (including restating structural results of Sorin [2003] within our formalism, see Lemma 3.3 and Theorem 3.15). This analysis will later help us to prove the correctness of our scalable algorithm for solving one-sided POSGs. Namely, we represent and later approximate the value function of one-sided POSGs by considering values of strategies of the imperfectly informed player. We discuss the structural properties of values of strategies, as well as, the value function of the game in Section 3.2. We show that these values of strategies are linear in the belief of the imperfectly informed player and that we can find the optimal value function by iteratively constructing improving strategies (Sections 3.3 and 3.4). Then, based on this idea, we devise an exact value iteration algorithm (Section 3.5) that similarly to the related algorithm for POMDPs relies on manipulating finite sets of α-vectors. However, since one-sided POSGs extend POMDPs[1], we cannot expect that this exact method translates to a scalable algorithm. To this end, our main contribution is a scalable algorithm (Section 3.6) that is inspired by the heuristic search value iteration method from the domain of POMDPs [Smith and Simmons, 2004, 2005]. This algorithm keeps and iteratively improves lower and upper bounds on the optimal value function of one-sided POSGs. We provide a detailed analysis of the algorithm, and we prove that the algorithm is guaranteed to approximate the value

[1]One can model a POMDP as a one-sided POSG by preventing the perfectly-informed player from having impact on transitions and rewards, e.g., by setting $A_2 = \{\texttt{noop}\}$.

of one-sided POSGs within arbitrary $\varepsilon > 0$ starting from an arbitrary initial belief of the imperfectly-informed player. Furthermore, we provide online algorithms that use the bounds computed by the proposed algorithm to play the game (Section 3.7). These algorithms allow us to obtain ε-Nash equilibrium strategies for the players.

In Section 3.9, we then provide one of the early applications of the ideas presented in this chapter to the problem of active deception in cybersecurity. Here, we use the one-sided POSG model to understand the beliefs of the attacker and how the beliefs of the attacker influence the efficiency of deceptive techniques. In the provided case study, we show that the use of game-theoretic active deception can significantly improve the security level of network operation.

3.1 Game Model

Definition 3.1 (One-sided POSGs)**.** A *one-sided POSG* (or OS-POSG) is a tuple $G = (S, A_1, A_2, O, T, R, \gamma)$ where

- S is a finite set of of game *states*,
- A_1 and A_2 are finite sets of *actions* of player 1 and player 2, respectively,
- O is a finite set of *observations*
- for every $(s, a_1, a_2) \in S \times A_1 \times A_2$, $T(\cdot \mid s, a_1, a_2) \in \Delta(O \times S)$ represents probabilistic transition function,
- $R : S \times A_1 \times A_2 \to \mathbb{R}$ is a reward function of player 1,
- $\gamma \in (0, 1)$ is a discount factor.

The game starts by sampling the initial state $s^{(1)} \sim b^{\text{init}}$ from a distribution b^{init} called the *initial belief*. Then the game proceeds for an infinite number of *stages* where players choose their actions simultaneously and receive feedback from the environment. At the beginning of i-th stage, the current state $s^{(i)}$ is revealed to player 2, but not to player 1. Then player 1 selects action $a_1^{(i)} \in A_1$ and player 2 selects action $a_2^{(i)} \in A_2$. Based on the current state of the game $s^{(i)}$ and the actions $(a_1^{(i)}, a_2^{(i)})$ taken by the players, an unobservable reward $R(s^{(i)}, a_1^{(i)}, a_2^{(i)})$ is assigned[2] to player 1, and the game transitions to a state $s^{(i+1)}$ while generating observation $o^{(i)}$ with probability $T(o^{(i)}, s^{(i+1)} \mid s^{(i)}, a_1^{(i)}, a_2^{(i)})$. After committing to action $a_2^{(i)}$, player 2 observes the entire outcome of the current stage, including the action $a_1^{(i)}$ taken by player 1 and the observation $o^{(i)}$. The player 1, on the other hand, knows only his own action $a_1^{(i)}$ and the observation $o^{(i)}$, while the action $a_2^{(i)}$ of player 2 and both the past and new states of the system $s^{(i)}$ and $s^{(i+1)}$ remain unknown to him.

The information asymmetry in the game means that while player 2 can observe entire course of the game $(s^{(i)} a_1^{(i)} a_2^{(i)} o^{(i)})_{i=1}^{t} s^{(t+1)} \in (S A_1 A_2 O)^* S$ up to the current

[2]Note that we consider a zero-sum setting, hence the reward of player 2 is $-R(s^{(i)}, a_1^{(i)}, a_2^{(i)})$. We do however consider that player 2 focuses on minimizing the reward of player 1 instead of reasoning about the rewards of player 2 directly.

decision point at time $t+1$, the player 1 only knows his own actions and observations $(a_1^{(i)} o_{i=1}^{(i)})^t \in (A_1 O)^*$. The knowledge of players translates to the probabilistic decision rules the players use to act in the game.

Definition 3.2 (Behavioral strategy)**.** Let G be a one-sided POSG. Mappings $\sigma_1 : (A_1 O)^* \to \Delta(A_1)$ and $\sigma_2 : (S A_1 A_2 O)^* S \to \Delta(A_2)$ are *behavioral strategies* of imperfectly informed player 1 and perfectly informed player 2, respectively. The sets of all behavioral strategies of player 1 and player 2 are denoted Σ_1 and Σ_2, respectively.

Plays in OS-POSGs Players use their behavioral strategies (σ_1, σ_2) to play the game. A *play* is an infinite word $(s^{(i)} a_1^{(i)} a_2^{(i)} o^{(i)})_{i=1}^{\infty}$, while finite prefixes of plays $w = (s^{(i)} a_1^{(i)} a_2^{(i)} o^{(i)})_{i=1}^{T} s^{(T+1)}$ are called *histories* of length T, and plays having w as a prefix are denoted $\mathsf{Cone}(w)$. Formally, a *cone* of w is a set of all plays extending w,

$$\mathsf{Cone}(w) = \left\{ (s^{(i)} a_1^{(i)} a_2^{(i)} o^{(i)})_{i=1}^{\infty} \in (S A_1 A_2 O)^* \mid (s^{(i)} a_1^{(i)} a_2^{(i)} o^{(i)})_{i=1}^{\infty} \text{ extends } w \right\} . \quad (3.1)$$

At a decision point at time t, players extend a history $(s^{(i)} a_1^{(i)} a_2^{(i)} o^{(i)})_{i=1}^{t} s^{(t+1)}$ of length t by sampling actions from their strategies $a_1^{(t+1)} \sim \sigma_1((a_1^{(i)} o^{(i)})_{i=1}^{t})$ and $a_2^{(t+1)} \sim \sigma_2((s^{(i)} a_1^{(i)} a_2^{(i)} o^{(i)})_{i=1}^{t} s^{(t+1)})$. We consider a discounted-sum objective with discount factor $\gamma \in (0,1)$, denoted Disc^{γ}, and the payoff associated with a play $(s^{(i)} a_1^{(i)} a_2^{(i)} o^{(i)})_{i=1}^{\infty}$ is thus $\sum_{i=1}^{\infty} \gamma^{i-1} R(s^{(i)}, a_1^{(i)}, a_2^{(i)})$. Player 1 is aiming to maximize this quantity while player 2 is minimizing it.

Apart from reasoning about decision rules of the players for the entire game (i.e., their behavioral strategies σ_1 and σ_2), we also consider the strategies they use for a single decision point—or stage—of the game only (i.e., assuming that the course of the previous stages $(s^{(i)} a_1^{(i)} a_2^{(i)} o^{(i)})_{i=1}^{t}$ is fixed and considered a parameter of the given stage).

Definition 3.3 (Stage strategy)**.** Let G be a one-sided POSG. A *stage strategy* of player 1 is a distribution $\pi_1 \in \Delta(A_1)$ over the actions player 1 can use at the current stage. A *stage strategy* of player 2 is a mapping $\pi_2 : S \to \Delta(A_2)$ from the possible current states of the game (player 2 observes the true state at the beginning of the current stage) to a distribution over actions of player 2. The sets of all stage strategies of player 1 and player 2 are denoted Π_1 and Π_2, respectively.

Note that a stage strategy of player 2 is essentially a conditional probability distribution given the current state of the game. For the reasons of notational convenience, we use notation $\pi_2(a_2 \,|\, s)$ instead of $\pi_2(s)(a_2)$ wherever applicable.

3.1.1 Probability measures

We now proceed by defining a probability measure on the space of infinite plays in one-sided POSGs. Assuming that $b \in \Delta(S)$ is the initial belief characterizing the distribution over possible initial states, and players use strategies (σ_1, σ_2) to play the game from the current situation, we can define the probability distribution over histories (i.e., prefixes of plays) recursively as follows.

$$\mathbb{P}_{b,\sigma_1,\sigma_2}[s^{(1)}] = b(s^{(1)}) \tag{3.2a}$$

$$\begin{aligned}\mathbb{P}_{b,\sigma_1,\sigma_2}[(s^{(i)}a_1^{(i)}a_2^{(i)}o^{(i)})_{i=1}^{t}s^{(t+1)}] = \mathbb{P}_{b,\sigma_1,\sigma_2}[(s^{(i)}a_1^{(i)}a_2^{(i)}o^{(i)})_{i=1}^{t-1}s^{(t)}] \cdot \\ \cdot \sigma_1((a_1^{(i)}o^{(i)})_{i=1}^{t-1}, a_1^{(t)}) \cdot \sigma_2((s^{(i)}a_1^{(i)}a_2^{(i)}o^{(i)})_{i=1}^{t-1}s^{(t)}, a_2^{(t)}) \cdot \\ \cdot T(o^{(t)}, s^{(t+1)} \mid s^{(t)}, a_1^{(t)}, a_2^{(t)})\end{aligned} \tag{3.2b}$$

This probability distribution also coincide with a measure μ defined over the cones, i.e. plays having w as a prefix.

$$\mu(\mathsf{Cone}(w)) = \mathbb{P}_{b,\sigma_1,\sigma_2}[w] \tag{3.3}$$

The measure μ uniquely extends to the probability measure $\mathbb{P}_{b,\sigma_1,\sigma_2}[\cdot]$ over infinite plays of the game, which allows us to define the expected utility $\mathbb{E}_{b,\sigma_1,\sigma_2}[\mathsf{Disc}^\gamma]$ of the game when the initial belief of the game is b and strategies $\sigma_1 \in \Sigma_1$ and $\sigma_2 \in \Sigma_2$ are played by player 1 and player 2, respectively.

Apart from probability measure $\mathbb{P}_{b,\sigma_1,\sigma_2}[\cdot]$ over plays and histories starting in state $s^{(1)} \sim b$ when strategies $\sigma_1 \in \Sigma_1$ and $\sigma_2 \in \Sigma_2$ of player 1 and player 2, respectively, are played, we also introduce a probability measure $\mathbb{P}_{b,\pi_1,\pi_2}[\cdot]$ over the possible outcomes of a single stage. When the current state $s \sim b$ and players follow stage strategies $\pi_1 \in \Pi_1$ and $\pi_2 \in \Pi_2$, the probability $\mathbb{P}_{b,\pi_1,\pi_2}[s, a_1, a_2, o, s']$ that the stage starts in state $s \in S$, player 1 and player 2 play actions $a_1 \in A_1$ and $a_2 \in A_2$, respectively, and the observation $o \in O$ is generated while transitioning to a new state $s' \in S$ satisfies

$$\mathbb{P}_{b,\pi_1,\pi_2}[s, a_1, a_2, o, s'] = b(s)\pi_1(a_1)\pi_2(a_2 \mid s)T(o, s' \mid s, a_1, a_2) \ . \tag{3.4}$$

Probability distribution in Equation (3.4) can be marginalized to obtain, e.g., the probability that player 1 plays action $a_1 \in A_1$ and observes $o \in O$,

$$\mathbb{P}_{b,\pi_1,\pi_2}[a_1, o] = \sum_{(s,a_2,s') \in S \times A_2 \times S} \mathbb{P}_{b,\pi_1,\pi_2}[s, a_1, a_2, o, s'] = \sum_{(s,a_2,s') \in S \times A_2 \times S} b(s)\pi_1(a_1)\pi_2(a_2 \mid s)T(o, s' \mid s, a_1, a_2) \ . \tag{3.5}$$

An important characteristic for the decision making of the imperfectly informed player 1 is his belief about the true current state of the game at the beginning of each stage. If player 1 knows the stage strategy π_2 used by player 2 in the current stage[3], player 1 is able to derive a distribution over possible states at the beginning of the next

[3]It cannot be, however, assumed that π_2 is known when playing the game. We address this in Section 3.7.

stage. If player 1 played $a_1 \in A_1$ and observed $o \in O$, his updated belief $\tau(b, a_1, \pi_2, o)$ over states is

$$\tau(b, a_1, \pi_2, o)(s') = \mathbb{P}_{b,\pi_1,\pi_2}[s' \mid a_1, o] = \sum_{(s,a_2)\in S\times A_2} \mathbb{P}_{b,\pi_1,\pi_2}[s, a_2, s' \mid a_1, o] \tag{3.6a}$$

$$= \frac{1}{\mathbb{P}_{b,\pi_1,\pi_2}[a_1, o]} \sum_{(s,a_2)\in S\times A_2} \mathbb{P}_{b,\pi_1,\pi_2}[s, a_1, a_2, o, s'] \tag{3.6b}$$

$$= \frac{1}{\mathbb{P}_{b,\pi_1,\pi_2}[a_1, o]} \sum_{(s,a_2)\in S\times A_2} b(s)\pi_1(a_1)\pi_2(a_2 \mid s)T(o, s' \mid s, a_1, a_2) \ . \tag{3.6c}$$

3.2 Value of One-Sided POSGs

We now proceed by establishing the value function of one-sided POSGs. The value function represents the utility player 1 is able to achieve in each possible initial belief of the game. First, we define the value of a strategy $\sigma_1 \in \Sigma_1$ of player 1, which assigns a payoff player 1 is guaranteed to get by playing σ_1 in the game (parameterized by the initial belief of the game). Based on the value of strategies, we define the optimal value function of the game where the player 1 chooses the best strategy for the given initial belief.

Definition 3.4 (Value of strategy)**.** Let G be a one-sided POSG and $\sigma_1 \in \Sigma_1$ be a behavioral strategy of the imperfectly informed player 1. The *value of strategy* σ_1, denoted val^{σ_1}, is a function mapping each belief $b \in \Delta(S)$ to the expected utility that σ_1 guarantees against a best-responding player 2 given that the initial belief is b,

$$\mathrm{val}^{\sigma_1}(b) = \inf_{\sigma_2\in\Sigma_2} \mathbb{E}_{b,\sigma_1,\sigma_2}[\mathsf{Disc}^\gamma] \ . \tag{3.7}$$

When given an instance of a one-sided POSG with initial belief b, player 1 seeks a strategy to play that yields the best possible expected utility $\mathrm{val}^{\sigma_1}(b)$. The value player 1 can guarantee in belief b is characterized by the optimal value function V^* of the game.

Definition 3.5 (Optimal value function)**.** Let G be a one-sided POSG. The *optimal value function* $V^* : \Delta(S) \to \mathbb{R}$ of G represents the supinf value of player 1 for each of the beliefs, i.e.

$$V^*(b) = \sup_{\sigma_1\in\Sigma_1} \mathrm{val}^{\sigma_1}(b) \ . \tag{3.8}$$

Note that according to Theorem 2.3, every zero-sum POSG with discounted-sum objective Disc^γ is determined in the sense that the lower (in the supinf sense) and upper (in the infsup sense) values of the game coincide and represent the value of the game.

Therefore, $V^*(b)$ also represents the value of the game when the initial belief of the game is $b \in \Delta(S)$.

Since the Disc^γ objective is considered (for $0 < \gamma < 1$), the infinite discounted sum of rewards of player 1 converge. As a result, the values of strategies $\mathrm{val}^{\sigma_1}(b)$ and value of the game $V^*(b)$ can be bounded.

Proposition 3.1. *Let G be a one-sided POSG. Disc^γ payoff of an arbitrary play in G is bounded by values*

$$L = \min_{(s,a_1,a_2)} R(s,a_1,a_2)/(1-\gamma) \qquad U = \max_{(s,a_1,a_2)} R(s,a_1,a_2)/(1-\gamma)\ . \tag{3.9}$$

Therefore also $L \leq V^*(b) \leq U$ and $L \leq \mathrm{val}^{\sigma_1}(b) \leq U$ for every belief $b \in \Delta(S)$ and every strategy $\sigma_1 \in \Sigma_1$ of the imperfectly informed player 1.

Proof. The smallest payoff player 1 can hypothetically achieve in any play consists of getting $\underline{r} = \min_{(s,a_1,a_2)} R(s,a_1,a_2)$ in every timestep. The infinite discounted sum $\sum_{t=1}^{\infty} \gamma^{t-1}\underline{r}$ converges to $\underline{r}/(1-\gamma) = L$. Conversely, the maximum payoff can be achieved if player 1 obtains $\overline{r} = \max_{(s,a_1,a_2)} R(s,a_1,a_2)$ in every timestep. Expected values of strategies (and therefore also the value of the game) are expectation over the payoffs of individual plays—hence are bounded by L and U as well. □

We now focus on the discussion of structural properties of solutions of one-sided POSGs. First, we show that the value of an arbitrary strategy $\sigma_1 \in \Sigma_1$ of player 1 is linear in $b \in \Delta(S)$. Note that every linear function $\alpha : \Delta(S) \to \mathbb{R}$ defined on the probability simplex $\Delta(S)$ can be represented as a convex combination of its values in the vertices of the simplex. We overload the notation and denote the value of α in vertex corresponding to $s \in S$ by $\alpha(s)$, and hence

$$\alpha(b) = \sum_{s \in S} \alpha(s) \cdot b(s) \qquad \text{where } \alpha(s) = \alpha(\mathbb{1}_s),\ \ \mathbb{1}_s(s') = \begin{cases} 1 & s = s' \\ 0 & \text{otherwise} \end{cases}\ . \tag{3.10}$$

In accordance with the POMDP notation, we will call such linear functions defined over the $\Delta(S)$ simplex α-vectors.

Lemma 3.2. *Let G be a one-sided POSG and $\sigma_1 \in \Sigma_1$ be an arbitrary behavioral strategy of player 1. Then the value val^{σ_1} of strategy σ_1 is a linear function in the belief space $\Delta(S)$.*

Proof. According to the Definition 3.4, the value val^{σ_1} of strategy σ_1 is defined as the expected utility of σ_1 against the best-response strategy σ_2 of player 2. However, before having to act, player 2 observes the true initial state $s \sim b$. Therefore, he will play a

best-response strategy σ_2 against σ_1 (with expected utility $\mathrm{val}^{\sigma_1}(s)$) given that the initial state is s. Since the probability that the initial state is s is $b(s)$, we have

$$\mathrm{val}^{\sigma_1}(b) = \sum_{s \in S} b(s)\mathrm{val}^{\sigma_1}(s) \ . \tag{3.11}$$

This shows that val^{σ_1} is a linear function in the belief $b \in \Delta(S)$. □

As the point-wise supremum of linear functions is convex, Lemma 3.2 directly implies that the optimal value function V^* is convex.

Lemma 3.3. *Optimal value function V^* of a one-sided POSG is convex.*

Proof. Definition 3.5 defines V^* as the point-wise supremum over linear functions val^{σ_1} (over all strategies $\sigma_1 \in \Sigma_1$ of player 1). This means that V^* is convex [Boyd and Vandenberghe, 2004, p.81]. □

Unless otherwise specified, we endow any space $\Delta(U)$ with the $\|\cdot\|_1$ metric. To later prove the correctness of the algorithm (presented in Section 3.6), we leverage the fact that both the value of strategies and the optimal value function V^* are Lipschitz continuous. Recall that for $k > 0$ a function $f : \Delta(U) \to \mathbb{R}$ is k-Lipschitz continuous if for every $p, q \in \Delta(U)$ it holds $|f(p) - f(q)| \leq k \cdot \|p - q\|_1$.

Lemma 3.4. *Let U be a finite set and let $f : \Delta(U) \to [y_{\min}, y_{\max}]$ be a linear function. Then f is k-Lipschitz continuous for $k = (y_{\max} - y_{\min})/2$.*

Proof. Let $p, q \in \Delta(U)$ be arbitrary two points in the probability simplex over the set U. Since f is a linear function, it can be represented as a convex combination of values $\alpha(u)$ in the vertices of the simplex corresponding to the elements $u \in U$,

$$f(p) = \sum_{u \in U} \alpha(u) \cdot p(u) \qquad \text{where} \qquad \alpha(u) = f(\mathbb{1}_u), \quad \mathbb{1}_u(v) = \begin{cases} 1 & v = u \\ 0 & \text{otherwise} \end{cases} \ . \tag{3.12a}$$

Without loss of generality, let us assume $f(p) \geq f(q)$. Now, the difference $|f(p) - f(q)|$ satisfies

$$|f(p) - f(q)| = f(p) - f(q) = \sum_{u \in U} \alpha(u) \cdot [p(u) - q(u)] \ . \tag{3.12b}$$

Denote $U^+ = \{u \in U \mid p(u) - q(u) \geq 0\}$ and $U^- = \{u \in U \mid p(u) - q(u) < 0\}$. We can now bound the difference from Equation (3.12b) by

$$|f(p) - f(q)| = \sum_{u \in U^+} \alpha(u) \cdot [p(u) - q(u)] + \sum_{u \in U^-} \alpha(u) \cdot [p(u) - q(u)] \tag{3.12c}$$

$$\leq \sum_{u \in U^+} y_{\max} \cdot [p(u) - q(u)] + \sum_{u \in U^-} y_{\min} \cdot [p(u) - q(u)] \tag{3.12d}$$

$$= y_{\max} \sum_{u \in U^+} [p(u) - q(u)] + y_{\min} \sum_{u \in U^-} [p(u) - q(u)] \ . \tag{3.12e}$$

Since both $p, q \in U$ (i.e., $\|p\|_1 = \|q\|_1 = 1$), it holds

$$\sum_{u \in U^+} [p(u) - q(u)] = - \sum_{u \in U^-} [p(u) - q(u)] = \|p - q\|_1 / 2 \ . \tag{3.12f}$$

Therefore

$$|f(p) - f(q)| \leq y_{\max} \|p - q\|_1 / 2 + y_{\min}(-\|p - q\|_1 / 2) = (y_{\max} - y_{\min})/2 \cdot \|p - q\|_1 \tag{3.12g}$$

which completes the proof. □

This Lemma 3.4 directly implies that both values val^{σ_1} of strategies σ_1 of the imperfectly informed player 1, as well as the optimal value function V^* are Lipschitz continuous.

Lemma 3.5. *Let $\sigma_1 \in \Sigma_1$ be an arbitrary strategy of the imperfectly informed player 1. Then val^{σ_1} is $(U - L)/2$-Lipschitz continuous.*

Proof. Value val^{σ_1} of strategy σ_1 is linear (Lemma 3.2) and its values are bounded by L and U (Proposition 3.1). Therefore, according to Lemma 3.4, function val^{σ_1} is $(U - L)/2$-Lipschitz. □

For the reasons of notational convenience, we denote the Lipschitz constant $\delta = (U - L)/2$ in the remainder of the text.

Theorem 3.6. *Value function V^* of one-sided POSGs is δ-Lipschitz continuous.*

Proof. V^* is defined as a supremum over δ-Lipschitz continuous values val^{σ_1} of strategies $\sigma_1 \in \Sigma_1$ of the imperfectly informed player 1. Therefore for arbitrary $b, b' \in \Delta(S)$, it holds

$$V^*(b) = \sup_{\sigma_1 \in \Sigma_1} \text{val}^{\sigma_1}(b) \leq \sup_{\sigma_1 \in \Sigma_1} [\text{val}^{\sigma_1}(b') + \delta \|b - b'\|_1] = V^*(b') + \delta \|b - b'\|_1 \ . \tag{3.13}$$

□

3.2.1 Elementary Properties of Convex Functions

In Lemma 3.3, we have shown that the optimal value function V^* of one-sided POSGs is convex. In this section, we will explicitly state some of the important properties of convex functions that motivate our approach and are used throughout the rest of the text.

Proposition 3.7. *Let $f : \Delta(S) \to \mathbb{R}$ be a point-wise supremum of linear functions, i.e.,*

$$f(b) = \sup_{\alpha \in \Gamma} \alpha(b) \ , \qquad \Gamma \subseteq \{\alpha : \Delta(S) \to \mathbb{R} \mid \alpha \text{ is linear}\} \ . \tag{3.14}$$

Then f is convex and continuous. Furthermore, if every $\alpha \in \Gamma$ is k-Lipschitz continuous, f is k-Lipschitz continuous.

Proof. Let $b, b' \in \Delta(S)$ and $\lambda \in [0, 1]$ be arbitrary. We have

$$\begin{aligned}
\lambda f(b) + (1-\lambda) f(b') &= \lambda \sup_{\alpha \in \Gamma} \alpha(b) + (1-\lambda) \sup_{\alpha \in \Gamma} \alpha(b') \\
&= \sup_{\alpha \in \Gamma} \lambda\alpha(b) + \sup_{\alpha \in \Gamma} (1-\lambda)\alpha(b') \\
&\geq \sup_{\alpha \in \Gamma} [\lambda\alpha(b) + (1-\lambda)\alpha(b')] \\
&= \sup_{\alpha \in \Gamma} \alpha(\lambda b + (1-\lambda)b') = f(\lambda b + (1-\lambda)b')
\end{aligned}$$

and f is therefore convex.

The continuity follows from the following argument. Every convex function is continuous on the interior of its domain. We will now show that it is continuous even on the boundary of $\Delta(S)$. Assume to the contradiction that it is not continuous, i.e., there exists b_0 on the boundary such that for all b from its neighborhood $f(b_0) > f(b) + C$ for some $C > 0$. Since f is a pointwise supremum of linear functions, there exists $\alpha \in \Gamma$ such that $\alpha(b_0) > f(b_0) - C/2$, however, at the same time $\alpha(b) \leq f(b_0) - C$. This is in contradiction with the fact that all $\alpha \in \Gamma$ are linear, and hence continuous.

Furthermore, suppose that every $\alpha \in \Gamma$ is k-Lipschitz continuous and let $b, b' \in \Delta(S)$. We have

$$\begin{aligned}
f(b) &= \sup_{\alpha \in \Gamma} \alpha(b) \\
&\leq \sup_{\alpha \in \Gamma} \left[\alpha(b') + k\|b - b'\|_1\right] && \text{(since every } \alpha \in \Gamma \text{ is } k\text{-Lipschitz)} \\
&= \left[\sup_{\alpha \in \Gamma} \alpha(b')\right] + k\|b - b'\|_1 \\
&= f(b') + k\|b - b'\|_1.
\end{aligned}$$

Since the identical argument proves the inequality $f(b') \leq f(b) + k\|b - b'\|_1$, this shows that f is k-Lipschitz continuous. □

Proposition 3.8. *Let $\Gamma \subseteq \{\alpha : \Delta(S) \to \mathbb{R} \mid \alpha \text{ is linear}\}$ be a set of linear functions. Then for every $b \in \Delta(S)$ we have*

$$\sup_{\alpha \in \Gamma} \alpha(b) = \sup_{\alpha \in \mathsf{Conv}(\Gamma)} \alpha(b) \ . \tag{3.15}$$

Proof. Clearly, it suffices to prove the inequality $\geq$. Let $b \in \Delta(S)$ be arbitrary and let $\sum_{i=1}^{k} \lambda_i \alpha_i$ be an arbitrary[4] convex combination of linear functions from Γ, i.e., $\alpha_i \in \Gamma$ for every $1 \leq i \leq k$. We need to show that $\alpha(b) \geq \sum_{i=1}^{k} \lambda_i \alpha_i(b)$ holds for some $\alpha \in \Gamma$. This is straightforward, as can be witnessed by the function $\alpha_{i^*} \in \Gamma$, $i^* := \arg\max_i \alpha_i(b)$:

$$\sum_{i=1}^{k} \lambda_i \alpha_i(b) \leq \sum_{i=1}^{k} \lambda_i \max_{1 \leq i \leq k} \alpha_i(b) = \max_{1 \leq i \leq k} \alpha_i(b) = \alpha_{i^*}(b) \ .$$

□

Proposition 3.9. *Let $f : \Delta(S) \to \mathbb{R}$ be a convex continuous function. Then there exists a set Γ of linear functions such that $\alpha \leq f$ for every $\alpha \in \Gamma$ and $f(b) = \sup_{\alpha \in \Gamma} \alpha(b)$ for every $b \in \Delta(S)$.*

Proof. Let $\Gamma := \{\alpha : \Delta(S) \to \mathbb{R} \text{ linear} \mid \alpha \leq f\}$. Clearly, the pointwise supremum of Γ is no greater than f. It remains to show that $\sup_{\alpha \in \Gamma} \alpha(b_0) \geq f(b_0)$ for each b_0. Let b_0 be an interior point of $\Delta(S)$. By the standard convex-analysis result, there exists a subdifferential of f at b_0, that is, a vector v such that $f(b) \geq f(b_0) + v \cdot (b - b_0)$ holds for each $b \in \Delta(S)$. The function $\alpha(b) := f(b_0) + v \cdot (b - b_0)$ is therefore contaned in Γ and witnesses that $\sup_{\alpha \in \Gamma} \alpha(b_0) \geq f(b_0)$.

Suppose that b_0 lies at the boundary of $\Delta(S)$ and let η, $\|\eta\|_1 = 1$, be a direction for which every $b_\delta := b_0 - \delta\eta$, $\delta \in (0, \Delta]$, lies in the interior of $\Delta(S)$ (for some $\Delta > 0$). Since f is convex, the directional derivatives $f'_\eta(b_\delta) = \lim_{g \to 0_+} \frac{f(b_\delta + g\eta) - f(b_\delta)}{g}$ are non-decreasing as b_δ get closer to b_0. In particular, the linear functions α_δ found for b_δ in the previous step satisfy

$$\alpha_\delta(b_0) \geq f(b_\delta) + f'_\eta(b_\delta)\delta \geq f(b_\delta) + f'_\eta(b_\Delta)\delta.$$

The right-hand side converges to $f(b_0) + f'_\eta(b_\Delta) \cdot 0 = f(b_0)$, which shows that the supremum of $\alpha_\delta(b_0)$ is at least $f(b_0)$. Since $\alpha_\delta \in \Gamma$, this proves the remaining part of the proposition. □

In the proposed algorithm, we approximate V^* using piecewise linear and convex functions. Such functions can be defined as a pointwise maximum over a *finite* set of linear functions.

[4] Recall that according to the Carathéodory's theorem, it suffices to consider finite convex combinations.

Definition 3.6 (Piecewise linear and convex function on $\Delta(S)$)**.** A function $f : \Delta(S) \to \mathbb{R}$ is said to be *piecewise linear and convex* (PWLC) if it is of the form $f(b) = \max_{\alpha \in \Gamma} \alpha(b)$ (for each b) for some finite set $\Gamma \subset \{\alpha : \Delta(S) \to \mathbb{R} \mid \alpha \text{ is linear}\}$ of linear functions.

3.3 Composing Strategies

Every behavioral strategy of the imperfectly informed player 1 can be split into the stage strategy π_1 player 1 uses in the first stage of the game, and behavioral strategies he uses in the rest of the game after reaching a subgame where he has already played action $a_1 \in A_1$ and observed $o \in O$. We can also use the inverse principle, called *strategy composition*, to form new strategies by choosing the stage strategy π_1 for the first stage and then deciding behavioral strategies for each one of the subgames.

Definition 3.7 (Strategy composition)**.** Let G be a one-sided POSG and $\pi_1 \in \Pi_1$ be a stage strategy of player 1. Furthermore, let $\overline{\zeta} \in (\Sigma_1)^{A_1 \times O}$ be a vector representing behavioral strategies of player 1 for each subgame following $a_1 \in A_1$ and $o \in O$. The *strategy composition* $\mathsf{comp}(\pi_1, \overline{\zeta})$ is a behavioral strategy of player 1 such that

$$\mathsf{comp}(\pi_1, \overline{\zeta})(\omega) = \begin{cases} \pi_1 & \omega = \varnothing \\ \zeta_{a_1,o}(\omega') & \omega = a_1 o \omega' \end{cases} \qquad \text{for each } \omega \in (A_1 O)^* \,. \tag{3.16}$$

By composing strategies $\overline{\zeta}$ using π_1, we obtain a new strategy where the probability of playing a_1 in the first stage of the game is $\pi_1(a_1)$, and strategy $\zeta_{a_1,o}$ is played after playing action a_1 and observing observation o in the first stage of the game. Importantly, the newly formed strategy $\mathsf{comp}(\pi_1, \overline{\zeta}) \in \Sigma_1$ is also a behavioral strategy of imperfectly informed player 1, and therefore the properties of strategies presented in Section 3.2 apply also to $\mathsf{comp}(\pi_1, \overline{\zeta})$. Conversely, for each strategy $\sigma_1 \in \Sigma_1$ of player 1, we can find the appropriate π_1 and $\overline{\zeta}$ such that $\sigma_1 = \mathsf{comp}(\pi_1, \overline{\zeta})$.

Proposition 3.10. *Every behavioral strategy $\sigma_1 \in \Sigma_1$ of player 1 can be represented as a strategy composition.*

Proof. Let $\sigma_1 \in \Sigma_1$ be an arbitrary behavioral strategy of player 1, and let $\pi_1 = \sigma_1(\varnothing)$ and $\zeta_{a_1,o}(\omega') = \sigma_1(\omega')$ for every $(a_1, o) \in A_1 \times O$ and $\omega' \in (A_1 O)^*$. It can be easily verified that $\mathsf{comp}(\pi_1, \overline{\zeta})$ defined in Definition 3.7 satisfies $\mathsf{comp}(\pi_1, \overline{\zeta}) = \sigma_1$. □

Importantly, we can obtain values $\mathrm{val}^{\mathsf{comp}(\pi_1, \overline{\zeta})}$ of composite strategies without considering the entire strategy $\mathsf{comp}(\pi_1, \overline{\zeta})$. Instead, it is sufficient to consider only the first

stage of the game and the *values* of the strategies $\overline{\zeta} \in (\Sigma_1)^{A_1 \times O}$ used in the subgames as the following lemma shows.

Lemma 3.11. *Let G be a one-sided POSG and $\mathsf{comp}(\pi_1, \overline{\zeta})$ a composite strategy. Then the following holds:*

$$\mathrm{val}^{\mathsf{comp}(\pi_1,\overline{\zeta})}(s) = \min_{a_2 \in A_2} \mathbb{E}_{a_1 \sim \pi_1,\, (o,s') \sim T(\cdot \mid s,a_1,a_2)} \left[R(s,a_1,a_2) + \gamma \mathrm{val}^{\zeta_{a_1,o}}(s') \right] \quad (3.17)$$

$$= \min_{a_2 \in A_2} \sum_{a_1 \in A_1} \pi_1(a_1) \left[R(s,a_1,a_2) + \gamma \sum_{(o,s) \in O \times S} T(o,s' \mid s,a_1,a_2) \mathrm{val}^{\zeta_{a_1,o}}(s') \right] .$$

The proof relies on the fact that upon reaching a subgame after player 1 played a_1 and observed o, the strategy $\zeta_{a_1,o}$ of player 1 guarantees that player 1 gets $\mathrm{val}^{\zeta_{a_1,o}}(s')$ when the given subgame starts in state s' (and player 2 uses a best-response strategy). Therefore, it is sufficient to focus on the best-response strategy of player 2 in the first stage of the game only—as the values in the rest of the game are already known.

Proof. Let us evaluate the payoff if player 2 uses a_2 in the first stage of the game given that the initial state of the game is s. The expected reward of playing action a_2 against $\mathsf{comp}(\pi_1, \overline{\zeta})$ in the first stage is $\sum_{a_1 \in A_1} \pi_1(a_1) R(s,a_1,a_2)$, i.e., the expectation over the actions player 1 can take. Now, at the beginning of the next stage, player 2 knows everything about the past stage—including action a_1 taken by player 1, observation o he received and the new state of the game s'. Therefore, player 2 knows the strategy $\zeta_{a_1,o}$ player 1 is about to use in the rest of the game. By definition of $\mathrm{val}^{\zeta_{a_1,o}}$ (Definition 3.4), the best payoff player 2 can achieve in this subgame is $\mathrm{val}^{\zeta_{a_1,o}}(s')$. After reaching the subgame, however, one stage has already passed and the rewards originally received at time t are now received at time $t+1$. To this end, $\mathrm{val}^{\zeta_{a_1,o}}(s')$ gets multiplied by γ. The probability that the subgame is reached is $\sum_{(a_1,o,s') \in A_1 \times O \times S} \pi_1(a_1) T(o,s' \mid s,a_1,a_2)$, and the expectation over $\gamma \mathrm{val}^{\zeta_{a_1,o}}(s')$ is thus computed. Player 2 chooses an action which achieves the minimum payoff which completes the proof. □

3.3.1 Generalized Strategy Composition

Lemma 3.11 suggests that we can use composition of *values* of strategies $\mathrm{val}^{\zeta_{a_1,o}}$ to form values of composite strategies $\mathrm{val}^{\mathsf{comp}(\pi_1,\overline{\zeta})}$. In this section, we relax the assumption that the linear functions $\mathrm{val}^{\zeta_{a_1,o}}$ represents values of *some* strategy to obtain a generalized principle of composition. This allows us to approximate value function V^* as a supremum of arbitrary linear functions, instead of val^{σ_1} only. Throughout the text, we will use $\mathsf{lin}_{\Delta(S)}$ to denote the set of linear functions on $\Delta(S)$ (i.e., α-vectors). We will also use the term 'linear' to refer to functions that are linear on $\Delta(S)$.

Definition 3.8 (Value composition)**.** Let $\pi_1 \in \Pi_1$ and $\overline{\alpha} \in (\mathsf{lin}_{\Delta(S)})^{A_1 \times O}$. *Value composition* $\mathsf{valcomp}(\pi_1, \overline{\alpha}) : \Delta(S) \to \mathbb{R}$ is a linear function defined by the values in vertices of the $\Delta(S)$ simplex as follows:

$$\mathsf{valcomp}(\pi_1, \overline{\alpha})(s) = \min_{a_2 \in A_2} \sum_{a_1 \in A_1} \pi_1(a_1) \Big[R(s, a_1, a_2) + \gamma \sum_{(o,s') \in O \times S} T(o, s' \mid s, a_1, a_2) \alpha_{a_1,o}(s') \Big] . \tag{3.18}$$

Observe that according to Lemma 3.11, $\mathsf{valcomp}(\pi_1, \overline{\alpha}) = \mathrm{val}^{\mathsf{comp}(\pi_1, \overline{\zeta})}$ for $\alpha_{a_1,o} = \mathrm{val}^{\zeta_{a_1,o}}$. The value composition $\mathsf{valcomp}(\pi_1, \overline{\alpha})$, however, admits arbitrary linear function $\alpha_{a_1,o}$ and not only the value $\mathrm{val}^{\zeta_{a_1,o}}$ of some strategy $\zeta_{a_1,o} \in \Sigma_1$. Importantly, as long as linear functions $\alpha_{a_1,o}$ underestimate values $\mathrm{val}^{\zeta_{a_1,o}}$ of some strategies $\zeta_{a_1,o} \in \Sigma_1$, the value composition $\mathsf{valcomp}(\pi_1, \overline{\alpha})$ underestimates value val^{σ_1} of some strategy $\sigma_1 \in \Sigma_1$:

Lemma 3.12. *Let $\pi_1 \in \Pi_1$ be a stage strategy of player 1 and let $\overline{\alpha} \in (\mathsf{lin}_{\Delta(S)})^{A_1 \times O}$ be a vector of linear functions such that for each $\alpha_{a_1,o}$ there exists a strategy $\zeta_{a_1,o} \in \Sigma_1$ with $\mathrm{val}^{\zeta_{a_1,o}} \geq \alpha_{a_1,o}$. Then there exists a strategy $\sigma_1 \in \Sigma_1$ such that $\sigma_1(\varnothing) = \pi_1$ and $\mathrm{val}^{\sigma_1} \geq \mathsf{valcomp}(\pi_1, \overline{\alpha})$.*

Proof. Let $\overline{\zeta} \in (\Sigma_1)^{A_1 \times O}$ from the assumption of the lemma, and let $\overline{\alpha}^{\zeta}$ such that $\alpha^{\zeta}_{a_1,o} = \mathrm{val}^{\zeta_{a_1,o}}$. According to the assumption we have $\alpha^{\zeta}_{a_1,o} \geq \alpha_{a_1,o}$. Replacing $\alpha_{a_1,o}$ by $\alpha^{\zeta}_{a_1,o}$ in Equation (3.18) can only increase the objective value, hence

$$\mathsf{valcomp}(\pi_1, \overline{\alpha})(s) \leq \mathsf{valcomp}(\pi_1, \overline{\alpha}^{\zeta})(s) = \mathrm{val}^{\mathsf{comp}(\pi_1, \overline{\zeta})}(s) . \tag{3.19}$$

Composite strategies are behavioral strategies of player 1, hence $\sigma_1 = \mathsf{comp}(\pi_1, \overline{\zeta})$. □

In case of value of composite strategies, we know that $\mathrm{val}^{\mathsf{comp}(\pi_1, \zeta)}$ is a δ-Lipschitz continuous linear function (since $\mathsf{comp}(\pi_1, \zeta) \in \Sigma_1$ is a behavioral strategy of player 1 and Lemma 3.5 applies). We prove, however, that as long as linear functions $\alpha_{a_1,o}$ are bounded by $L \leq \alpha_{a_1,o}(b) \leq U$ for every belief $b \in \Delta(S)$, and are therefore δ-Lipschitz continuous, the value composition $\mathsf{valcomp}(\pi_1, \overline{\alpha})$ is also δ-Lipschitz.

Lemma 3.13. *Let $\pi_1 \in \Pi_1$ and $\overline{\alpha} \in (\mathsf{lin}_{\Delta(S)})^{A_1 \times O}$ such that $L \leq \alpha^{a_1,o}(b) \leq U$ for every $b \in \Delta(S)$. Then $L \leq \mathsf{valcomp}(\pi_1, \overline{\alpha})(b) \leq U$ for every $b \in \Delta(S)$ and $\mathsf{valcomp}(\pi_1, \overline{\alpha})$ is a δ-Lipschitz continuous function.*

Proof. Since $\mathsf{valcomp}(\pi_1, \overline{\alpha})(b)$ is a convex combination of values $\mathsf{valcomp}(\pi_1, \overline{\alpha})(s)$ in the vertices of the $\Delta(S)$ simplex, it is sufficient to show that $L \leq \mathsf{valcomp}(\pi_1, \overline{\alpha})(s) \leq$

U for every state $s \in S$. Let $a_2^* \in A_2$ be the minimizing action of player 2 in Equation (3.18). It holds $\underline{r} \leq R(s, a_1, a_2^*) \leq \overline{r}$, where $\underline{r}$ and $\overline{r}$ are minimum and maximum rewards in the game. Hence $\underline{r} \leq \sum_{a_1 \in A_1} \pi_1(a_1) R(s, a_1, a_2^*) \leq \overline{r}$. Similarly, from the assumption of the lemma, we have $L \leq \alpha_{a_1,o}(s') \leq U$ and hence $L \leq \sum_{(a_1,o,s') \in A_1 \times O \times S} \pi_1(a_1) T(o, s' \mid s, a_1, a_2^*) \alpha_{a_1,o}(s') \leq U$. We will now prove that $\mathsf{valcomp}(\pi_1, \overline{\alpha})(s) \leq U$ (the proof of $\mathsf{valcomp}(\pi_1, \overline{\alpha})(s) \geq L$ is analogous):

$$\begin{aligned}\mathsf{valcomp}(\pi_1, \overline{\alpha})(s) &= \sum_{a_1 \in A_1} \pi_1(a_1) R(s, a_1, a_2^*) + \gamma \sum_{(a_1,o,s') \in A_1 \times O \times S} \pi_1(a_1) T(o, s' \mid s, a_1, a_2^*) \alpha_{a_1,o}(s') \\ &\leq \overline{r} + \gamma U = \overline{r} + \gamma \frac{\overline{r}}{1-\gamma} = U \ .\end{aligned}$$

The δ-Lipschitz continuity of $\mathsf{valcomp}(\pi_1, \overline{\alpha})$ then follows directly from Lemma 3.4. □

3.4 Bellman Equation for One-Sided POSGs

In Section 3.2, we have defined the value function V^* as the supremum over the strategies player 1 can achieve in each of the beliefs (see Definition 3.5). However, while this correctly defines the value function, it does not provide a straightforward recipe to obtaining value $V^*(b)$ for the given belief $b \in \Delta(S)$. In fact, obtaining the value for the given belief according to Definition 3.5 is as hard as solving the game itself.

In this section, we provide an alternative characterization of the optimal value function V^* inspired by the the value iteration methods, e.g., for Markov decision processes (MDPs) and their partially observable variant (POMDPs). The high-level idea behind these approaches is to start with a coarse approximation $V_0 : \Delta(S) \to \mathbb{R}$ of the value function V^*, and then iteratively improve the approximation by applying the Bellman's operator H, i.e., generate a sequence such that $V_{i+1} = HV_i$. In our case, the improvement is based on finding a new, previously unknown, strategy that achieves higher values for each of the beliefs by means of value composition principle (Definition 3.8). Throughout this section, we will consider value functions that are represented as a point-wise supremum over a (possibly infinite) set Γ of linear functions (called α-vectors), i.e.,

$$V(b) = \sup_{\alpha \in \Gamma} \alpha(b) \qquad \text{for } \Gamma \subset \{\alpha : \Delta(S) \to \mathbb{R} \mid \alpha \text{ is linear}\} \ . \tag{3.20}$$

Furthermore, without loss of generality, Proposition 3.8 allows us to assume that the set Γ is convex. For more details on this representation of value functions see Section 3.2.1.

Definition 3.9 (Max-composition)**.** Let $V : \Delta(S) \to \mathbb{R}$ be a convex continuous function and let Γ be a convex set of linear functions such that $V(b) = \sup_{\alpha \in \Gamma} \alpha(b)$. The *max-composition* operator H is defined as

$$[HV](b) = \max_{\pi_1 \in \Pi_1} \sup_{\overline{\alpha} \in \Gamma^{A_1 \times O}} \mathsf{valcomp}(\pi_1, \overline{\alpha})(b) \ . \tag{3.21}$$

We will now prove several fundamental properties of the max-composition operator H (Definition 3.9). First, we will show that the function HV resulting from applying H on a convex continuous function V is also convex and continuous. This allows us to apply the operator H repeatedly even though it is defined only for convex continuous functions. Second, we introduce equivalent formulations of the operator H which represent the solution of $[HV](b)$ in a more traditional form of finding a Nash equilibrium of a stage-game. These formulations also allow us to show that the operator H is in fact independent on the set Γ used to represent the value function V. Finally, we conclude by showing that the operator H can indeed be used to approximate the optimal value function V^*. Namely, we show that H is a contraction mapping, and repeated application of thereof hence converges to the unique fixpoint, and we show that this fixpoint is the optimal value function V^* we seek for.

Proposition 3.14. *Let $V : \Delta(S) \to \mathbb{R}$ be a convex continuous function and let Γ be a convex set of linear functions such that $V(b) = \sup_{\alpha\in\Gamma} \alpha(b)$. Then HV is also convex and continuous. Furthermore, if V is δ-Lipschitz continuous, the function HV is δ-Lipschitz continuous as well.*

Proof. According to Definition 3.9, operator H can be rewritten as a supremum over all possible value compositions:

$$[HV](b) = \max_{\pi_1\in\Pi_1} \sup_{\overline{\alpha}\in\Gamma^{A_1\times O}} \mathsf{valcomp}(\pi_1,\overline{\alpha})(b) = \sup_{(\pi_1,\overline{\alpha})\in\Pi_1\times\Gamma^{A_1\times O}} \mathsf{valcomp}(\pi_1,\overline{\alpha})(b) \text{ , and} \tag{3.22a}$$

$$[HV](b) = \sup_{\alpha\in\Gamma'} \alpha(b) \qquad \Gamma' = \left\{\mathsf{valcomp}(\pi_1,\overline{\alpha}) \mid \pi_1 \in \Pi_1, \overline{\alpha} \in \Gamma^{A_1\times O}\right\} . \tag{3.22b}$$

In Equation (3.22b), HV is represented as a point-wise supremum from a set Γ' of linear functions $\mathsf{valcomp}(\pi_1,\overline{\alpha})$, which is a convex continuous function (see Proposition 3.7).

Moreover, in case V is δ-Lipschitz continuous, the set Γ representing V can be assumed to contain only δ-Lipschitz continuous linear functions. According to Lemma 3.13, $\mathsf{valcomp}(\pi_1,\overline{\alpha})$ is δ-Lipschitz continuous for every $\pi_1 \in \Pi_1$ and $\alpha^{a_1,o} \in \Gamma$. Hence, Γ' contains δ-Lipschitz continuous linear functions only and the point-wise maximum HV over Γ' is δ-Lipschitz continuous. □

We will now prove that the max-composition operator H can be alternatively characterized using max-min and min-max optimization. Recall that $\tau(b, a_1, \pi_2, o)$ denotes the Bayesian update of belief b given that player 1 played a_1 and observed o, and player 2 is assumed to follow stage strategy π_2 in the current round (see Equation (3.6)).

Theorem 3.15. *Let $V : \Delta(S) \to \mathbb{R}$ be a convex continuous function and let Γ be a convex set of linear functions on $\Delta(S)$ such that $V(b) = \sup_{\alpha\in\Gamma} \alpha(b)$ for every belief $b \in \Delta(S)$. Then the following definitions of operator H are equivalent:*

$$[HV](b) = \max_{\pi_1\in\Delta(S)} \sup_{\overline{\alpha}\in\Gamma^{A_1\times O}} \mathsf{valcomp}(\pi_1, \overline{\alpha})(b) \tag{3.23a}$$

$$= \max_{\pi_1\in\Pi_1} \min_{\pi_2\in\Pi_2} \left[\mathbb{E}_{b,\pi_1,\pi_2}[R(s,a_1,a_2)] + \gamma \sum_{a_1,o} \mathbb{P}_{b,\pi_1,\pi_2}[a_1,o] \cdot V(\tau(b,a_1,\pi_2,o)) \right] \tag{3.23b}$$

$$= \min_{\pi_2\in\Pi_2} \max_{\pi_1\in\Pi_1} \left[\mathbb{E}_{b,\pi_1,\pi_2}[R(s,a_1,a_2)] + \gamma \sum_{a_1,o} \mathbb{P}_{b,\pi_1,\pi_2}[a_1,o] \cdot V(\tau(b,a_1,\pi_2,o)) \right] . \tag{3.23c}$$

Proof. We first prove the equality of (3.23b) and (3.23c). Let us define a payoff function $u : \Pi_1 \times \Pi_2 \to \mathbb{R}$ to be the objective of the maximin and minimax optimization in (3.23b) and (3.23c).

$$u(\pi_1,\pi_2) = \mathbb{E}_{b,\pi_1,\pi_2}[R(s,a_1,a_2)] + \gamma \sum_{a_1,o} \mathbb{P}_{b,\pi_1,\pi_2}[a_1,o] \cdot V(\tau(b,a_1,\pi_2,o)) \tag{3.24a}$$

After expanding the expectation $\mathbb{E}_{b,\pi_1,\pi_2}[R(s,a_1,a_2)]$ and expressing V as a supremum over linear functions $\alpha \in \Gamma$, we get

$$\begin{aligned} u(\pi_1,\pi_2) &= \sum_{s,a_1,a_2} b(s)\pi_1(a_1)\pi_2(a_2|s)R(s,a_1,a_2) + \\ &\quad + \gamma \sum_{a_1,o} \mathbb{P}_{b,\pi_1,\pi_2}[a_1,o] \cdot \sup_{\alpha\in\Gamma} \sum_{s'} \tau(b,a_1,\pi_2,o)(s') \cdot \alpha(s') \qquad (3.24b) \\ &= \sum_{s,a_1,a_2} b(s)\pi_1(a_1)\pi_2(a_2|s)R(s,a_1,a_2) + \qquad (3.24c) \\ &\quad + \gamma \sum_{a_1,o} \pi_1(a_1) \cdot \sup_{\alpha\in\Gamma} \sum_{s,a_2,s'} b(s)\pi_2(a_2|s)T(o,s' \mid s,a_1,a_2)\alpha(s') . \end{aligned}$$

Note that the term $\mathbb{P}_{b,\pi_1,\pi_2}[a_1,o]$ cancels out after expanding $\tau(b,a_1,\pi_2,o)$ in Equation (3.24c).

We now show that the von Neumann's minimax theorem [von Neumann, 1928; Nikaido, 1953] applies to the game with utility function u and strategy spaces Π_1 and Π_2 for player 1 and player 2, respectively. The von Neumann's minimax theorem requires that the strategy spaces Π_1 and Π_2 are convex compact sets (which is clearly the case), and that the utility function u (as characterized by Equation (3.24c)) is continuous, convex in Π_2 and concave in Π_1. We will now prove the latter and show that u is a convex-concave utility function. Clearly, for every $\pi_2 \in \Pi_2$, the function $u(\cdot,\pi_2) : \Pi_1 \to \mathbb{R}$ (where π_2 is considered constant) is linear in π_1, and hence also concave. The convexity of $u(\pi_1,\cdot) : \Pi_2 \to \mathbb{R}$ (after fixing arbitrary $\pi_1 \in \Pi_1$) is more involved. As weighted sum of convex functions with positive coefficients $\pi_1(a_1) \geq 0$ is also convex, it is sufficient to

show that $f(\pi_2) = \sup_{\alpha\in\Gamma} \sum_{s,a_2,s'} b(s)\pi_2(a_2|s)T(o,s' \mid s,a_1,a_2)\alpha(s')$ is convex. Observe that for every $\alpha \in \Gamma$, the expression $\sum_{s,a_2,s'} b(s)\pi_2(a_2|s)T(o,s' \mid s,a_1,a_2)\alpha(s')$ is linear in π_2—hence the supremum over such linear expressions in π_2 is convex in π_2 (see Proposition 3.7). According to von Neumann's minimax theorem $\max_{\pi_1\in\Pi_1} \min_{\pi_2\in\Pi_2} u(\pi_1,\pi_2) = \min_{\pi_2\in\Pi_2} \max_{\pi_1\in\Pi_1} u(\pi_1,\pi_2)$ which concludes the proof of equality of (3.23b) and (3.23c).

We now proceed by showing the equality of (3.23a) and (3.23b). By further rearranging Equation (3.24c), we get

$$u(\pi_1,\pi_2) = \sup_{\overline{\alpha}\in\Gamma^{A_1\times O}} \Big[\sum_{s,a_1,a_2} b(s)\pi_1(a_1)\pi_2(a_2|s)R(s,a_1,a_2) + \tag{3.25}$$
$$+ \gamma \sum_{a_1,o} \pi_1(a_1) \sum_{s,a_2,s'} b(s)\pi_2(a_2|s)T(o,s' \mid s,a_1,a_2)\alpha_{a_1,o}(s') \Big] .$$

Let us define a game with strategy spaces Γ and Π_2 and payoff function $u'_{\pi_1} : \Gamma \times \Pi_2 \to \mathbb{R}$ where u'_{π_1} is the objective of the supremum in Equation (3.25) (Equation (3.26b) is an algebraic simplification of Equation (3.26a)).

$$u'_{\pi_1}(\overline{\alpha},\pi_2) = \sum_{s,a_1,a_2} b(s)\pi_1(a_1)\pi_2(a_2|s)R(s,a_1,a_2) + \tag{3.26a}$$
$$+ \gamma \sum_{a_1,o} \pi_1(a_1) \sum_{s,a_2,s'} b(s)\pi_2(a_2|s)T_{s,a_1,a_2}(o,s')\alpha_{a_1,o}(s')$$
$$= \sum_s b(s) \sum_{a_2} \pi_2(a_2|s) \sum_{a_1} \pi_1(a_1) \Big[R(s,a_1,a_2) + \tag{3.26b}$$
$$+ \gamma \sum_{o,s'} T(o,s' \mid s,a_1,a_2)\alpha_{a_1,o}(s') \Big] .$$

Plugging (3.26b) into (3.25), we can write

$$\max_{\pi_1\in\Pi_1} \min_{\pi_2\in\Pi_2} u(\pi_1,\pi_2) = \max_{\pi_1\in\Pi_1} \min_{\pi_2\in\Pi_2} \sup_{\overline{\alpha}\in\Gamma^{A_1\times O}} u'_{\pi_1}(\pi_2,\overline{\alpha}) . \tag{3.27}$$

To prove the equivalence of (3.23a) and (3.23b), we need to show that the minimum and supremum can be swapped. Since u'_{π_1} is linear in both π_2 and $\overline{\alpha}$, Π_2 is a compact convex set and Γ (and therefore also the set of mappings $\overline{\alpha} \in \Gamma^{A_1\times O}$) is convex, it is possible to apply Sion's minimax theorem [Sion, 1958] to get

$$\max_{\pi_1\in\Pi_1} \min_{\pi_2\in\Pi_2} \sup_{\overline{\alpha}\in\Gamma^{A_1\times O}} u'_{\pi_1}(\pi_2,\overline{\alpha}) = \max_{\pi_1\in\Pi_1} \sup_{\overline{\alpha}\in\Gamma^{A_1\times O}} \min_{\pi_2\in\Pi_2} u'_{\pi_1}(\pi_2,\overline{\alpha}) . \tag{3.28}$$

As u'_{π_1} is linear in π_2 (when π_1 and $\overline{\alpha}$ are fixed), the minimum over π_2 is attained in pure strategies. Denote $\hat{\pi}_2 : S \to A_2$ a pure strategy of player 2 assigning action $\hat{\pi}_2(s)$ to be played in state s, and $\hat{\Pi}_2$ the set of all pure strategies of player 2. We now rewrite u'_{π_1} to use pure strategies $\hat{\Pi}_2$ instead of randomized stage strategies Π_2. First, in Equation (3.29a), we replace the maximization over Π_2 by maximization over the pure strategies $\hat{\Pi}_2$ and replace expectation over actions of player 2 by using the deterministic action $\hat{\pi}_2(s)$ where appropriate. Then, in Equation (3.29b), we leverage the fact that

player 2 *knows* the state before having to act, and hence he can optimize his actions $\hat{\pi}_2(s)$ independently. And, finally, in Equation (3.29c), we use Definition 3.8.

$$\max_{\pi_1\in\Pi_1}\min_{\pi_2\in\Pi_2} u(\pi_1,\pi_2) = \max_{\pi_1\in\Pi_1}\sup_{\overline{\alpha}\in\Gamma^{A_1\times O}}\min_{\pi_2\in\Pi_2} u'_{\pi_1}(\pi_2,\overline{\alpha}) =$$

$$= \max_{\pi_1\in\Pi_1}\sup_{\overline{\alpha}\in\Gamma^{A_1\times O}}\min_{\hat{\pi}_2\in\hat{\Pi}_2}\sum_s b(s)\sum_{a_1}\pi_1(a_1)\Big[R(s,a_1,\hat{\pi}_2(s)) + \tag{3.29a}$$

$$+\gamma\sum_{o,s'} T(o,s' \mid s,a_1,\hat{\pi}_2(s))\alpha_{a_1,o}(s')\Big]$$

$$= \max_{\pi_1\in\Pi_1}\sup_{\overline{\alpha}\in\Gamma^{A_1\times O}}\sum_s b(s)\min_{\hat{\pi}_2(s)\in A_2}\sum_{a_1}\pi_1(a_1)\Big[R(s,a_1,\hat{\pi}_2(s)) + \tag{3.29b}$$

$$+\gamma\sum_{o,s'} T(o,s' \mid s,a_1,\hat{\pi}_2(s))\alpha_{a_1,o}(s')\Big]$$

$$= \max_{\pi_1\in\Pi_1}\sup_{\overline{\alpha}\in\Gamma^{A_1\times O}}\sum_s b(s)\cdot\mathsf{valcomp}(\pi_1,\overline{\alpha})(s) = \max_{\pi_1\in\Pi_1}\sup_{\overline{\alpha}\in\Gamma^{A_1\times O}}\mathsf{valcomp}(\pi_1,\overline{\alpha})(b)\ . \tag{3.29c}$$

This concludes the proof of the equality of Equations (3.23a) and (3.23b). □

Corollary 3.16. *Bellman's operator H does not depend on the convex set Γ of linear functions used to represent the convex value function V.*

Since the maximin and minimax values of the game (Equations (3.23b) and (3.23c), respectively) coincide, the value $[HV](b)$ corresponds to the Nash equilibrium in the stage game. We define the stage game formally.

Definition 3.10 (Stage game)**.** A *stage game* with respect to a convex continuous value function $V:\Delta(S)\to\mathbb{R}$ and belief $b\in\Delta(S)$ is a two-player zero sum game with strategy spaces Π_1 for the maximizing player 1 and Π_2 for the minimizing player 2, and payoff function

$$u^{V,b}(\pi_1,\pi_2) = \mathbb{E}_{b,\pi_1,\pi_2}[R(s,a_1,a_2)] + \gamma\sum_{a_1,o}\mathbb{P}_{b,\pi_1,\pi_2}[a_1,o]\cdot V(\tau(b,a_1,\pi_2,o))\ . \tag{3.30}$$

With a slight abuse of notation, we use $[HV](b)$ to refer both to the max-composition operator (Definition 3.9) as well as to this stage game.

We will now show that the Bellman's operator H is a contraction mapping. Recall that the mapping H is a contraction, if there exists $0\le k<1$ such that $\|HV_1-HV_2\|\le k\|V_1-V_2\|$. We consider uniform metric $\|\cdot\|_\infty$ such that $\|V_1-V_2\|_\infty=\max_{b\in\Delta(S)}|V_1(b)-V_2(b)|$. First, we focus on a single belief point and establish condition that guarantee that $|HV_1(b)-HV_2(b)|\le k|V_1(b)-V_2(b)|$. Then, we show this condition directly implies contractivity of H.

Lemma 3.17. *Let $V, W : \Delta(S) \to \mathbb{R}$ be two convex continuous value functions and $b \in \Delta(S)$ a belief such that $[HV](b) \leq [HW](b)$. Let (π_1^V, π_2^V) and (π_1^W, π_2^W) be Nash equilibrium strategy profiles in stage games $[HV](b)$ and $[HW](b)$, respectively, and $C \geq 0$. If $W(\tau(b, a_1, o, \pi_2^V)) - V(\tau(b, a_1, o, \pi_2^V)) \leq C$ for every action $a_1 \in \mathsf{Supp}(\pi_1^W)$ of player 1 and every observation $o \in O$ such that $\mathbb{P}_{b,\pi_1^W,\pi_2^V}[o \,|\, a_1] > 0$, then $[HW](b) - [HV](b) \leq \gamma C$.*

Proof. By deviating from the equilibrium strategy profiles in stage games $[HV](b)$ and $[HW](b)$, the players can only worsen their payoffs. Therefore, we have

$$\begin{aligned} u^{V,b}(\pi_1^W, \pi_2^V) \leq u^{V,b}(\pi_1^V, \pi_2^V) = [HV](b) \leq \\ \leq [HW](b) = u^{W,b}(\pi_1^W, \pi_2^W) \leq u^{W,b}(\pi_1^W, \pi_2^V) \ . \end{aligned} \tag{3.31}$$

We can thus bound the difference $[HW](b) - [HV](b)$ by $u^{W,b}(\pi_1^W, \pi_2^V) - u^{V,b}(\pi_1^W, \pi_2^V)$ where, according to Definition 3.10,

$$\begin{aligned} u^{W,b}(\pi_1^W, \pi_2^V) - u^{V,b}(\pi_1^W, \pi_2^V) = \\ = \gamma \sum_{a_1, o} \mathbb{P}_{b,\pi_1^W,\pi_2^V}[a_1, o] \cdot [W(\tau(b, a_1, \pi_2^V, o)) - V(\tau(b, a_1, \pi_2^V, o))] \ . \end{aligned} \tag{3.32}$$

Since every $W(\tau(b, a_1, o, \pi_2^V)) - V(\tau(b, a_1, o, \pi_2^V)$ considered in Equation (3.32) with non-zero probability $\mathbb{P}_{b,\pi_1^W,\pi_2^V}[a_1, o]$ is assumed to be bounded by C, also the expectation over such $W(\tau(b, a_1, o, \pi_2^V)) - V(\tau(b, a_1, o, \pi_2^V)$ is bounded by C. Hence $u^{W,b}(\pi_1^W, \pi_2^V) - u^{V,b}(\pi_1^W, \pi_2^V) \leq \gamma C$, and also $[HW](b) - [HV](b) \leq \gamma C$. □

Theorem 3.18. *Operator H is a contraction mapping in the space of convex continuous functions $V : \Delta(S) \to \mathbb{R}$ with contractivity factor γ under max-norm. Hence V^* is the unique fixpoint of H, and every sequence $\{V_i\}_{i=0}^{\infty}$ of value functions such that $V_i = HV_{i-1}$ converges to V^*.*

Proof. Let $V, W : \Delta(S) \to \mathbb{R}$ be convex functions such that $\|V - W\|_\infty = \max_{b \in \Delta(S)} |V(b) - W(b)| \leq C$. To prove the contractivity of H, it suffices to show that $\|HV - HW\|_\infty \leq \gamma C$, i.e., $|[HV](b) - [HW](b)| \leq \gamma C$ for every belief $b \in \Delta(S)$. Since $|V(b) - W(b)| \leq C$ for every belief $b \in \Delta(S)$, both $HV(b) - HW(b) \leq \gamma C$ and $HW(b) - HV(b) \leq \gamma C$ according to Lemma 3.17. Hence also $|HV(b) - HW(b)| \leq \gamma C$ which completes the proof of contractivity of H. The uniqueness of the fixed-point and the convergence properties follow directly from Banach's fixed-point theorem [Ciesielski et al., 2007], while in Lemma 3.19 we show that V^* is the fixpoint. □

Finally, we will prove that the optimal value function from Definition 3.5 is the fixpoint of the Bellman's operator H. Hence, according to Theorem 3.18, we can iteratively generate sequences of value functions to gradually improve the approximation of V^* by means of the operator H.

Lemma 3.19. *Optimal value function V^* satisfies $V^* = HV^*$.*

Proof. According to Corollary 3.16, the Bellman's operator does not depend on the set Γ used to represent the value function V^*. To this end, we will assume that the set Γ used to represent V^* is

$$\Gamma = \mathsf{Conv}\{\mathrm{val}^{\sigma_1} \mid \sigma_1 \in \Sigma_1\} \ . \tag{3.33a}$$

To prove the equivalence of value functions V^* and HV^* we consider that these functions are represented as follows:

$$V^*(b) = \sup_{\alpha \in \Gamma_{V^*}} \alpha(b) \qquad \Gamma_{V^*} = \{\mathrm{val}^{\sigma_1} \mid \sigma_1 \in \Sigma_1\} \tag{3.33b}$$

$$[HV^*](b) = \sup_{\alpha \in \Gamma_{HV^*})} \qquad \Gamma_{HV^*} = \left\{\mathsf{valcomp}(\pi_1, \overline{\alpha}) \mid \pi_1 \in \Pi_1, \overline{\alpha} \in \Gamma^{A_1 \times O}\right\} \ . \tag{3.33c}$$

To prove the equivalence of V^* and HV^*, it suffices to show that for every $\alpha \in \Gamma_{V^*}$ there exists $\alpha' \in \Gamma_{HV^*}$ such that $\alpha' \geq \alpha$, and vice versa.

First, from Proposition 3.10, Lemma 3.11 and Definition 3.8, it follows that every strategy $\sigma_1 \in \Sigma_1$ can be represented as a value composition $\mathsf{valcomp}(\pi_1, \overline{\zeta})$, and we have

$$\mathrm{val}^{\sigma_1} = \mathrm{val}^{\mathsf{comp}(\pi_1, \overline{\zeta})} = \mathsf{valcomp}(\pi_1, \overline{\alpha}^{\overline{\zeta}}) \tag{3.33d}$$

where $\alpha^{\overline{\zeta}}_{a_1,o} = \mathrm{val}^{\overline{\zeta}_{a_1,o}} \in \Gamma$. Hence $\mathrm{val}^{\sigma_1} = \mathsf{valcomp}(\pi_1, \overline{\zeta}) \in \Gamma_{HV^*}$.

The opposite direction of the proof, i.e., that for every $\alpha \in \Gamma_{HV^*}$ there exists $\alpha' \in \Gamma_{V^*}$ such that $\alpha' \geq \alpha$, is more involved. Let $\alpha = \mathsf{valcomp}(\pi_1, \overline{\alpha}) \in \Gamma_{HV^*}$ be arbitrary. From (3.33c), each $\alpha_{a_1,o}$ can be written as a convex combination of finitely many elements of $\{\mathrm{val}^{\sigma_1} \mid \sigma_1 \in \Sigma_1\}$.

$$\alpha_{a_1,o} = \sum_{i=1}^{K} \lambda_i^{a_1,o} \mathrm{val}^{\sigma_1^{a_1,o,i}} \tag{3.33e}$$

Let us form a vector of strategies $\overline{\zeta} \in (\Sigma_1)^{A_1 \times O}$ such that each $\zeta_{a_1,o}$ is a convex combination of strategies $\sigma_1^{a_1,o,i}$ using coefficients from Equation (3.33e),

$$\zeta_{a_1,o} = \sum_{i=1}^{K} \lambda_i^{a_1,o} \sigma_1^{a_1,o,i} \ . \tag{3.33f}$$

We can interpret strategy $\zeta_{a_1,o}$ as player 1 first randomly choosing among strategies $\sigma_1^{a_1,o,i}$, and then following the chosen strategy in the rest of the game. If the player 2 knew which strategy $\sigma_1^{a_1,o,i}$ has been chosen, he is able to achieve utility $\mathrm{val}^{\sigma_1^{a_1,o,i}}$. However, he has no access to this information, and hence $\mathrm{val}^{\zeta^{a_1,o}} \geq \sum_{i=1}^{K} \lambda_i^{a_1,o} \mathrm{val}^{\sigma_1^{a_1,o,i}} = \alpha^{a_1,o}$. Now, we have

$$\alpha' = \mathrm{val}^{\mathsf{comp}(\pi_1, \overline{\zeta})} \geq \mathsf{valcomp}(\pi_1, \overline{\alpha}) = \alpha \tag{3.33g}$$

which concludes the proof. □

3.5 Exact Value Iteration

In Section 3.4, we have shown that the optimal value function can be approximated by means of composing strategies in the sense of max-composition introduced in Definition 3.9. In this section, we provide a linear programming formulation to perform such optimal composition for value functions that are piecewise linear and convex, i.e., can be represented as a point-wise maximum of a finite set Γ of linear functions. Furthermore, we show that as long as the value function V is piecewise linear and convex, HV is also piecewise linear and convex. This allows for using the same linear program iteratively to approximate the optimal value function V^* by means of constructing a sequence of piecewise linear and convex value functions $\{V_i\}_{i=1}^{\infty}$ such that $V_i = HV_{i-1}$.

3.5.1 Computing Max-compositions

In order to compute HV given a piecewise linear and convex (PWLC) value function V, it is essential to solve Equation (3.21). Every PWLC value function can be represented as a point-wise maximum over a finite set of linear functions $\{\alpha_1, \ldots, \alpha_k\}$ (see Definition 3.6). Without loss of generality, we consider that the set Γ used to represent the value function V is the convex hull of the aforementioned set:

$$\Gamma \coloneqq \mathsf{Conv}\left(\{\alpha_1, \ldots, \alpha_k\}\right) = \left\{ \sum_{i=1}^{k} \lambda_i \alpha_i \mid \lambda \in \mathbb{R}^k_{\geq 0}, \|\lambda\|_1 = 1 \right\} . \tag{3.34}$$

Recall that convexifying the set of linear functions used to represent V does not affect the values V attains (see Proposition 3.8). The set Γ is a convex compact set, and we have

$$[HV](b) = \max_{\pi_1 \in \Pi_1} \sup_{\overline{\alpha} \in \Gamma^{A_1 \times O}} \mathsf{valcomp}(\pi_1, \overline{\alpha})(b) \tag{3.35a}$$

$$= \max_{\pi_1 \in \Pi_1} \max_{\overline{\alpha} \in \Gamma^{A_1 \times O}} \mathsf{valcomp}(\pi_1, \overline{\alpha})(b) \tag{3.35b}$$

$$= \max_{\pi_1 \in \Pi_1} \max_{\overline{\alpha} \in \Gamma^{A_1 \times O}} \sum_{s \in S} b(s) \cdot \mathsf{valcomp}(\pi_1, \overline{\alpha})(s) \tag{3.35c}$$

$$= \max_{\pi_1 \in \Pi_1} \max_{\overline{\alpha} \in \Gamma^{A_1 \times O}} \sum_{s \in S} b(s) \cdot \min_{a_2} \left[\sum_{a_1} \pi_1(a_1) R(s, a_1, a_2) + \right. \tag{3.35d}$$

$$\left. + \gamma \sum_{(a_1, o, s') \in A_1 \times O \times S} T(o, s' \mid s, a_1, a_2) \pi_1(a_1) \alpha^{a_1, o}(s') \right] .$$

Equation (3.35b) follows from the fact that $\mathsf{valcomp}(\pi_1, \overline{\alpha})$ is continuous in $\overline{\alpha}$, and Γ is a compact set (and hence is also $\Gamma^{A_1 \times O}$). The Equation (3.35c) represents value of the linear function $\mathsf{valcomp}(\pi_1, \overline{\alpha})$ as the convex combination of its values in the vertices of the $\Delta(S)$ simplex, and, finally, Equation (3.35d) rewrites $\mathsf{valcomp}(\pi_1, \overline{\alpha})(s)$ using Definition 3.8.

Equation (3.35d) can be directly formalized as a mathematical program in (3.36). The minimization over $a_2 \in A_2$ is rewritten as a set of constraints for each value of state $V(s)$ (one for each action $a_2 \in A_2$ of player 2) in Equation (3.36b). The convexification of set $\{\alpha_1, \ldots, \alpha_k\}$ is represented by (3.36c) where variables $\lambda_i^{a_1,o}$ represent coefficients of the convex combination. The stage strategy π_1 is characterized by (3.36e) and (3.36f).

$$\max_{\pi_1,\lambda,\overline{\alpha},V} \sum_{s\in S} b(s)\cdot V(s) \tag{3.36a}$$

$$\text{s.t.} \quad V(s) \le \sum_{a_1\in A_1} \pi_1(a_1)R(s,a_1,a_2) + \qquad \forall (s,a_2)\in S\times A_2 \tag{3.36b}$$
$$+\gamma \sum_{(a_1,o,s')\in A_1\times O\times S} T(o,s' \mid s,a_1,a_2)\pi_1(a_1)\alpha^{a_1,o}(s')$$

$$\alpha^{a_1,o}(s') = \sum_{i=1}^{k} \lambda_i^{a_1,o}\cdot \alpha_i(s') \qquad \forall (a_1,o,s')\in A_1\times O\times S \tag{3.36c}$$

$$\sum_{i=1}^{k} \lambda_i^{a_1,o} = 1 \qquad \forall (a_1,o)\in A_1\times O \tag{3.36d}$$

$$\sum_{a_1\in A_1} \pi_1(a_1) = 1 \tag{3.36e}$$

$$\pi_1(a_1) \ge 0 \qquad \forall a_1 \in A_1 \tag{3.36f}$$

$$\lambda_i^{a_1,o} \ge 0 \qquad \forall (a_1,o)\in A_1\times O, 1\le i\le k \tag{3.36g}$$

The mathematical program (3.36) is not linear since it contains a product of variables $\pi_1(a)\cdot\alpha^{a_1,o}(s')$. The program can be, however, linearized by introducing substitution $\hat{\alpha}^{a_1,o}(s') = \pi_1(a_1)\alpha^{a_1,o}(s')$ and $\hat{\lambda}_i^{a_1,o} = \pi_1(a_1)\lambda_i^{a_1,o}$. The resulting linear programming formulation of max-composition $[HV](b)$ is shown in (3.37).

$$\max_{\pi_1,\lambda,\overline{\alpha},V} \sum_{s\in S} b(s)\cdot V(s) \tag{3.37a}$$

$$\text{s.t.} \quad V(s) \le \sum_{a_1\in A_1} \pi_1(a_1)R(s,a_1,a_2) + \qquad \forall (s,a_2)\in S\times A_2 \tag{3.37b}$$
$$+\gamma \sum_{(a_1,o,s')\in A_1\times O\times S} T(o,s' \mid s,a_1,a_2)\hat{\alpha}^{a_1,o}(s')$$

$$\hat{\alpha}^{a_1,o}(s') = \sum_{i=1}^{k} \hat{\lambda}_i^{a_1,o}\cdot \alpha_i(s') \qquad \forall (a_1,o,s')\in A_1\times O\times S \tag{3.37c}$$

$$\sum_{i=1}^{k} \hat{\lambda}_i^{a_1,o} = \pi_1(a_1) \qquad \forall (a_1,o)\in A_1\times O \tag{3.37d}$$

$$\sum_{a_1\in A_1} \pi_1(a_1) = 1 \tag{3.37e}$$

$$\pi_1(a_1) \ge 0 \qquad \forall a_1 \in A_1 \tag{3.37f}$$

$$\hat{\lambda}_i^{a_1,o} \ge 0 \qquad \forall (a_1,o)\in A_1\times O, 1\le i\le k \tag{3.37g}$$

For the sake of completeness, we provide a dual formulation of the linear program (3.37) (with some minor modifications to improve readability).

$$\min_{V,\pi_2,\hat{\tau}} V \tag{3.38a}$$

$$\text{s.t. } V \geq \sum_{(s,a_2)\in S\times A_2} \pi_2(s \wedge a_2) R(s,a_1,a_2) + \gamma \sum_{o\in O} \hat{V}(a_1,o) \qquad \forall a_1 \tag{3.38b}$$

$$\hat{V}(a_1,o) \geq \sum_{s'\in S} \hat{\tau}(b,a_1,o,\pi_2)(s') \cdot \alpha_i(s') \qquad \forall (a_1,o), 1 \leq i \leq k \tag{3.38c}$$

$$\hat{\tau}(b,a_1,\pi_2,o)(s') = \sum_{(s,a_2)\in S\times A_2} T(o,s' \mid s,a_1,a_2)\pi_2(s \wedge a_2) \qquad \forall (a_1,o,s') \tag{3.38d}$$

$$\sum_{a_2\in A_2} \pi_2(s \wedge a_2) = b(s) \qquad \forall s \tag{3.38e}$$

$$\pi_2(s \wedge a_2) \geq 0 \qquad \forall (s,a_2) \tag{3.38f}$$

Here, the stage strategy of player 2 is represented as a joint probability $\pi_2(s \wedge a_2)$ of playing action $a_2 \in A_2$ while being in state $s \in S$ (i.e., $\pi_2(a_2 \mid s) = \pi_2(s \wedge a_2)/b(s)$ where applicable). Player 1 then seeks the best response $a_1 \in A_1$ (constraint (3.38b)) that maximizes the sum of expected immediate reward and γ-discounted utility in the subgames after playing action a_1 and seeing observation $o \in O$. The beliefs $\tau(b,a_1,\pi_2,o)$ in the subgames are multiplied by the probability of reaching the subgame (i.e., there is no division by $\mathbb{P}_{b,a_1,\pi_2}[a_1,o]$ in Equation (3.38d)), hence also the values of subgames $V(a_1,o)$ need not be multiplied by $\mathbb{P}_{b,a_1,\pi_2}[a_1,o]$. The value of a subgame $V(a_1,o)$ is expressed as a maximum $\max_{\alpha\in\Gamma} \alpha(\tau(b,a_1,\pi_2,o))$ expressed by constraints (3.38c).

3.5.2 Value Iteration

To apply linear program (3.37) repeatedly to enable a value iteration algorithm, we require that every V_i in the sequence $\{V_i\}_{i=0}^{\infty}$, starting from an arbitrary PWLC value function V_0, is also piecewise linear and convex. By Theorem 3.20 this is always the case.

Theorem 3.20. *Let V be a piecewise linear and convex function. Then HV is also piecewise linear and convex.*

Proof. Consider the linear program (3.37) which computes the optimal value composition $\mathsf{valcomp}(\pi_1,\overline{\alpha})$ in $[HV](b)$ according to Definition 3.9. The polytope of feasible solutions of the linear program defined by the constraints (3.37b)–(3.37g) is independent of belief b (which only appears in the objective (3.37a)). Therefore, the set Q of vertices of this polytope is also independent of belief $b \in \Delta(S)$. The optimal solution of a linear programming problem (3.37) representing $[HV](b)$ can be found within the vertices Q of the polytope of feasible solutions [Vanderbei, 2015]. There is a finite number of vertices $q \in Q$, and each vertex $q \in Q$ corresponds to some assignment of variables defining the value composition $\mathsf{valcomp}(\pi_1^q,\overline{\alpha}^q)$. Since the set Q of the vertices of the polytope is

independent of the belief b, we have

$$[HV](b) = \max_{q \in Q} \mathsf{valcomp}(\pi_1^q, \overline{\alpha}^q) \ . \tag{3.39}$$

The set Q is finite and hence the Equation (3.39) defines a piecewise linear and convex function HV. □

The proof of Theorem 3.20 provides a straightforward algorithm to generate the set Γ' of linear functions that support HV. For every vertex $q \in Q$ of the polytope of the linear program (3.37), we are able to generate the corresponding value composition $\mathsf{valcomp}(\pi_1^q, \overline{\alpha}^q)$. The value function HV then satisfies

$$[HV](b) = \max_{\alpha \in \Gamma'} \alpha(b) \qquad \Gamma' = \{\mathsf{valcomp}(\pi_1^q, \overline{\alpha}^q) \mid q \in Q\} \ . \tag{3.40}$$

A more efficient algorithm can be devised based on, e.g., the linear support algorithm for POMDPs [Cheng, 1988]. Here, the set Γ' of linear functions defining HV is constructed incrementally until it is provably sufficient to represent value function HV. Exact value iteration algorithms to solve POMDPs are, however, generally considered to be capable of solving problems of very small sizes only. We cannot, therefore, expect decent performance of such approaches when solving one-sided POSGs that are more general than POMDPs. To this end, we provide a point-based approach to solve one-sided POSGs in Section 3.6.

3.6 Heuristic Search Value Iteration for OS-POSGs

In this section, we provide a scalable algorithm to solve one-sided POSGs inspired by the *heuristic search value iteration* (HSVI) algorithm [Smith and Simmons, 2004, 2005] for approximating value function of POMDPs presented in Section 2.2. Our algorithm approximates the convex optimal value function V^* using a pair of piecewise linear and convex value functions V_{LB}^{Γ} (lower bound on V^*) and $V_{\mathrm{UB}}^{\Upsilon}$ (upper bound on V^*). These bounds are refined over time and, given the initial belief b^{init} and the desired precision $\varepsilon > 0$, the algorithm is guaranteed to approximate the value $V^*(b^{\mathrm{init}})$ within ε. In Section 3.7, we show that this process also generates value functions that allow us to extract ε-Nash equilibrium strategies of the game.

We first show the approximation schemes used to represent V_{LB}^{Γ} and $V_{\mathrm{UB}}^{\Upsilon}$, and the methods to initialize these bounds (Section 3.6.1). The bounds induced by functions V_{LB}^{Γ} and $V_{\mathrm{UB}}^{\Upsilon}$ are refined by means of so-called point-based updates that are discussed in Section 3.6.2. Finally, in Section 3.6.3 we state the algorithm and we prove its correctness.

3.6.1 Value Function Representations

Following the results on POMDPs and the original HSVI algorithm [Hauskrecht, 2000; Smith and Simmons, 2004], we use two distinct methods to represent upper and lower PWLC bounds on V^*.

Lower bound V^{Γ}_{LB} Similarly as in the previous sections, the lower bound $V^{\Gamma}_{\mathrm{LB}} : \Delta(S) \to \mathbb{R}$ is represented as a point-wise maximum over a finite set Γ of linear functions called α-vectors, i.e., $V^{\Gamma}_{\mathrm{LB}}(b) = \max_{\alpha\in\Gamma} \alpha(b)$. Each $\alpha \in \Gamma$ is a linear function $\alpha : \Delta(S) \to \mathbb{R}$ represented by its values $\alpha(s)$ in the vertices of the $\Delta(S)$ simplex, i.e., $\alpha(b) = \sum_{s\in S} b(s) \cdot \alpha(s)$.

Upper bound $V^{\Upsilon}_{\mathrm{UB}}$ Upper bound $V^{\Upsilon}_{\mathrm{UB}} : \Delta(S) \to \mathbb{R}$ is represented as a lower convex hull of a set of points $\Upsilon = \{(b_i, y_i) \mid 1 \le i \le k\}$. Each point $(b_i, y_i) \in \Upsilon$ provides an upper bound y_i on the value $V^*(b_i)$ in belief b_i, i.e., $y_i \ge V^*(b_i)$. Since the value function V^* is convex, it holds that

$$V^*\left(\sum_{i=1}^{k} \lambda_i b_i\right) \le \sum_{i=1}^{k} \lambda_i \cdot V^*(b_i) \le \sum_{i=1}^{k} \lambda_i \cdot y_i \qquad \text{for every } \lambda \in \mathbb{R}^k_{\ge 0} \text{ such that } \sum_{i=1}^{k} \lambda_i = 1\ . \tag{3.41}$$

This fact is used in the first variant of the HSVI algorithm (HSVI1 [Smith and Simmons, 2004]) to obtain the value of the upper bound $V^{\Upsilon}_{\mathrm{HSVI1}}(b)$ for belief b: A linear program can be used to find coefficients $\lambda \in \mathbb{R}^k_{\ge 0}$ such that $b = \sum_{i=1}^{k} \lambda_i \cdot y_i$ holds and $\sum_{i=1}^{k} \lambda_i \cdot y_i$ is minimal:

$$V^{\Upsilon}_{\mathrm{HSVI1}}(b) = \min\left\{\sum_{i=1}^{k} \lambda_i y_i \mid \lambda \in \mathbb{R}^k_{\ge 0} : \sum_{i=1}^{k} \lambda_i = 1 \ \wedge\ \sum_{i=1}^{k} \lambda_i b_i = b\right\}\ , \tag{3.42}$$

In the latter proof of the Theorem 3.25 showing the correctness of the algorithm, we require the bounds V^{Γ}_{LB} and $V^{\Upsilon}_{\mathrm{UB}}$ to be δ-Lipschitz continuous. Since this needs not hold for $V^{\Upsilon}_{\mathrm{HSVI1}}$, we define $V^{\Upsilon}_{\mathrm{UB}}$ as a lower δ-Lipschitz envelope of $V^{\Upsilon}_{\mathrm{HSVI1}}$:

$$V^{\Upsilon}_{\mathrm{UB}}(b) = \min_{b'\in\Delta(S)} \left[V^{\Upsilon}_{\mathrm{HSVI1}}(b') + \delta\|b - b'\|_1\right]\ . \tag{3.43}$$

This computation can be expressed as a linear programming problem

$$V^{\Upsilon}_{\mathrm{UB}}(b) = \min_{\lambda,\Delta,b'} \sum_{i=1}^{k} \lambda_i y_i + \delta \sum_{s\in S} \Delta_s \tag{3.44a}$$

$$\text{s.t.} \sum_{i=1}^{k} \lambda_i b_i(s) = b'(s) \qquad \forall s \in S \tag{3.44b}$$

$$\Delta_s \ge b'(s) - b(s) \qquad \forall s \in S \tag{3.44c}$$

$$\Delta_s \ge b(s) - b'(s) \qquad \forall s \in S \tag{3.44d}$$

$$\sum_{i=1}^{k} \lambda_i = 1 \tag{3.44e}$$

$$\lambda_i \ge 0 \qquad \forall 1 \le i \le k \tag{3.44f}$$

Here, we have $\Delta_s = |b'(s) - b(s)|$ (and hence $\sum_{s\in S} \Delta_s = \|b - b'\|_1$). We now prove that the function $V^{\Upsilon}_{\mathrm{UB}}$ represents an upper bound on V^*.

Lemma 3.21. *Let* $\Upsilon = \{(b_i, y_i) \,|\, 1 \leq i \leq k\}$ *such that* $y_i \geq V^*(b_i)$ *for every* $1 \leq i \leq k$. *Then* $V^*(b) \leq V^{\Upsilon}_{\mathrm{UB}}(b) \leq V^{\Upsilon}_{\mathrm{HSVI1}}(b)$ *for every* $b \in \Delta(S)$. *Furthermore, value function* $V^{\Upsilon}_{\mathrm{UB}}$ *is* δ*-Lipschitz continuous.*

Proof. We first prove $V^{\Upsilon}_{\mathrm{UB}}(b) \leq V^{\Upsilon}_{\mathrm{HSVI1}}(b)$ for every $b \in \Delta(S)$. This is clearly the case from Equation (3.43):

$$V^{\Upsilon}_{\mathrm{UB}}(b) = \min_{b' \in \Delta(S)} [V^{\Upsilon}_{\mathrm{HSVI1}}(b') + \delta \|b - b'\|_1] \leq V^{\Upsilon}_{\mathrm{HSVI1}}(b) + \delta \|b - b\|_1 = V^{\Upsilon}_{\mathrm{HSVI1}}(b) \ . \quad (3.45)$$

Proving $V^*(b) \leq V^{\Upsilon}_{\mathrm{UB}}(b)$ is more involved. Consider that $\lambda \in \mathbb{R}^k$ is the minimizer of the linear program (3.44) corresponding to the solution of $V^{\Upsilon}_{\mathrm{UB}}(b)$, i.e.,

$$V^{\Upsilon}_{\mathrm{UB}}(b) = \sum_{i=1}^{k} \lambda_i y_i + \delta \left\| b - \sum_{i=1}^{k} \lambda_i b_i \right\|_1 \quad (3.46)$$

holds. According to the assumption of Lemma 3.21, for every $(b_i, y_i) \in \Upsilon$, $V^*(b_i) \leq y_i$ holds. Since V^* is convex and δ-Lipschitz continuous (Lemma 3.3 and Theorem 3.6), we have

$$\begin{aligned} V^*(b) &\leq V^* \left(\sum_{i=1}^{k} \lambda_i b_i \right) + \delta \left\| b - \sum_{i=1}^{k} \lambda_i b_i \right\|_1 \leq \\ &\leq \sum_{i=1}^{k} \lambda_i V^*(b_i) + \delta \left\| b - \sum_{i=1}^{k} \lambda_i b_i \right\|_1 \leq \sum_{i=1}^{k} \lambda_i y_i + \delta \left\| b - \sum_{i=1}^{k} \lambda_i b_i \right\|_1 = V^{\Upsilon}_{\mathrm{UB}}(b) \ . \end{aligned}$$

Finally, let us prove that $V^{\Upsilon}_{\mathrm{UB}}$ is δ-Lipschitz continuous. Let us consider beliefs $b_1, b_2 \in \Delta(S)$. Without loss of generality, assume that $V^{\Upsilon}_{\mathrm{UB}}(b_1) \geq V^{\Upsilon}_{\mathrm{UB}}(b_2)$. Let $b_{\arg\min}$ be the minimizer of $V^{\Upsilon}_{\mathrm{UB}}(b_2)$, i.e.,

$$b_{\arg\min} = \arg\min_{b'} [V^{\Upsilon}_{\mathrm{HSVI1}}(b') + \delta \|b_2 - b'\|_1]. \quad (3.47)$$

Due to triangle inequality, we have

$$\begin{aligned} V^{\Upsilon}_{\mathrm{UB}}(b_1) &= \min_{b' \in \Delta(S)} [V^{\Upsilon}_{\mathrm{HSVI1}}(b') + \delta \|b_1 - b'\|_1] \leq V^{\Upsilon}_{\mathrm{HSVI1}}(b_{\arg\min}) + \delta \|b_1 - b_{\arg\min}\|_1 \leq \\ &\leq [V^{\Upsilon}_{\mathrm{HSVI1}}(b_{\arg\min}) + \delta \|b_2 - b_{\arg\min}\|_1] + \delta \|b_1 - b_2\|_1 = V^{\Upsilon}_{\mathrm{UB}}(b_2) + \delta \|b_1 - b_2\|_1 \end{aligned}$$

which completes the proof. □

The dichotomy in representation of value functions V^{Γ}_{LB} and $V^{\Upsilon}_{\mathrm{UB}}$ allows for easy refinement of the bounds. By adding new elements to the set Γ, the value $V^{\Gamma}_{\mathrm{LB}}(b) = \max_{\alpha \in \Gamma} \alpha(b)$ can only increase—and hence the lower bound V^{Γ}_{LB} gets tighter. Similarly, by adding new elements to the set of points Υ, the solution of linear program (3.44) can only decrease and hence the upper bound $V^{\Upsilon}_{\mathrm{UB}}$ tightens.

Initial Bounds

We now describe our approach to form initial bounds V^{Γ}_{LB} and V^{Υ}_{UB} on the optimal value function V^* of the game:

Lower bound V^{Γ}_{LB} We initially set the lower bound to the value $\text{val}^{\sigma_1^{\text{unif}}}$ of the uniform strategy $\sigma_1^{\text{unif}} \in \Sigma_1$ of player 1 (i.e., one that plays each action $a_1 \in A_1$ with probability $1/|A_1|$ in each stage of the game). Recall that the value $\text{val}^{\sigma_1^{\text{unif}}}$ of the strategy σ_1^{unif} is a linear function (see Lemma 3.2), and hence the initial lower bound V^{Γ}_{LB} is a piecewise linear and convex function represented as a pointwise maximum of the set $\Gamma = \{\text{val}^{\sigma_1^{\text{unif}}}\}$.

Upper bound V^{Υ}_{UB} We use the solution of a perfect information variant of the game (i.e., where player 1 is assumed to know the entire history of the game, unlike in the original game). We form a modified game G' which is identical to the OS-POSG G (i.e., has the same states S, actions A_1 and A_2, dynamics T and rewards R), except that all information is revealed to player 1 in each step. G' is a perfect information stochastic game (Section 2.1.3) and we can apply the value iteration algorithm to solve G'. The additional information player 1 in G' (compared to G) can only increase the utility he can achieve. Hence V^*_s of the state s of game G' forms an upper bound on the utility player 1 can achieve in G if he knew that the initial state of the game is s (i.e., his belief is b_s where $b_s(s) = 1$). We set Υ to contain one point for each state $s \in S$ of the game, i.e., vertex of the $\Delta(S)$ simplex,

$$\Upsilon = \{(b_s, V^*_s) \mid s \in S\} \qquad b_s(s') = \begin{cases} 1 & s = s' \\ 0 & \text{otherwise .} \end{cases} \tag{3.48}$$

3.6.2 Point-based Updates

Unlike the exact value iteration algorithm (Section 3.5) which constructs all α-vectors needed to represent HV in each iteration, the HSVI algorithm focuses on a single belief at a time. Performing a *point-based* update in belief $b \in \Delta(S)$ corresponds to solving the stage-games $[HV^{\Gamma}_{\text{LB}}](b)$ and $[HV^{\Upsilon}_{\text{UB}}](b)$ where the values of subsequent stages are represented using value functions V^{Γ}_{LB} and V^{Υ}_{UB}, respectively.

Update of lower bound V^{Γ}_{LB} First, the linear program (3.37) is used to compute the optimal value composition $\mathsf{valcomp}(\pi_1^{\text{LB}}, \overline{\alpha}^{\text{LB}})$ in $[HV^{\Gamma}_{\text{LB}}](b)$, i.e.,

$$(\pi_1^{\text{LB}}, \overline{\alpha}^{\text{LB}}) \quad = \quad \operatorname*{arg\,max}_{\substack{\pi_1 \in \Pi_1 \\ \overline{\alpha} \in \mathsf{Conv}(\Gamma)^{A_1 \times O}}} \mathsf{valcomp}(\pi_1, \overline{\alpha})(b) \ . \tag{3.49}$$

The $\mathsf{valcomp}(\pi_1^{\text{LB}}, \overline{\alpha}^{\text{LB}})$ function is a linear function corresponding to a new α-vector that forms a lower bound on V^*. This new α-vector is used to refine the bound by setting $\Gamma \coloneqq \Gamma \cup \{\mathsf{valcomp}(\pi_1^{\text{LB}}, \overline{\alpha}^{\text{LB}})\}$. The application of point-based updates to refine lower bound V^{Γ}_{LB} preserves desirable properties of V^{Γ}_{LB} as the following lemma shows.

Lemma 3.22. *Application of point-based updates preserve the following properties:*
(1) V^{Γ}_{LB} *is* δ*-Lipschitz continuous.*
(2) V^{Γ}_{LB} *is lower bound on* V^**.*

Proof. Initially, value function V^{Γ}_{LB} satisfies both conditions. The set Γ contains only the value $\mathrm{val}^{\sigma_1^{\mathrm{unif}}}$ of the uniform strategy σ_1^{unif}, i.e., $V^{\Gamma}_{\mathrm{LB}}(b) = \mathrm{val}^{\sigma_1^{\mathrm{unif}}}(b)$ for every belief $b \in \Delta(S)$. Value $\mathrm{val}^{\sigma_1^{\mathrm{unif}}}$ is the value for a valid strategy σ_1^{unif} of player 1—hence it is δ-Lipschitz continuous (Lemma 3.5) and lower bounds V^*.

Assume that every α-vector in the set Γ is δ-Lipschitz continuous, and that for each $\alpha \in \Gamma$ there exists strategy $\sigma_1 \in \Sigma_1$ with $\mathrm{val}^{\sigma_1} \geq \alpha$ (which holds also for the initial V^{Γ}_{LB}). Let $\mathsf{valcomp}(\pi_1^{\mathrm{LB}}, \overline{\alpha}^{\mathrm{LB}})$ be the value composition from Equation (3.49) obtained when performing the point-based update of V^{Γ}_{LB} by solving $[HV^{\Gamma}_{\mathrm{LB}}](b)$. We will now show that the refined function $V^{\Gamma'}_{\mathrm{LB}}$ represented by the set $\Gamma' = \Gamma \cup \{\mathsf{valcomp}(\pi_1^{\mathrm{LB}}, \overline{\alpha}^{\mathrm{LB}})\}$ satisfies both properties, and hence any sequence of application of the point-based updates of V^{Γ}_{LB} preserves aforementioned properties.

(1) According to Lemma 3.13, $\mathsf{valcomp}(\pi_1^{\mathrm{LB}}, \overline{\alpha}^{\mathrm{LB}})$ is δ-Lipschitz continuous (and thus so is the value function $V^{\Gamma'}_{\mathrm{LB}}$ represented by the set $\Gamma' = \Gamma \cup \{\mathsf{valcomp}(\pi_1, \overline{\alpha})\}$).
(2) All α-vectors in Γ form lower bound on the value of some strategy of player 1. Since $\overline{\alpha}^{\mathrm{LB}} \in \Gamma^{A_1 \times O}$, we have that every $\alpha_{a_1,o}$ lower bounds the value of some strategy of player 1. The fact that $\mathsf{valcomp}(\pi_1^{\mathrm{LB}}, \overline{\alpha}^{\mathrm{LB}})$ is also a lower bound follows from Lemma 3.12—and hence every α-vector from the set $\Gamma' = \Gamma \cup \{\mathsf{valcomp}(\pi_1^{\mathrm{LB}}, \overline{\alpha}^{\mathrm{LB}})\}$ is a lower bound on V^*. Hence also $V^{\Gamma'}_{\mathrm{LB}}(b) = \sup_{\alpha \in \Gamma'} \alpha(b) \leq V^*(b)$.

□

Update of upper bound $V^{\Upsilon}_{\mathrm{UB}}$ Similarly to the case of the point-based update of the lower bound V^{Γ}_{LB}, the update of upper bound is performed by solving the stage game $[HV^{\Upsilon}_{\mathrm{UB}}](b)$. Since $V^{\Upsilon}_{\mathrm{UB}}$ is represented by a set of points Υ, it is not necessary to compute the optimal value composition. Instead, we form a refined upper bound $V^{\Upsilon'}_{\mathrm{UB}}$ (which corresponds to $V^{\Upsilon}_{\mathrm{UB}}$ after the point-based update is made) by adding a new point $(b, [HV^{\Upsilon}_{\mathrm{UB}}](b))$ to the set Υ' representing $V^{\Upsilon'}_{\mathrm{UB}}$, i.e., $\Upsilon' = \Upsilon \cup \{(b, [HV^{\Upsilon}_{\mathrm{UB}}](b))\}$. We now show that the upper bound $V^{\Upsilon}_{\mathrm{UB}}$ has the desired properties, and these properties are retained when applying the point based update—and hence we can perform point-based updates of $V^{\Upsilon}_{\mathrm{UB}}$ repeatedly.

Lemma 3.23. *Application of point-based updates preserve the following properties:*
(1) $V^{\Upsilon}_{\mathrm{UB}}$ *is* δ*-Lipschitz continuous.*
(2) $V^{\Upsilon}_{\mathrm{UB}}$ *is an upper bound on* V^**.*

Proof. $V^{\Upsilon}_{\mathrm{UB}}$ has been defined as a lower δ-Lipschitz envelope of $V^{\Upsilon}_{\mathrm{HSVI1}}$, hence it is δ-Lipschitz continuous (Lemma 3.21). We will therefore focus only on the property (2). Since the upper bound is initialized by a solution of a perfect information variant of the

game, we have that $y_i \geq V^*(b_i)$ for every (b_i, y_i) from the initial set Υ (Equation (3.48)). Hence applying Lemma 3.21, $V^{\Upsilon}_{\mathrm{UB}}$ is an upper bound on V^*.

We will now show that if $y_i \geq V^*(b_i)$ holds for $(b_i, y_i) \in \Upsilon$ (and $V^{\Upsilon}_{\mathrm{UB}}$ is thus an upper bound on V^*), the application of a point-based update in any belief yields set Υ' such that $y_i \geq V^*(b_i)$ also holds for every $(b_i, y_i) \in \Upsilon'$—and the resulting value function $V^{\Upsilon'}_{\mathrm{UB}}$ is therefore upper bound on V^* as well. Since $V^{\Upsilon}_{\mathrm{UB}} \geq V^*$, the utility function of any stage game satisfies $u^{V^{\Upsilon}_{\mathrm{UB}},b}(\pi_1, \pi_2) \geq u^{V^*,b}(\pi_1, \pi_2)$ for every $b \in \Delta(S)$, $\pi_1 \in \Pi_1$ and $\pi_2 \in \Pi_2$. This implies that $[HV^{\Upsilon}_{\mathrm{UB}}](b) \geq [HV^*](b) = V^*(b)$. We already know that $y_i \geq V^*(b_i)$ holds for $(b_i, y_i) \in \Upsilon$, and now we have $[HV^{\Upsilon}_{\mathrm{UB}}](b) \geq V^*(b)$. Therefore, for every $(b_i, y_i) \in \Upsilon \cup \{(b, [HV^{\Upsilon}_{\mathrm{UB}}](b))\}$, we have $y_i \geq V^*(b_i)$, and applying the Lemma 3.21, we have that the value function $V^{\Upsilon'}_{\mathrm{UB}}$ is an upper bound on V^*. □

The linear programs (3.37) and (3.38) solve the stage game $[HV](b)$ when the value function V is represented as a maximum over a set of linear functions (i.e., the way lower bound V^{Γ}_{LB} is). It is, however, possible to adapt constraints in (3.38) to solve the $[HV^{\Upsilon}_{\mathrm{UB}}](b)$ problem. We replace constraint (3.38c) by constraints inspired by the linear program (3.44) used to solve $V^{\Upsilon}_{\mathrm{UB}}(b)$.

$$\hat{V}(a_1, o) = \sum_{i=1}^{|\Upsilon|} \lambda_i^{a_1,o} y_i + \delta \sum_{s' \in S} \Delta^{s'}_{a_1,o} \qquad \forall (a_1, o) \in A_1 \times O \qquad (3.50a)$$

$$\sum_{i=1}^{|\Upsilon|} \lambda^i_{a_1,o} b_i(s') = b'_{a_1,o}(s') \qquad \forall (a_1, o, s') \in A_1 \times O \times S \qquad (3.50b)$$

$$\Delta^{s'}_{a_1,o} \geq b'_{a_1,o}(s') - \hat{\tau}(b, a_1, \pi_2, o)(s') \qquad \forall (a_1, o, s') \in A_1 \times O \times S \qquad (3.50c)$$

$$\Delta^{s'}_{a_1,o} \geq \hat{\tau}(b, a_1, \pi_2, o)(s') - b'_{a_1,o}(s') \qquad \forall (a_1, o, s') \in A_1 \times O \times S \qquad (3.50d)$$

$$\sum_{i=1}^{|\Upsilon|} \lambda_i^{a_1,o} = \sum_{s' \in S} \hat{\tau}(b, a_1, \pi_2, o)(s') \qquad \forall (a_1, o) \in A_1 \times O \qquad (3.50e)$$

$$\lambda^i_{a_1,o} \geq 0 \qquad \forall (a_1, o) \in A_1 \times O, 1 \leq i \leq |\Upsilon| \qquad (3.50f)$$

3.6.3 The Algorithm

We are now ready to present the heuristic search value iteration (HSVI) algorithm for one-sided POSGs (Algorithm 3.1) and prove its correctness. The algorithm is similar to the HSVI algorithm for POMDPs [Smith and Simmons, 2004, 2005]. First, the bounds V^{Γ}_{LB} and $V^{\Upsilon}_{\mathrm{UB}}$ on the optimal value function V^* are initialized (as described in Section 3.6.1) on line 1. Then, until the desired precision $V^{\Upsilon}_{\mathrm{UB}}(b^{\mathrm{init}}) - V^{\Gamma}_{\mathrm{LB}}(b^{\mathrm{init}}) \leq \varepsilon$ is reached, the algorithm performs a sequence of trials using the `Explore` procedure, starting from the initial belief b^{init} (lines 2–3).

The recursive procedure `Explore` generates a sequence of beliefs $\{b_i\}_{i=0}^{k}$ (for some $k \geq 0$) where $b_0 = b^{\mathrm{init}}$ and each belief b_t reached at the recursion depth t satisfied $\mathrm{excess}_t(b_t) > 0$ on line 2 or 10. The algorithm tries to ensure that values of beliefs b_t reached at t-th level of recursion (i.e., t-th stage of the game) are approximated with

Algorithm 3.1: HSVI algorithm for one-sided POSGs

Data: Game G, initial belief b^{init}, discount factor $\gamma \in (0,1)$, desired precision $\varepsilon > 0$, neighborhood parameter D

Result: Approximate value functions V^{Γ}_{LB} and V^{Υ}_{UB} satisfying $V^{\Upsilon}_{\text{UB}}(b) - V^{\Gamma}_{\text{LB}}(b) \leq \varepsilon$, sets Γ and Υ constructed by point-based updates that represent V^{Γ}_{LB} and V^{Υ}_{UB}

1 Initialize V^{Γ}_{LB} and V^{Υ}_{UB} (see Section 3.6.1)
2 **while** $\text{excess}_0(b^{\text{init}}) > 0$ **do**
3 `Explore`$(b^{\text{init}}, 0)$
4 **return** V^{Γ}_{LB} and V^{Υ}_{UB}, sets Γ and Υ that represent V^{Γ}_{LB} and V^{Υ}_{UB}
5 **procedure** `Explore`(b_t, t)
6 $(\pi_1^{\text{LB}}, \pi_2^{\text{LB}}) \leftarrow$ equilibrium strategy profile in $[HV^{\Gamma}_{\text{LB}}](b_t)$
7 $(\pi_1^{\text{UB}}, \pi_2^{\text{UB}}) \leftarrow$ equilibrium strategy profile in $[HV^{\Upsilon}_{\text{UB}}](b_t)$
8 Perform point-based updates of V^{Γ}_{LB} and V^{Υ}_{UB} at belief b_t (see Section 3.6.2)
9 $(a_1^*, o^*) \leftarrow$ select according to forward exploration heuristic
10 **if** $\mathbb{P}_{b,\pi_1^{\text{UB}},\pi_2^{\text{LB}}}[a_1^*, o^*] \cdot \text{excess}_{t+1}(\tau(b_t, a_1^*, \pi_2^{\text{LB}}, o^*)) > 0$ **then**
11 `Explore`$(\tau(b_t, a_1^*, \pi_2^{\text{LB}}, o^*), t+1)$
12 Perform point-based updates of V^{Γ}_{LB} and V^{Υ}_{UB} at belief b_t (see Section 3.6.2)

sufficient accuracy and the gap between $V^{\Upsilon}_{\text{UB}}(b)$ and $V^{\Gamma}_{\text{LB}}(b)$ is at most $\rho(t)$, where $\rho(t)$ is defined by

$$\rho(0) = \varepsilon \qquad \rho(t+1) = [\rho(t) - 2\delta D]/\gamma \ . \tag{3.51}$$

We require that ρ is a monotonically increasing and unbounded sequence, which holds for an arbitrary value of parameter D satisfying $0 < D < (1-\gamma)\varepsilon/2\delta$. When the approximation quality $V^{\Upsilon}_{\text{UB}}(b_t) - V^{\Gamma}_{\text{LB}}(b_t)$ of the value of a belief b_t reached at the t-th recursion level of `Explore` (i.e., at the $(t+1)$-th stage of the game) exceeds the desired approximation quality $\rho(t)$, it is said to have a positive *excess gap* $\text{excess}_t(b_t)$,

$$\text{excess}_t(b_t) = V^{\Upsilon}_{\text{UB}}(b_t) - V^{\Gamma}_{\text{LB}}(b_t) - \rho(t) \ . \tag{3.52}$$

Note that our definition of excess gap is more strict compared to the original HSVI algorithm for POMDPs, where the $-2\delta D$ term from Equation (3.51) is absent (see Equation (2.16)). Unlike in POMDPs which are single-agent, the belief transitions $\tau(b, a_1, \pi_2, o)$ in one-sided POSGs depend also on player 2 (and her strategy π_2). The tighter bounds on the approximation quality allows us to prove the correctness of the proposed algorithm in Theorem 3.25.

Forward exploration heuristic The algorithm uses a heuristic approach to select which belief $\tau(b, a_1, \pi_2^{\text{LB}}, o)$ will be considered in the next recursion level of the `Explore` procedure, i.e., what action observation pair $(a_1, o) \in A_1 \times O$ will be chosen by player 1, on line 9. This selection is motivated by Lemma 3.17—in order to ensure that $\text{excess}_t(b_t) \leq 0$ (or more precisely $\text{excess}_t(b_t) \leq -2\delta D$) at the currently considered belief b_t in t-th

recursion level, all beliefs $\tau(b_t, a_1, \pi_2^{\mathrm{LB}}, o)$ reached with positive probability when playing π_1^{UB} have to satisfy $\mathrm{excess}_{t+1}(\tau(b_t, a_1, \pi_2^{\mathrm{LB}}, o)) \leq 0$. Specifically, we focus on a belief that has the highest *weighted excess gap*. Inspired by the original HSVI algorithm for POMDPs [Smith and Simmons, 2004, 2005]), we define the weighted excess gap as the excess gap $\mathrm{excess}_{t+1}(\tau(b_t, a_1, \pi_2^{\mathrm{LB}}, o))$ multiplied by the probability that the action-observation pair (a_1, o) that leads to the belief $\tau(b_t, a_1, \pi_2^{\mathrm{LB}}, o)$ occurs. Hence, we select the action-observation pair (a_1^*, o^*) for further exploration, where

$$(a_1^*, o^*) = \operatorname*{arg\,max}_{(a_1,o)\in A_1\times O} \mathbb{P}_{b,\pi_1^{\mathrm{UB}},\pi_2^{\mathrm{LB}}}[a_1, o] \cdot \mathrm{excess}_{t+1}(\tau(b_t, a_1, \pi_2^{\mathrm{LB}}, o)) \ . \tag{3.53}$$

We now show formally that if the weighted excess gap of the optimal (a_1^*, o^*) satisfies $\mathbb{P}_{b,\pi_1^{\mathrm{UB}},\pi_2^{\mathrm{LB}}}[a_1^*, o^*] \cdot \mathrm{excess}_{t+1}(\tau(b_t, a_1^*, \pi_2^{\mathrm{LB}}, o^*)) \leq 0$, performing the point based update at b_t ensures that $\mathrm{excess}_t(b_t) \leq -2\delta D$.

Lemma 3.24. *Let b_t be a belief reached at t-th recursion level of* `Explore` *procedure such that the action observation pair (a_1^*, o^*) selected at line 9 of Algorithm 3.1 satisfies*

$$\mathbb{P}_{b,\pi_1^{\mathrm{UB}},\pi_2^{\mathrm{LB}}}[a_1^*, o^*] \cdot \mathrm{excess}_{t+1}(\tau(b_t, a_1^*, \pi_2^{\mathrm{LB}}, o^*)) \leq 0 \ . \tag{3.54}$$

Then $\mathrm{excess}_t(b_t) \leq -2\delta D$ after performing a point based update at b_t. Furthermore, all beliefs $b_t' \in \Delta(S)$ such that $\|b_t - b_t'\|_1 \leq D$ satisfy $\mathrm{excess}_t(b_t') \leq 0$.

Proof. Since $V_{\mathrm{LB}}^{\Gamma} \leq V^* \leq V_{\mathrm{UB}}^{\Upsilon}$, it holds that $[HV_{\mathrm{LB}}^{\Gamma}](b_t) \leq [HV_{\mathrm{UB}}^{\Upsilon}](b_t)$. Applying Lemma 3.17 with $C = \rho(t+1)$ yields that in case beliefs $\tau(b_t, a_1, \pi_2^{\mathrm{LB}}, o)$ satisfy $V_{\mathrm{UB}}^{\Upsilon}(\tau(b_t, a_1, \pi_2^{\mathrm{LB}}, o)) - V_{\mathrm{LB}}^{\Gamma}(\tau(b_t, a_1, \pi_2^{\mathrm{LB}}, o)) \leq \rho(t+1)$ then $[HV_{\mathrm{UB}}^{\Upsilon}](b_t) - [HV_{\mathrm{LB}}^{\Gamma}](b_t) \leq \gamma\rho(t+1)$. This has to hold in the considered situation—otherwise there will be $(a_1, o) \in A_1 \times O$ with $V_{\mathrm{UB}}^{\Upsilon}(\tau(b_t, a_1, \pi_2^{\mathrm{LB}}, o)) - V_{\mathrm{LB}}^{\Gamma}(\tau(b_t, a_1, \pi_2^{\mathrm{LB}}, o)) > \rho(t+1)$ (i.e., $\mathrm{excess}_{t+1}(\tau(b_t, a_1, \pi_2^{\mathrm{LB}}, o)) > 0$) such that $\mathbb{P}_{b,\pi_1^{\mathrm{UB}},\pi_2^{\mathrm{LB}}}[a_1, o] > 0$, which would have contradicted $\mathbb{P}_{b,\pi_1^{\mathrm{UB}},\pi_2^{\mathrm{LB}}}[a_1^*, o^*] \cdot \mathrm{excess}_{t+1}(\tau(b_t, a_1^*, \pi_2^{\mathrm{LB}}, o^*)) \leq 0$.

Now, according to Equation (3.51), $[HV_{\mathrm{UB}}^{\Upsilon}](b_t) - [HV_{\mathrm{LB}}^{\Gamma}](b_t) \leq \gamma\rho(t+1) = \rho(t) - 2\delta D$. Hence the excess gap after performing the point-based update in b_t satisfies

$$\mathrm{excess}_t(b_t) = V_{\mathrm{UB}}^{\Upsilon}(b_t) - V_{\mathrm{LB}}^{\Gamma}(b_t) - \rho(t) \leq \gamma\rho(t+1) - \rho(t) = [\rho(t) - \rho(t)] - 2\delta D = -2\delta D \tag{3.55}$$

which completes the proof of the first part of the lemma.

Now since the value functions V_{LB}^{Γ} and $V_{\mathrm{UB}}^{\Upsilon}$ are δ-Lipschitz continuous (Lemma 3.22 and Lemma 3.23), the difference $V_{\mathrm{UB}}^{\Upsilon} - V_{\mathrm{LB}}^{\Gamma}$ is 2δ-Lipschitz continuous. Thus for every belief $b_t' \in \Delta(S)$ satisfying $\|b_t - b_t'\|_1 \leq D$ we have

$$V_{\mathrm{UB}}^{\Upsilon}(b_t') - V_{\mathrm{LB}}^{\Gamma}(b_t') \leq V_{\mathrm{UB}}^{\Upsilon}(b_t) - V_{\mathrm{LB}}^{\Gamma}(b_t) + 2\delta\|b_t - b_t'\|_1 \leq V_{\mathrm{UB}}^{\Upsilon}(b_t) - V_{\mathrm{LB}}^{\Gamma}(b_t) + 2\delta D \ . \tag{3.56}$$

Now since $\mathrm{excess}_t(b_t) \leq -2\delta D$, we have $\mathrm{excess}_t(b_t') \leq 0$ which proves the second part of the lemma. □

We now use Lemma 3.24 (especially its second part) to prove that the Algorithm 3.1 terminates with $V^{\Upsilon}_{\mathrm{UB}}(b^{\mathrm{init}}) - V^{\Gamma}_{\mathrm{LB}}(b^{\mathrm{init}}) \leq \varepsilon$. Furthermore, we later show that the value functions V^{Γ}_{LB} and $V^{\Upsilon}_{\mathrm{UB}}$ can be used to obtain ε-Nash equilibrium strategies for the game in Section 3.7.

Theorem 3.25. *The Algorithm 3.1 terminates with $V^{\Upsilon}_{\mathrm{UB}}(b^{\mathrm{init}}) - V^{\Gamma}_{\mathrm{LB}}(b^{\mathrm{init}}) \leq \varepsilon$ for arbitrary $\varepsilon > 0$ and $0 < D < (1-\gamma)\varepsilon/2\delta$.*

Proof. By the choice of parameter D, the sequence $\rho(t)$ (for $\rho(0) = \varepsilon$) is monotonically increasing and unbounded, and the difference between value functions V^{Γ}_{LB} and $V^{\Upsilon}_{\mathrm{UB}}$ is bounded by $U - L$ (since $L \leq V^{\Gamma}_{\mathrm{LB}}(b) \leq V^{\Upsilon}_{\mathrm{UB}}(b) \leq U$ for every belief $b \in \Delta(S)$). Therefore, there exists $T_{\max}$ such that $\rho(T_{\max}) \geq U - L \geq V^{\Upsilon}_{\mathrm{UB}}(b) - V^{\Gamma}_{\mathrm{LB}}(b)$ for every $b \in \Delta(S)$, and the recursive procedure `Explore` thus always terminates.

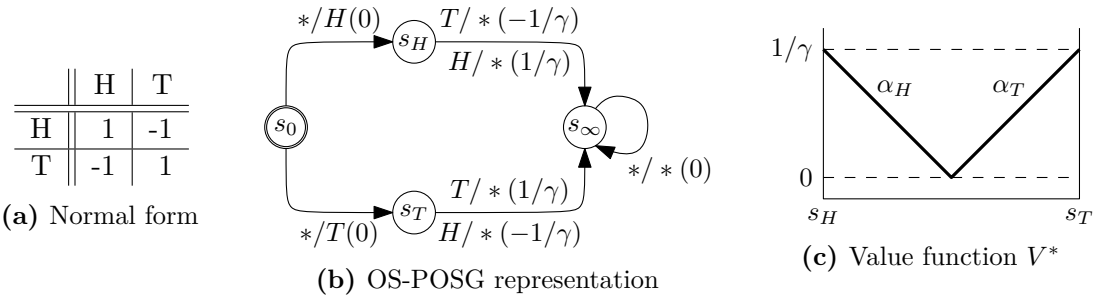
We prove that the Algorithm 3.1 terminates by reasoning about sets $\Psi_t \subset \Delta(S)$ of belief points where the trials performed by the `Explore` terminated. Initially, $\Psi_t = \varnothing$ for every $0 \leq t < T_{\max}$. Whenever the `Explore` recursion terminates at recursion level t (i.e., the condition on line 10 does not hold), the belief b_t (which was the last belief considered during the trial) is added into set Ψ_t ($\Psi_t \coloneqq \Psi_t \cup \{b_t\}$). Recall that since $\Delta(S)$ is compact, it is, in particular, totally bounded, i.e., if a subset $\Psi_t \subset \Delta(S)$ satisfies $\forall b, b' : \|b - b'\|_1 > D$ then Ψ_t must be finite. As the number of possible termination depths is finite ($0 \leq t \leq T_{\max}$), the algorithm has to terminate unless some Ψ_t is infinite. We show that this, however, cannot happen as every $b, b' \in \Psi_t$ has to satisfy $\|b - b'\| > D$.

Assume to the contradiction that two trials terminated at recursion level t with the last beliefs considered $b_t^{(1)}$ (for the earlier trial) and $b_t^{(2)}$ (for the trial that occurred at a later time), and $\|b_t^{(1)} - b_t^{(2)}\|_1 \leq D$ holds. When the former trial has been terminated in belief $b_t^{(1)}$, all reachable beliefs from $b_t^{(1)}$ had negative excess gap (otherwise the trial would have continued as the condition on line 10 would have been satisfied). According to Lemma 3.24, after the point-based update is performed in $b_t^{(1)}$, the excess gap of all beliefs b'_t with $\|b_t^{(1)} - b'_t\|_1 \leq D$ have negative excess gap $\mathrm{excess}_t(b'_t) \leq 0$. When $b_t^{(2)}$ has been selected for exploration in $(t-1)$-th level of recursion, the condition on line (10) was met and $b_t^{(2)}$ must have had positive excess gap $\mathrm{excess}_t(b_t^{(2)}) > 0$. This, however, contradicts that all beliefs b'_t with $\|b_t^{(1)} - b'_t\|_1 \leq D$ (i.e., including $b_t^{(2)}$) already have negative excess gap.

Note that at least one trial must have terminated in the first level of recursion (unless the Algorithm 3.1 has terminated on line 2 with $\mathrm{excess}_0(b^{\mathrm{init}}) \leq 0$ beforehand). By Lemma 3.24, the update in b^{init} then renders $\mathrm{excess}_0(b^{\mathrm{init}}) \leq -2\delta D \leq 0$. We then have that $V^{\Upsilon}_{\mathrm{UB}}(b^{\mathrm{init}}) - V^{\Gamma}_{\mathrm{LB}}(b^{\mathrm{init}}) \leq \rho(0) = \varepsilon$ which completes the proof. □

3.7 Using Value Function to Play

In the previous section, we have presented an algorithm that is able to approximate the value $V^*(b^{\mathrm{init}})$ of the game within an arbitrary given precision $\varepsilon > 0$ starting from an



(a) Normal form

(b) OS-POSG representation

(c) Value function V^*

Figure 3.1: Example of a game where the belief is not a sufficient statistic for the imperfectly informed player 1.

arbitrary initial belief b^{init}. However, in many cases, knowing the value of the game only cannot be considered as a solution of the game since the strategies that achieve this near-optimal performance are required. In this section, we show that using the value functions V^{Γ}_{LB} and V^{Υ}_{UB} computed by the HSVI algorithm (Algorithm 3.1) is sufficient to devise ε-Nash equilibrium strategies for both players.

The existence of the Bellman's equation for one-sided POSGs (see Theorem 3.15) may suggest that the near-optimal strategies can be extracted by employing the lookahead decision rule (similarly to POMDPs) and obtaining strategies to play in the current stage by computing the Nash equilibrium of stage games $[HV^{\Gamma}_{\text{LB}}](b)$ and $[HV^{\Upsilon}_{\text{UB}}](b)$, respectively. However, this is *not* possible in case of one-sided POSGs—and unlike in the case of POMDPs and MaGIIs, belief of player 1 does not constitute a sufficient statistic for playing the game. The reasons for this are similar to the usage of unsafe resolving [Burch, 2018; Seitz et al., 2019] in the realm of extensive-form games. We use the following example to demonstrate the insufficiency of the belief to play the game.

Example 3.1. Consider a *matching pennies* game shown in Figure 3.1a. This game can be formalized as a one-sided POSG that is shown in Figure 3.1b. The game starts in the state s_0 (i.e., the initial belief is $b^{\text{init}}(s_0) = 1$) and the player 2 chooses her action H or T. Next, after transitioning to s_H or s_T (based on the decision of player 2), player 1 is *unaware* of the true state of the game (i.e., the past decision of player 1) and chooses his action H or T. Based on the combination of decisions taken by the players, player 1 gets either $1/\gamma$ or $-1/\gamma$ and the game transitions to the state s_∞ where it stays forever with zero future rewards.

To understand the caveats of using belief $b \in \Delta(S)$ to derive the stage strategy to play, let us consider the optimal value function V^* of the OS-POSG representation (Figure 3.1b) of the matching pennies game. Figure 3.1c shows the values of V^* over simplex $\Delta(\{s_H, s_T\})$. If it is more likely that the player 2 played H in the first stage of the game (i.e., the current state is s_H), it is optimal for player 1 to play strategy prescribing him to play H in the current stage (with value α_H). Conversely, if it is more likely that the current state is s_T, the player 1 is better of with playing T (with value α_T). The value function V^* is then a point-wise maximum over these two linear functions.

Now, since the uniform mixture between H and T is the Nash equilibrium strategy for both players in the matching pennies game, player 1 will find himself in a situation when he assumes that the current belief is $\{s_H : 0.5, s_T : 0.5\}$. In this belief, any decision of player 1 yields expected reward 0—hence based purely on the belief, the player 1 may opt to play, e.g., "always T". However, such strategy is not in equilibrium and player 2 is able to exploit it by playing "always H". This example illustrates that the belief alone does not provide sufficient information to choose the right strategy π_1 for the current stage based on the Equation (3.23b).

3.7.1 Justified Value Functions

First of all, we define conditions under which it makes sense to use value function to play a one-sided POSG. The conditions are similar to *uniform improvability* in, e.g., POMDPs. Our definitions, however, reflect the fact that we deal with a two-player problem (and we thus introduce the condition for each player separately). Moreover, we use a stricter condition for the player 1 who does *not* have a perfect information about the belief—and thus defining the condition based solely on the beliefs is not sufficient.

Definition 3.11 (Min-justified value function)**.** Convex continuous value function V is said to be *min-justified* (or, justified from the perspective of the minimizing player 2) if for every belief $b \in \Delta(S)$ it holds that $[HV](b) \leq V(b)$.

Definition 3.12 (Max-justified value function)**.** Let Γ be a compact set of linear functions, and V be a value function such that $V(b) = \sup_{\alpha \in \Gamma} \alpha(b)$. V is said to be *max-justified* by a set of α-vectors Γ (or, justified from the perspective of the maximizing player 1) if for every $\alpha \in \Gamma$ there exists $\pi_1 \in \Pi_1$ and $\overline{\alpha} \in \Gamma^{A_1 \times O}$ such that $\mathsf{valcomp}(\pi_1, \overline{\alpha}) \geq \alpha$.

Note that if the value function V is min-justified, then there exists a strategy σ_2 of player 2 that *justifies* the value $V(b)$ for every belief $b \in \Delta(S)$, i.e., we have $\mathbb{E}_{b,\sigma_1,\sigma_2}[\mathsf{Disc}^\gamma] \leq V(b)$ against every strategy σ_1 of the player 1. Similarly, if the value function V is max-justified by Γ, for every belief $b \in \Delta(S)$, there exists a strategy σ_1 of player 1 such that $\mathrm{val}^{\sigma_1}(b) \geq V(b)$. In Sections 3.7.2 and 3.7.3, we use these technical definitions to show that such strategies not only exist, but we are able to construct them in an algorithmic way.

We will now support this intuition by showing that if a value function is min-justified or max-justified, respectively, it cannot attain better utility than any strategy of player 2 or player 1 can achieve.

Lemma 3.26. *Let V be a value function that is min-justified. Then $V(b) \geq L$.*

Proof. Assume for the contradiction that $V(b) < L$ for some belief $b \in \Delta(S)$. We pick $b = \arg\min_{b' \in \Delta(S)} V(b')$ and denote $\varepsilon = L - V(b)$. Now, using the utility $u^{V,b}$ from Definition 3.10 and using our choice of b, we have

$$\begin{aligned} u^{V,b}(\pi_1, \pi_2) &= \mathbb{E}_{b,\pi_1,\pi_2}[R(s, a_1, a_2)] + \gamma \sum_{a_1,o} \mathbb{P}_{b,\pi_1,\pi_2}[a_1, o] V(\tau(b, a_1, \pi_2, o)) \\ &\geq \underline{r} + \gamma \sum_{a_1,o} \mathbb{P}_{b,\pi_1,\pi_2}[a_1, o] V(b) = \underline{r} + \gamma V(b) = \underline{r} + \gamma(L - \varepsilon) \end{aligned}$$

where $\underline{r}$ is the minimum reward in the game. Since $L = \sum_{t=1}^{\infty} \gamma^{t-1} \underline{r} = \underline{r} + \sum_{t=2}^{\infty} \gamma^{t-1} \underline{r} = \underline{r} + \gamma L$, we also have that $u^{V,b}(\pi_1, \pi_2) \geq L - \gamma\varepsilon$. Therefore it would have to also hold that $[HV](b) = \max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2} u^{V,b}(\pi_1, \pi_2) \geq L - \gamma\varepsilon > L - \varepsilon = V(b)$ which contradicts that V is min-justified. □

Lemma 3.27. *Let V be a value function that is max-justified by a set of α-vectors Γ. Then for every $\alpha \in \Gamma$ we have $\alpha \leq U$.*

Proof. Let V be max-justified by Γ and let us assume for contradiction that there exists $\alpha \in \Gamma$ and $s \in S$ such that $\alpha(s) > U$. We pick α and s such that $(\alpha, s) = \arg\max_{\alpha \in \Gamma, s \in S} \alpha(s)$ and denote $\varepsilon = \alpha(s) - U$. Using Definition 3.8 and our choice of (α, s), we get the following for every $\pi_1 \in \Pi_1$ and $\overline{\alpha} \in \Gamma^{A_1 \times O}$:

$$\begin{aligned} \mathsf{valcomp}(\pi_1, \overline{\alpha})(s) &= \min_{a_2 \in A_2} \sum_{a_1 \in A_1} \pi_1(a_1) \Big[R(s, a_1, a_2) + \gamma \sum_{o, s' \in O \times S} T(o, s' \mid s, a_1, a_2) \alpha_{a_1,o}(s') \Big] \\ &\leq \min_{a_2 \in A_2} \sum_{a_1 \in A_1} \pi_1(a_1) \Big[\overline{r} + \gamma \sum_{o, s' \in O \times S} T(o, s' \mid s, a_1, a_2) \alpha(s) \Big] \\ &= \min_{a_2 \in A_2} [\overline{r} + \gamma \alpha(s)] \end{aligned}$$

where $\overline{r} = \max_{(s,a_1,a_2)} R(s, a_1, a_2)$ is the maximum reward in the game. Since $U = \sum_{t=1}^{\infty} \gamma^{t-1} \overline{r} = \overline{r} + \sum_{t=2}^{\infty} \gamma^{t-1} \overline{r} = \overline{r} + \gamma U$, we have

$$\mathsf{valcomp}(\pi_1, \overline{\alpha})(s) \leq \min_{a_2 \in A_2} [\overline{r} + \gamma \alpha(s)] = \overline{r} + \gamma(U + \varepsilon) = U + \gamma\varepsilon < U + \varepsilon = \alpha(s) \ .$$

Now, we have that for every value composition $\mathsf{valcomp}(\pi_1, \overline{\alpha})$ it holds $\mathsf{valcomp}(\pi_1, \overline{\alpha})(s) < \alpha(s)$. This contradicts Definition 3.12 as no value composition can satisfy $\mathsf{valcomp}(\pi_1, \overline{\alpha}) \geq \alpha$, and V cannot thus be max-justified by Γ. □

In the later proof showing that the value function V^{Γ}_{LB} resulting from the execution of the Algorithm 3.1 is max-justified by $\mathsf{Conv}(\Gamma)$, we will be using the following technical lemma.

Lemma 3.28. *Let Γ be a set of linear functions, and V a value function that is max-justified by Γ. Then V is also max-justified by $\mathsf{Conv}(\Gamma)$.*

Proof. Recall that V is max-justified by Ω if 1) $V(b) = \sup_{\alpha\in\Omega} \alpha(b)$ and 2) for every $\alpha \in \Omega$ there exists $\pi_1 \in \Pi_1$ and $\overline{\alpha} \in \Omega^{A_1\times O}$ such that $\mathsf{valcomp}(\pi_1, \overline{\alpha}) \geq \alpha$. We will now verify that these properties hold for $\mathsf{Conv}(\Gamma)$. First of all, 1) follows directly from Proposition 3.8 and we can replace Γ by $\mathsf{Conv}(\Gamma)$ without changing the values V attains. We will now prove 2) by showing that for every $\alpha \in \mathsf{Conv}(\Gamma)$, there exists $\pi_1 \in \Pi_1$ and $\overline{\alpha} \in \mathsf{Conv}(\Gamma)^{A_1\times O}$ such that $\mathsf{valcomp}(\pi_1, \overline{\alpha}) \geq \alpha$.

First of all, let us write $\alpha \in \mathsf{Conv}(\Gamma)$ as a finite convex combination $\sum_{i=1}^{k} \lambda_i \alpha^i$ of α-vectors $\alpha^i \in \Gamma$. Using the assumption that V is max-justified by Γ, we have that for every α^i there exists $\mathsf{valcomp}(\pi_1^i, \overline{\alpha}^i)$ such that $\overline{\alpha}^i \in \Gamma^{A_1\times O}$ and $\mathsf{valcomp}(\pi_1^i, \overline{\alpha}^i) \geq \alpha^i$. We claim that the desired π_1 and $\overline{\alpha}$ that witness that V is max-justified by $\mathsf{Conv}(\Gamma)$ satisfy $\pi_1(a_1) = \sum_{i=1}^{k} \lambda_i \pi_1^i(a_1)$ and $\alpha_{a_1,o} = \sum_{i=1}^{k} \lambda_i \pi_1^i(a_1)\alpha^i_{a_1,o}/\pi_1(a_1)$.[5] Clearly, $\sum_{i=1}^{k} \lambda_i \mathsf{valcomp}(\pi_1^i, \overline{\alpha}^i) \geq \sum_{i=1}^{k} \lambda_i \alpha^i = \alpha \in \mathsf{Conv}(\Gamma)$. To finish the proof, we show that $\mathsf{valcomp}(\pi_1, \overline{\alpha}) \geq \sum_{i=1}^{k} \lambda_i \mathsf{valcomp}(\pi_1^i, \overline{\alpha}^i)$. By Definition 3.8, we have

$$
\begin{aligned}
&\mathsf{valcomp}(\pi_1, \overline{\alpha}) \\
&= \min_{a_2\in A_2} \Big[\sum_{a_1\in A_1} \pi_1(a_1)R(s,a_1,a_2) + \gamma \sum_{(a_1,o,s')\in A_1\times O\times S} \pi_1(a_1)T(o,s' \mid s,a_1,a_2)\alpha_{a_1,o}(s') \Big] \\
&= \min_{a_2\in A_2} \Big[\sum_{i=1}^{k} \sum_{a_1\in A_1} \lambda_i \pi_1^i(a_1)R(s,a_1,a_2) + \gamma \sum_{(a_1,o,s')\in A_1\times O\times S} T(o,s' \mid s,a_1,a_2) \sum_{i=1}^{k} \lambda_i \pi_1^i(a_1)\alpha^i_{a_1,o}(s') \Big] \\
&= \min_{a_2\in A_2} \sum_{i=1}^{k} \lambda_i \Big[\sum_{a_1\in A_1} \pi_1^i(a_1)R(s,a_1,a_2) + \gamma \sum_{(a_1,o,s')\in A_1\times O\times S} \pi_1^i(a_1)T(o,s' \mid s,a_1,a_2)\alpha^i_{a_1,o}(s') \Big] \\
&\geq \sum_{i=1}^{k} \lambda_i \min_{a_2\in A_2} \Big[\sum_{a_1\in A_1} \pi_1^i(a_1)R(s,a_1,a_2) + \gamma \sum_{(a_1,o,s')\in A_1\times O\times S} \pi_1^i(a_1)T(o,s' \mid s,a_1,a_2)\alpha^i_{a_1,o}(s') \Big] \\
&= \sum_{i=1}^{k} \lambda_i \mathsf{valcomp}(\pi_1^i, \overline{\alpha}^i) \ .
\end{aligned}
$$

□

3.7.2 Strategy of Player 1

In this section, we will show that when the value function V is max-justified by a set of α-vectors Γ, then we can implicitly form a strategy σ_1 of player 1 that achieves utility of at least $V(b^{\text{init}})$ for any given initial belief b^{init}. The Algorithm 3.2 is inspired by the ideas of continual resolving for extensive-form games [Moravčík et al., 2017].

[5] Observe that $\alpha_{a_1,o} \in \mathsf{Conv}(\Gamma)$ since $\pi_1(a_1) = \sum_{i=1}^{k} \lambda_i \pi_1^i(a_1)$ and the coefficients $\lambda_i \pi_1^i(a_1)/\pi_1(a_1)$ thus sum to 1.

In the course of playing according to the Algorithm 3.2, the algorithm keeps a lower bound ρ on the values the reconstructed strategy has to achieve. In accordance with the terminology of continual resolving for extensive-form games, we term the linear function ρ a *gadget*. The goal of the $\texttt{Act}(b, \rho)$ method is to reconstruct a strategy σ_1 of player such that its value satisfies $\text{val}^{\sigma_1} \geq \rho$. We will now show that the `Act` method achieves exactly this. The reasoning about the current gadget allows us to obtain guarantees on the quality of the reconstructed strategy, even though the player 1 does not know the stage strategies used by the adversary, and thus cannot trust his current belief.

Algorithm 3.2: Continual resolving algorithm for one-sided POSGs

input : one-sided POSG G
a compact set Γ of linear functions representing convex value function V

1 $b \leftarrow b^{\text{init}}$
2 $\rho^{\text{init}} \leftarrow \arg\max_{\alpha \in \Gamma} \alpha(b_{\text{init}})$
3 $\texttt{Act}(b^{\text{init}}, \rho^{\text{init}})$
4 **procedure** $\texttt{Act}(b, \rho)$
5 $\quad (\pi_1^*, \overline{\alpha}^*) \leftarrow \arg\max_{\pi_1, \overline{\alpha}} \{\mathsf{valcomp}(\pi_1, \overline{\alpha})(b) \mid \pi_1 \in \Pi_1, \overline{\alpha} \in \Gamma^{A_1 \times O} \text{ s.t. } \mathsf{valcomp}(\pi_1, \overline{\alpha}) \geq \rho\}$
6 $\quad \pi_2 \leftarrow$ solve $[HV](b)$ to obtain assumed stage strategy of the adversary
7 $\quad$ sample and play $a_1 \sim \pi_1^*$
8 $\quad o \leftarrow$ observed observation
9 $\quad b' \leftarrow \tau(b, a_1, \pi_2, o)$
10 $\quad \texttt{Act}(b', \alpha^*_{a_1, o})$

Theorem 3.29. *Let V be a value function that is max-justified by a set of α-vectors Γ. Let $b^{\text{init}} \in \Delta(S)$ and $\rho^{\text{init}} \in \Gamma$ be arbitrary. By playing according to the $\texttt{Act}(b^{\text{init}}, \rho^{\text{init}})$, player 1 implicitly forms a strategy σ_1 such that $\text{val}^{\sigma_1} \geq \rho^{\text{init}}$.*

Proof. Let b^{init} and ρ^{init} be as in the theorem. To get to our result, we will first consider arbitrary belief $b \in \Delta(S)$ and gadget $\rho \in \Gamma$, and we will consider that player 1 follows $\texttt{Act}(b, \rho)$ for K stages of the game only, and then follows the uniform strategy σ_1^{unif} in the rest of the game. We denote such strategy of player 1 $\sigma_1^{b,\rho,K}$, and we prove that the value of such strategy satisfies $\text{val}^{\sigma_1^{b,\rho,K}} \geq \rho - \gamma^K \cdot (U - L)$. We prove this claim by induction.

First, assume that $K = 0$, i.e., player 1 plays the uniform strategy σ_1^{unif} immediately. Value of the uniform strategy σ_1^{unif} is at least $\text{val}^{\sigma_1^{\text{unif}}} \geq L$ (Proposition 3.1) while $\rho \leq U$ (Lemma 3.27). Hence $\text{val}^{\sigma_1^{b,\rho,0}} \geq L \geq L - (U - \rho) = \rho - \gamma^0(U - L)$.

Now let $K \geq 1$ and let us assume that $\text{val}^{\sigma_1^{b',\rho',K-1}} \geq \rho' - \gamma^{K-1}(U - L)$ for every $b' \in \Delta(S)$ and every gadget $\rho' \in \Gamma$. Observe that due to the recursive nature of the `Act` method, we can represent the strategy $\sigma_1^{b,\rho,K}$ as a composite strategy $\sigma_1^{b,\rho,K} = \mathsf{comp}(\pi_1^*, \overline{\zeta})$ where $\zeta_{a_1,o} = \sigma_1^{\tau(b,a_1,\pi_2,o),\alpha^*_{a_1,o},K-1}$ and π_1^* comes from line 5 of Algorithm 3.2. (To ensure that $\overline{\alpha}^*$ and π_1^* are correctly defined, the algorithm requires the existence of a value

composition satisfying $\mathsf{valcomp}(\pi_1, \overline{\alpha}) \geq \rho$. This requirement holds since V is max-justified by the set Γ and $\rho \in \Gamma$.) Applying Lemma 3.11, the induction hypothesis, and Definition 3.8 (in this order), we have

$$\begin{aligned}
\mathrm{val}^{\mathsf{comp}(\pi_1^*,\overline{\zeta})}(s) &= \min_{a_2 \in A_2} \sum_{a_1 \in A_1} \pi_1^*(a_1)\Big[R(s,a_1,a_2) + \gamma \sum_{(o,s') \in O \times S} T(o,s' \mid s,a_1,a_2)\mathrm{val}^{\zeta_{a_1,o}}(s')\Big] \\
&\geq \min_{a_2 \in A_2} \sum_{a_1 \in A_1} \pi_1^*(a_1)\Big[R(s,a_1,a_2) + \\
&\qquad + \gamma \sum_{(o,s') \in O \times S} T(o,s' \mid s,a_1,a_2)[\alpha^*_{a_1,o}(s') - \gamma^{K-1}(U-L)]\Big] \\
&= \min_{a_2 \in A_2} \sum_{a_1 \in A_1} \pi_1^*(a_1)\Big[R(s,a_1,a_2) + \\
&\qquad + \gamma \sum_{(o,s') \in O \times S} T(o,s' \mid s,a_1,a_2)\alpha^*_{a_1,o}(s')\Big] - \gamma^K(U-L) \\
&= \mathsf{valcomp}(\pi_1^*, \overline{\alpha}^*) - \gamma^K(U-L)\ .
\end{aligned}$$

Now, we have $\mathrm{val}^{\sigma_1^{b,\rho,K}} = \mathrm{val}^{\mathsf{comp}(\pi_1^*,\overline{\zeta})} \geq \mathsf{valcomp}(\pi_1^*, \overline{\alpha}^*) - \gamma^K(U-L)$. Furthermore, according to constraint on line 5 of Algorithm 3.2, we also have $\mathsf{valcomp}(\pi_1^*, \overline{\alpha}^*) \geq \rho$. Therefore, we also have $\mathrm{val}^{\sigma_1^{b,\rho,K}} \geq \rho - \gamma^K(U-L)$ which completes the induction step.

Denote by σ_1 the strategy where player 1 follows $\mathtt{Act}(b^{\mathrm{init}}, \rho^{\mathrm{init}})$ for *infinite* period of time (i.e., as $K \to \infty$). We then have

$$\mathrm{val}^{\sigma_1} = \lim_{K \to \infty} \mathrm{val}^{\sigma_1^{b^{\mathrm{init}},\rho^{\mathrm{init}},K}} \geq \lim_{K \to \infty} [\rho^{\mathrm{init}} - \gamma^K(U-L)] = \rho^{\mathrm{init}}$$

which completes the proof. □

Corollary 3.30. *Let $b^{\mathrm{init}} \in \Delta(S)$ be an arbitrary initial belief and let V be a value function max-justified by the compact set Γ of α-vectors. Then Algorithm 3.2 implicitly constructs a strategy σ_1 which guarantees utility of at least $V(b^{\mathrm{init}})$ to player 1.*

Proof. ρ^{init} from line 2 of Algorithm 3.2 has value $\rho^{\mathrm{init}}(b^{\mathrm{init}}) = V(b^{\mathrm{init}})$ in the initial belief b^{init}. According to Theorem 3.29, we have that we are able to construct a strategy σ_1 with value $\mathrm{val}^{\sigma_1} \geq \rho^{\mathrm{init}}$. Hence $\mathrm{val}^{\sigma_1}(b^{\mathrm{init}}) \geq \rho^{\mathrm{init}}(b^{\mathrm{init}}) = V(b^{\mathrm{init}})$. □

3.7.3 Strategy of Player 2

We will now present an analogous algorithm to obtain strategy for player 2 when the value function V is min-justified. Recall that the stage strategies π_2 of player 2 influence the belief of player 1 (Equation 3.6). Unlike player 1, player 2 knows which stage strategies π_2 have been used in the past, and he is thus able to infer the current belief of player 1.

To this end, the `Act` method of Algorithm 3.3 depends on the current belief of player 1 only (and not on the gadget ρ as it did when strategy of player 1 was considered).

Algorithm 3.3: Strategy of player 2

input : one-sided POSG G
convex value function V

```
1 Act(b^init)
2 procedure Act(b)
3     π2* ← optimal strategy of player 2 in the stage game [HV](b)
4     s ← currently observed state
5     sample and play a2 ~ π2*(· | s)
6     (a1, o) ← action of the adversary and the corresponding observation
7     Act(τ(b, a1, π2*, o))
```

We will now show that if the value function V is min-justified, then playing according to Algorithm 3.3 guarantees that the utility will be at most[6] $V(b^{\text{init}})$.

Theorem 3.31. *Let V be a min-justified value function and let b^{init} be the initial belief of the game. The Algorithm 3.3 implicitly constructs a strategy σ_2 which guarantees utility of at most $V(b^{\text{init}})$ to player 2.*

Proof. For the purposes of this proof, we will use $\text{val}_2(\sigma_2', b) = \sup_{\sigma_1 \in \Sigma_1} \mathbb{E}_{b,\sigma_1,\sigma_2'}[\mathsf{Disc}^\gamma]$ to denote the value of a strategy σ_2' of player 2 when the belief of player 1 is b. Similarly to the proof of Theorem 3.29, we will first consider strategies $\sigma_2^{b,K}$ where player 2 plays according to $\texttt{Act}(b)$ for K steps, and then follows an arbitrary (e.g., uniform) strategy in the rest of the game, and we show that $\text{val}_2(\sigma_2^{b,K}, b) \leq V(b) + \gamma^K(U - L)$.

First, let $K = 0$ and $b \in \Delta(S)$ be the belief of player 1. By Proposition 3.1, player 1 cannot achieve higher utility than U. Moreover, V is min-justified so we have $V(b) \geq L$ by Lemma 3.26. Therefore, player 1 cannot achieve higher utility than $\text{val}_2(\sigma_2^{b,0}, b) \leq U \leq U + V(b) - L = V(b) + \gamma^0(U - L)$ when his belief is b.

Now let $K \geq 1$ be arbitrary. By the induction hypothesis, we have that strategy $\sigma_2^{b',K-1}$ guarantees that the utility is at most $\text{val}_2(\sigma_2^{b',K-1}, b') \leq V(b') + \gamma^{K-1}(U - L)$ when the belief of player 1 is b'. Let us evaluate the utility that $\sigma_2^{b,K}$ guarantees against arbitrary strategy σ_1 of player 1 in belief b. In the first stage of the game, player 2 plays according to π_2^* obtained on line 3 of Algorithm 3.3, and the expected reward from the first stage is $\mathbb{E}_{b,\sigma_1,\pi_2^*}[R(s, a_1, a_2)]$. If player 1 plays a_1 and observes o, he reaches a subgame where the belief of player 1 is $\tau(b, a_1, \pi_2^*, o)$ and player 2 plays $\sigma_2^{\tau(b,a_1,\pi_2^*,o),K-1}$. Using the induction hypothesis, we know that player 1 is able to achieve utility of at most $\text{val}_2(\sigma_2^{\tau(b,a_1,\pi_2^*,o),K-1}, \tau(b, a_1, \pi_2^*, o)) \leq V(\tau(b, a_1, \pi_2^*, o)) + \gamma^{K-1}(U - L)$. This implies that an upper bound on the utility that σ_1 achieves against $\sigma_2^{b,K}$ (i.e., the strategy

[6] In other words, this is a performance guarantee for the (minimizing) player 2.

corresponding to player 2 following $\texttt{Act}(b)$ for K stages) is

$$\begin{aligned}
&\mathbb{E}_{b,\sigma_1,\pi_2^*}[R(s,a_1,a_2)] + \gamma \mathbb{E}_{b,\sigma_1,\pi_2^*}[V(\tau(b,a_1,\pi_2^*,o)) + \gamma^{K-1}(U-L)] \\
&\quad = \mathbb{E}_{b,\sigma_1,\pi_2^*}[R(s,a_1,a_2)] + \gamma \sum_{(a_1,o)\in A_1\times O} \mathbb{P}_{b,\sigma_1,\pi_2^*}[a_1,o] \cdot [V(\tau(b,a_1,\pi_2^*,o)) + \gamma^{K-1}(U-L)] \ .
\end{aligned}$$

By allowing the player 1 to maximize over σ_1, we get an upper bound on the value $\mathrm{val}_2(\sigma_2^{b,K}, b)$ strategy $\sigma_2^{b,K}$ guarantees when the belief of player 1 is b.

$$\begin{aligned}
&\mathrm{val}_2(\sigma_2^{b,K}, b) \leq \\
&\quad \leq \sup_{\sigma_1\in\Sigma_1} \Big[\mathbb{E}_{b,\sigma_1,\pi_2^*}[R(s,a_1,a_2)] + \\
&\qquad\qquad + \gamma \sum_{(a_1,o)\in A_1\times O} \mathbb{P}_{b,\sigma_1,\pi_2^*}[a_1,o] \cdot [V(\tau(b,a_1,\pi_2^*,o)) + \gamma^{K-1}(U-L)]\Big] \\
&\quad = \max_{\pi_1\in\Pi_1} \Big[\mathbb{E}_{b,\pi_1,\pi_2^*}[R(s,a_1,a_2)] + \gamma \sum_{(a_1,o)\in A_1\times O} \mathbb{P}_{b,\pi_1,\pi_2^*}[a_1,o] \cdot V(\tau(b,a_1,\pi_2^*,o))\Big] + \gamma^K(U-L) \\
&\quad = \max_{\pi_1\in\Pi_1} u^{V,b}(\pi_1,\pi_2^*) + \gamma^K(U-L)
\end{aligned}$$

Using the fact that π_2^* is the optimal strategy in the stage game $[HV](b)$, the definition of the stage game's value, and the fact that V is min-justified, we get

$$\begin{aligned}
\max_{\pi_1\in\Pi_1} u^{V,b}(\pi_1,\pi_2^*) + \gamma^K(U-L) &= \min_{\pi_2\in\Pi_2} \max_{\pi_1\in\Pi_2} u^{V,b}(\pi_1,\pi_2) + \gamma^K(U-L) \\
&= [HV](b) + \gamma^K(U-L) \leq V(b) + \gamma^K(U-L) \ .
\end{aligned}$$

Hence, the utility player 1 with belief b can achieve against player 2 who follows strategy $\sigma_2^{b,K}$ is at most $V(b) + \gamma^K(U-L)$, and we have $\mathrm{val}_2(\sigma_2^{b,K}, b) \leq V(b) + \gamma^K(U-L)$ which completes the induction step.

Now, similarly to the proof of Theorem 3.29, when player 2 follows $\texttt{Act}(b^{\text{init}})$ for *infinitely* many stages (i.e., plays strategy σ_2 from the theorem), player 1 is able to achieve utility at most

$$\mathrm{val}_2(\sigma_2, b^{\text{init}}) = \lim_{K\to\infty} \mathrm{val}_2(\sigma_2^{b^{\text{init}},K}, b^{\text{init}}) \leq \lim_{K\to\infty} [V(b^{\text{init}}) + \gamma^K(U-L)] = V(b^{\text{init}})$$

which completes the proof. □

3.7.4 Using Value Functions V_{LB}^{Γ} and V_{UB}^{Υ} to Play the Game

In Sections 3.7.2 and 3.7.3, we have shown that we can obtain strategies to play the game when the value functions are max-justified or min-justified, respectively. In this section, we will show that the heuristic search value iteration algorithm for solving one-sided POSGs (Section 3.6) generates value functions with these properties. Namely, at any time, the lower bound V_{LB}^{Γ} is max-justified value function by the set of α-vectors $\mathsf{Conv}(\Gamma)$, and the upper bound V_{UB}^{Υ} is min-justified.

This allows us to derive two important properties of the algorithm. First, since Theorem 3.25 guarantees that the algorithm terminates with $V^{\Upsilon}_{\mathrm{UB}}(b^{\mathrm{init}}) - V^{\Gamma}_{\mathrm{LB}}(b^{\mathrm{init}}) \leq \varepsilon$, we can use the resulting value functions V^{Γ}_{LB} (represented by Γ) and $V^{\Upsilon}_{\mathrm{UB}}$ to obtain ε-Nash equilibrium strategies for both players. Next, we can also run the algorithm in anytime fashion, and since the bounds V^{Γ}_{LB} and $V^{\Upsilon}_{\mathrm{UB}}$ satisfy the properties at any point of time, use these bounds to extract the strategies with performance guarantees.

We will first prove that at any point of time in the execution of Algorithm 3.1, the lower bound V^{Γ}_{LB} is max-justified by the set $\mathsf{Conv}(\Gamma)$, and the upper bound $V^{\Upsilon}_{\mathrm{UB}}$ is a min-justified value function. To prove this, it suffices to show that the initial value functions satisfy the property, and that the property is preserved after any sequence of point-based updates performed on V^{Γ}_{LB} and $V^{\Upsilon}_{\mathrm{UB}}$.

Lemma 3.32. *Let Γ be the set of α-vectors that have been generated at any time during the execution of the HSVI algorithm for one-sided POSGs (Algorithm 3.1). Then the lower bound V^{Γ}_{LB} is max-justified by the set $\mathsf{Conv}(\Gamma)$.*

Proof. Observe that the set Γ is modified only by the point-based updates on lines 8 and 12 of Algorithm 3.1. To this end, in order to prove the claim, it suffices to show that (1) the initial lower bound V^{Γ}_{LB} is max-justified by the set $\mathsf{Conv}(\Gamma) = \Gamma = \{\mathrm{val}^{\sigma_1^{\mathrm{unif}}}\}$ and that (2) if V^{Γ}_{LB} is max-justified by $\mathsf{Conv}(\Gamma)$ then any point-based update results in a value function $V^{\Gamma'}_{\mathrm{LB}}$ that is max-justified by the set $\mathsf{Conv}(\Gamma')$.

First, let us show that the initial lower bound V^{Γ}_{LB} is max-justified by the initial set of α-vectors $\Gamma = \{\mathrm{val}^{\sigma_1^{\mathrm{unif}}}\}$ (and therefore also by $\mathsf{Conv}(\Gamma) = \Gamma$). Clearly, $\sigma_1^{\mathrm{unif}} = \mathsf{comp}(\pi_1^{\mathrm{unif}}, \zeta^{\mathrm{unif}})$, i.e., the uniform strategy σ_1^{unif} can be composed from a uniform stage strategy π_1^{unif} for the first stage of the game, and playing uniform strategy $\zeta^{\mathrm{unif}}_{a_1,o} = \sigma_1^{\mathrm{unif}}$ in every subgame after playing and observing (a_1, o). Hence, $\mathrm{val}^{\sigma_1^{\mathrm{unif}}} = \mathsf{valcomp}(\pi_1^{\mathrm{unif}}, \overline{\alpha}^{\mathrm{unif}})$ for $\alpha^{\mathrm{unif}}_{a_1,o} = \mathrm{val}^{\sigma_1^{\mathrm{unif}}}$ and the initial V^{Γ}_{LB} is therefore max-justified by the set $\mathsf{Conv}(\Gamma) = \Gamma = \{\mathrm{val}^{\sigma_1^{\mathrm{unif}}}\}$.

Now let us assume a lower bound V^{Γ}_{LB} considered by the Algorithm 3.1 and assume that it is max-justified by a set $\mathsf{Conv}(\Gamma)$. The point-based update constructs a set $\Gamma' = \Gamma \cup \{\mathsf{valcomp}(\pi_1, \overline{\alpha})\}$ for some $\pi_1 \in \Pi_1$ and $\overline{\alpha} \in \mathsf{Conv}(\Gamma)^{A_1 \times O}$, see Equation (3.49). Since V^{Γ}_{LB} was max-justified by $\mathsf{Conv}(\Gamma)$, we know that for every $\alpha \in \mathsf{Conv}(\Gamma)$ there exists $\pi_1' \in \Pi_1$, $\overline{\alpha}' \in \mathsf{Conv}(\Gamma)^{A_1 \times O}$ such that $\mathsf{valcomp}(\pi_1', \overline{\alpha}') \geq \alpha$. The same holds for the newly constructed α vector $\mathsf{valcomp}(\pi_1, \overline{\alpha})$, and $V^{\Gamma'}_{\mathrm{LB}}$ is therefore max-justified by $\mathsf{Conv}(\Gamma) \cup \{\mathsf{valcomp}(\pi_1, \overline{\alpha})\}$. By Lemma 3.28, we also have that $V^{\Gamma'}_{\mathrm{LB}}$ is max-justified by $\mathsf{Conv}(\mathsf{Conv}(\Gamma) \cup \{\mathsf{valcomp}(\pi_1, \overline{\alpha})\}) = \mathsf{Conv}(\Gamma')$. Every point-based update thus results in a value function $V^{\Gamma'}_{\mathrm{LB}}$ which is max-justified by $\mathsf{Conv}(\Gamma')$ which completes the proof. □

Lemma 3.33. *Let $V^{\Upsilon}_{\mathrm{UB}}$ be the upper bound considered at any time of the execution of the HSVI algorithm for one-sided POSGs (Algorithm 3.1). Then $V^{\Upsilon}_{\mathrm{UB}}$ is min-justified.*

Proof. Upper bound $V^{\Upsilon}_{\mathrm{UB}}$ is only modified by means of point-based update on lines 8 and 12 of Algorithm 3.1. Therefore, it suffices to show that (1) the initial upper bound is min-justified and that (2) the upper bound $V^{\Upsilon'}_{\mathrm{UB}}$ resulting from applying a point-based update on a min-justified upper bound $V^{\Upsilon}_{\mathrm{UB}}$ is min-justified as well.

First, let us prove that the initial value function $V^{\Upsilon}_{\mathrm{UB}}$ is min-justified. Initially, $V^{\Upsilon}_{\mathrm{UB}}(b)$ is set to the value of a *perfect information* version of the game, where the imperfectly informed player 1 gets to know the initial state of the game. By removing this information from player 1, the utility player 1 is able to achieve can only decrease. Hence $[HV^{\Upsilon}_{\mathrm{UB}}](b) \leq V^{\Upsilon}_{\mathrm{UB}}(b)$, and the initial value function $V^{\Upsilon}_{\mathrm{UB}}(b)$ is therefore min-justified.

Now, let us consider an upper bound $V^{\Upsilon}_{\mathrm{UB}}$ represented by a set $\Upsilon = \{(b_i, y_i) \mid 1 \leq i \leq k\}$ that is considered by the Algorithm 3.1 and let us assume that $V^{\Upsilon}_{\mathrm{UB}}$ is min-justified. Consider that a point-based update in b_{k+1} is to be performed. We show that the function $V^{\Upsilon'}_{\mathrm{UB}}$ resulting from the point-based update in b_{k+1} is min-justified as well. Recall that $\Upsilon' = \Upsilon \cup \{(b_{k+1}, y_{k+1})\}$ and $y_{k+1} = [HV^{\Upsilon}_{\mathrm{UB}}](b_{k+1})$. Clearly, since $\Upsilon \subset \Upsilon'$, it holds $V^{\Upsilon'}_{\mathrm{UB}}(b) \leq V^{\Upsilon}_{\mathrm{UB}}(b)$ and $[HV^{\Upsilon'}_{\mathrm{UB}}](b) \leq [HV^{\Upsilon}_{\mathrm{UB}}](b)$ for every $b \in \Delta(S)$. Due to this and since $V^{\Upsilon}_{\mathrm{UB}}$ is assumed to be min-justified, we have $y_i \geq [HV^{\Upsilon'}_{\mathrm{UB}}](b)$ for every $1 \leq i \leq k+1$. We will now show that $V^{\Upsilon'}_{\mathrm{UB}}$ is min-justified by showing that $[HV^{\Upsilon'}_{\mathrm{UB}}](b) \leq V^{\Upsilon'}_{\mathrm{UB}}(b)$ holds for arbitrary belief $b \in \Delta$. Let λ_i and b' correspond to the optimal solution of the linear program (3.44) for solving $V^{\Upsilon'}_{\mathrm{UB}}(b)$. We have

$$
\begin{aligned}
V^{\Upsilon'}_{\mathrm{UB}}(b) &= \sum_{i=1}^{k+1} \lambda_i y_i + \delta \|b - b'\|_1 \ (\lambda_i \text{ and } b' \text{ represent an optimal solution of } V^{\Upsilon'}_{\mathrm{UB}}(b)) \\
&\geq \sum_{i=1}^{|\Upsilon|} \lambda_i \cdot [HV^{\Upsilon'}_{\mathrm{UB}}](b_i) + \delta \|b - b'\|_1 \\
&\geq [HV^{\Upsilon'}_{\mathrm{UB}}](b') + \delta \|b - b'\|_1 \ (HV^{\Upsilon'}_{\mathrm{UB}} \text{ is convex, see Proposition 3.14}) \\
&\geq [HV^{\Upsilon'}_{\mathrm{UB}}](b) \ (V^{\Upsilon'}_{\mathrm{UB}} \text{ is } \delta\text{-Lipschitz continuous, and hence according to} \\
&\qquad \text{Proposition 3.14 also } HV^{\Upsilon'}_{\mathrm{UB}} \text{ is}) \ .
\end{aligned}
$$

This shows that any point-based update results in a min-justified value function $V^{\Upsilon'}_{\mathrm{UB}}$. Therefore the Algorithm 3.1 only considers upper bounds $V^{\Upsilon}_{\mathrm{UB}}$ that are min-justified. □

We are now ready to show that the Algorithm 3.1 can be used to obtain ε-Nash equilibrium strategies for the given one-sided partially observable stochastic game.

Theorem 3.34. *Consider that Algorithm 3.1 has terminated. Then we can use Algorithms 3.2 and 3.3 to obtain ε-Nash equilibrium strategies for both players.*

Proof. According to Theorem 3.25, the Algorithm 3.1 terminates and the value functions V^{Γ}_{LB} and $V^{\Upsilon}_{\mathrm{UB}}$ that result from the execution of the algorithm satisfy $V^{\Upsilon}_{\mathrm{UB}}(b^{\mathrm{init}}) - V^{\Gamma}_{\mathrm{LB}}(b^{\mathrm{init}}) \leq \varepsilon$. Furthermore, we know that lower bound V^{Γ}_{LB} is max-justified by the set Γ resulting from the execution of Algorithm 3.1 (Lemma 3.32), and the upper bound $V^{\Upsilon}_{\mathrm{UB}}$ is

min-justified (Lemma 3.33). We can therefore use Algorithm 3.2 to obtain a strategy for player 1 that achieves utility of at least $V^{\Gamma}_{\mathrm{LB}}(b^{\mathrm{init}})$ for player 1 (Corollary 3.30). Similarly, we can use Algorithm 3.3 to obtain a strategy for player 2 that ensures that the utility of the player 1 will be at most $V^{\Upsilon}_{\mathrm{UB}}(b^{\mathrm{init}})$ (Theorem 3.31). Since $V^{\Upsilon}_{\mathrm{UB}}(b^{\mathrm{init}}) - V^{\Gamma}_{\mathrm{LB}}(b^{\mathrm{init}}) \leq \varepsilon$ these strategies form ε-Nash equilibrium of the game. □

3.8 Experimental evaluation

In this section, we focus on the experimental evaluation of the heuristic search value iteration algorithm for solving one-sided partially observable stochastic games from Section 3.6. We demonstrate the scalability of the algorithm in three security domains. Rewards in all of the domains have been scaled to the interval $[0, 100]$ or $[-100, 0]$, respectively, and we report the runtime required to reach $V^{\Upsilon}_{\mathrm{UB}}(b^{\mathrm{init}}) - V^{\Gamma}_{\mathrm{LB}}(b^{\mathrm{init}}) \leq 1$. We first outline the details of our experimental setup.

3.8.1 Algorithm Settings

Compared to the version of the HSVI algorithm presented in Section 3.6, we adopt several modifications to improve the scalability of the algorithm. In this section, we provide description of these modifications, and we show that the theoretical guarantees of the algorithm still hold.

Pruning the Sets Γ and Υ Each time a point-based update is performed, the size of the sets Γ and Υ used to represent value functions V^{Γ}_{LB} and $V^{\Upsilon}_{\mathrm{UB}}$ increases. As new elements are generated, some of the elements in these sets may become unnecessary for the representation of the bounds V^{Γ}_{LB} and $V^{\Upsilon}_{\mathrm{UB}}$. To prevent unnecessary growth of the size of the representation, we remove such obsolete elements. Whenever a new α-vector $\mathsf{valcomp}(\pi_1^{\mathrm{LB}}, \overline{\alpha}^{\mathrm{LB}})$ is generated according to Equation (3.49), all dominated elements in the set Γ get removed and only those elements of $\alpha \in \Gamma$ that dominate $\mathsf{valcomp}(\pi_1^{\mathrm{LB}}, \overline{\alpha}^{\mathrm{LB}})$ in at least one state remain, i.e.,

$$\Gamma \coloneqq \{\alpha' \mid \alpha' \in \Gamma : \exists s \in S : \alpha'(s) > \mathsf{valcomp}(\pi_1^{\mathrm{LB}}, \overline{\alpha}^{\mathrm{LB}})(s)\} \cup \{\mathsf{valcomp}(\pi_1^{\mathrm{LB}}, \overline{\alpha}^{\mathrm{LB}})\} \,. \tag{3.57}$$

For the set Υ used to represent the upper bound $V^{\Upsilon}_{\mathrm{UB}}$, we use a batch approach instead of removing dominated elements immediately. We remove dominated elements every time the size of the set Υ increases by 10% compared to the size after the last pruning was performed (this is analogous to the pruning technique proposed in [Smith and Simmons, 2004]). Algorithm 3.4 inspects each point $(b_i, y_i) \in \Upsilon$ and checks whether it is needed to represent value function $V^{\Upsilon}_{\mathrm{UB}}$—and if it is not needed, the point gets removed.

Removing elements from sets Γ and Υ does not violate theoretical properties of the algorithm. First of all, only elements that are not necessary to represent currently considered bounds are removed—hence the values of value functions V^{Γ}_{LB} and $V^{\Upsilon}_{\mathrm{UB}}$ considered at each step of the algorithm remain unchanged, and the convergence property

Algorithm 3.4: Pruning set Υ representing the upper bound $V^{\Upsilon}_{\mathrm{UB}}$

input : Set Υ used to represent $V^{\Upsilon}_{\mathrm{UB}}$

for $(b_i, y_i) \in \Upsilon$ **do**

  if $y_i > V^{\Upsilon}_{\mathrm{UB}}(b_i)$ **then** $\Upsilon \coloneqq \Upsilon \setminus \{(b_i, y_i)\}$

is hence retained. Furthermore, we can still use pruned value functions to extract strategies with guaranteed performance. Since the resulting upper bound value function $V^{\Upsilon}_{\mathrm{UB}}$ is identical to the one obtained without pruning, it is still min-justified, and can be used to obtain strategy of the minimizing player 2 with guaranteed utility at most $V^{\Upsilon}_{\mathrm{UB}}(b^{\mathrm{init}})$ (Section 3.7.3). Similarly, V^{Γ}_{LB} can be used to obtain strategy of player 1 (Section 3.7.2). Despite the fact that the resulting set Γ of α-vectors is different from the set constructed by Algorithm 3.1 when no pruning is used, we can see that for every missing element α' there has to exist an element α such that $\alpha \geq \alpha'$ (see Equation (3.57)). Therefore, we can always replace missing α-vectors in value compositions (i.e., linear functions $\alpha^{a_1,o}$) without decreasing the values of the resulting value composition—and hence V^{Γ}_{LB} remains max-justified by the set of α-vectors $\mathsf{Conv}(\Gamma)$.

Partitioning States and Value Functions In many games, even the imperfectly informed player 1 has access to some information about the game. For example, in the pursuit-evasion games we discuss below, the pursuer *knows* his position—and representing his uncertainty about his position within the belief is unnecessary. To reduce the dimension of the beliefs, we allow for partitioning states into disjoint sets such that the imperfectly informed player 1 always *knows* which set he is currently in. Formally, let $S = \bigcup_{i=1}^{K} S_i$ such that $S_i \cap S_j = \varnothing$ for every $i \neq j$. Player 1 has to know the initial partition, i.e., $\mathsf{Supp}(b^{\mathrm{init}}) \subseteq S_i$ for some $1 \leq i \leq K$. Furthermore, he has to be able to infer which partition he is in at any time, i.e., for every belief b over a partition S_i (i.e., $\mathsf{Supp}(b) \subseteq S_i$), every achievable action-observation pair (a_1, o) and every stage strategy $\pi_2 \in \Pi_2$ of player 2, we have $\mathsf{Supp}(\tau(b, a_1, \pi_2, o)) \subseteq S_j$ for some $1 \leq j \leq K$. We use $T(S_i, a_1, o)$ to denote such S_j.

This partitioning allows for reducing the size of linear program (3.37) used to compute stage game solutions. Namely, the quantification over $s \in S$ can be replaced by $s \in S_i$, where S_i is the current partition. Furthermore, since also the partition of the next stage has to be known, we can also replace $(a_1, o, s') \in A_1 \times O \times S$ by $(a_1, o, s') \in A_1 \times O \times T(S_i, a_1, o)$.

Parameters and Hardware We use value iteration for stochastic games, or MDPs, respectively, to initialize the upper and lower bounds. The upper bound is initialized by solving a perfect-information variant of the game (see Section 3.6.1). The lower bound is computed by fixing the uniform strategy σ_1^{unif} for player 1 and solving the resulting Markov decision process from the perspective of player 2. We terminate the algorithms when either change in valuations between iterations of value iteration is lower than 0.025, or $20s$ time limit has expired. The initialization time is included in the computation times of the HSVI algorithm.

We use $\varepsilon = 1$. However, similarly to Smith and Simmons [2004], we adjust ε in each iteration, and we get ε_{imm} that is about to be used in the current iteration using formula $\varepsilon_{\text{imm}} = 0.25 + \eta(V^{\Upsilon}_{\text{UB}}(b^{\text{init}}) - V^{\Gamma}_{\text{LB}}(b^{\text{init}}) - 0.25)$ with $\eta = 0.9$. We set the parameter D to the largest value such that $\rho(t) \geq 0.25^{-t}$ holds for every $t \geq 0$.

3.8.2 Experimental Results

We now turn our attention to the discussion of experimental results. We introduce the domains that have been used in the experiments and comment on the scalability of the proposed algorithm.

Pursuit-Evasion Games (inspired by [Chung et al., 2011; Isler and Karnad, 2008]) In pursuit-evasion games, a team of K centrally controlled pursuers (we consider a team of $K = 2$) is trying to locate and capture the evader—who is trying to avoid getting captured. The game is played on a grid (in our case the size of the grid is $3 \times N$), where the pursuers start in the top-left corner of the grid, while the evader starts in the bottom-right corner. In each step, the units move to one of their adjacent locations (i.e., the actions of the evader are $A_2 = \{\text{left}, \text{right}, \text{up}, \text{down}\}$, while the actions available to the team of pursuers are joint actions for all units in the team, $A_1 = (A_2)^K$). The game ends when one of the units from the team of pursuers enters the same cell as the evader—and the team of pursuers (player 1) then receives a reward of +100. The reward for all other transitions in the game is zero. The pursuer knows the location of their units but the current location of the evader is not known.

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 $\varepsilon = 1$. Sizes of the games range from 143 states and 2671 transitions to 1299 states and 34807 transitions.

Search Games (inspired by [Bošanský et al., 2014]) In search games that model intrusion, the defender patrols checkpoint zones (see Figure 3.2a, the zones are marked with box). The attacker aims to cross the graph, while not being captured by the defender. She can either wait for one move to conceal her presence (and clean up the trace), or move further. Each unit of the defender is able to move to adjacent nodes within its assigned zone. The goal of the attacker is to cross the graph to reach node marked by T without encountering any unit of the defender. If she manages to do so, the defender receives a reward of -100.

We consider games with 2 checkpoint zones with varying number of nodes in a zone W (i.e. width of the graph) and 2 configurations of the defending forces—with one defender in each of the checkpoint zones (we denote this configuration 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 3.2b (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

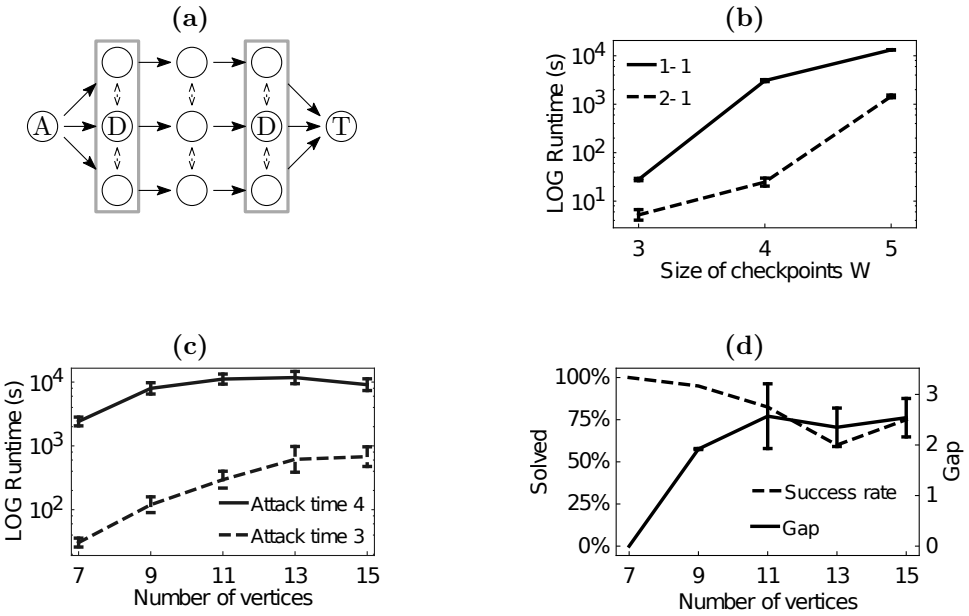


Figure 3.2: (a) Intrusion-search game: $W = 3$, configuration 1-1: A denotes initial position of the attacker, D initial positions of defender's units, T is attacker's target (b) Intrusion-search games with 2 zones, each with W vertices: Time to reach $V^{\Upsilon}_{\mathrm{UB}}(b^{\mathrm{init}}) - V^{\Gamma}_{\mathrm{LB}}(b^{\mathrm{init}}) \leq 1$ (c) Patrolling games played on graphs generated from $ER(0.25)$: Time to reach $V^{\Upsilon}_{\mathrm{UB}}(b^{\mathrm{init}}) - V^{\Gamma}_{\mathrm{LB}}(b^{\mathrm{init}}) \leq 1$ (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

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 (inspired by [Basilico et al., 2009a; Vorobeychik et al., 2014]) In patrolling game, a patroller (player 1) aims to protect a set of targets V. The targets are represented by vertices of a graph, and the possible movements of the patroller are represented by the edges of the graph. The attacker observes the movement of the patroller, and decides which target $v \in V$ he will attack, or whether he will postpone the decision. Once the attacker decides to attack a target v, the defender has $t_\times$ steps to reach the attacked vertex. If he fails to do so, he receives a negative reward $-C(v)$ associated to the target v—otherwise he successfully protects the target and the reward is zero. The patroller does not know whether and where the attack has already started.

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 $t_\times \in \{3, 4\}$ 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 3.2c). For attack time $t_\times = 4$, however, some number of instances failed to reach the precision $V^{\Upsilon}_{\mathrm{UB}}(b^{\mathrm{init}}) - V^{\Gamma}_{\mathrm{LB}}(b^{\mathrm{init}}) \leq 1$ 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 3.2d). For unsolved instances, mean $V^{\Upsilon}_{\mathrm{UB}}(b^{\mathrm{init}}) - V^{\Gamma}_{\mathrm{LB}}(b^{\mathrm{init}})$ after the cutoff after 10 hours is however reasonably small (also depicted in Figure 3.2d, see the solid line and right y-axes). The results include games with up to 856 states and 6409 transitions.

3.9 Cybersecurity Application: Active Deception

In Section 3.8, we have provided experimental evaluation of our novel algorithm for solving one-sided POSGs, and we have demonstrated its scalability. All the domains we have considered have their origin in security, and as the worst-case assumption the defender was the imperfectly informed player in the game. In [Horák et al., 2017b], we have used the model of one-sided POSGs in a different setting. In order to study active deception in computer network security, we assume that the attacker is the imperfectly informed player in the game, while the defender either has not detected the attacker yet (i.e., cannot actively influence the transitions), or he is able to reconstruct entire history of the attack (i.e., he has perfect information).

Underbrink [2016] classifies cyber deception into two broad categories—*passive* and *active deception*. The passive deception is targeted against attacker's reconnaissance efforts and relies on a proactively deployed *static* infrastructure of decoy systems, e.g., honeypots [Kreibich and Crowcroft, 2004; Spitzner, 2003] or fake documents [Bowen et al., 2009]. The active deception that we study in [Horák et al., 2017b], on the other hand, attempts to *interactively* engage the attacker who has been already detected by the sensing systems. The defender attempts to anticipate probable future actions of the

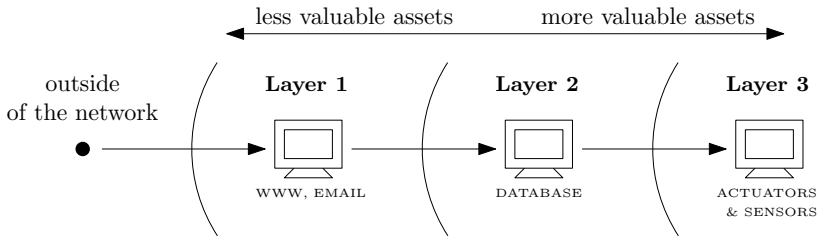


Figure 3.3: Network topology (attacker starts outside of the network and attempts to gain access to the most valuable assets in the network)

attacker and takes proactive countermeasures against them to prevent the attacker from achieving his goals. While a lot of work has been dedicated to understanding strategical aspects of passive deception techniques, e.g., [Durkota et al., 2015; Mohammadi et al., 2016; Zhu et al., 2012, 2013], very few works have focused on strategical aspects of active deception. Underbrink [2016] introduced the *Legerdemain* approach to active deception where critical assets in the network are secretly manipulated. In this work, however, it is assumed that the attacker will never realize that he is being deceived.

In contrast to the Legerdemain approach, in our game-theoretic model, the attacker is the imperfectly informed player in the one-sided POSG, and hence he actively reasons about the probability that he has been detected (and thus that he is subject to ongoing active deception). To this end, the strategy of the defender has to aim on creating and reinforcing the view of the attacker. We will now describe the game-theoretic model that we used for the case study of active deception, and we will discuss the results.

3.9.1 Game-Theoretic Model

We illustrate the concept of active deception using a network topology depicted in Figure 3.3. We use this topology 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. More critical assets are located in the deeper layers of the network. In our example, the second layer consists of data stores containing confidential data. 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 Layer 3 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 escalates 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 attempting to compromise 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 at a later time.

The second option is to cause smaller amount of damage while attempting not to attract defender's attention. The actions the attacker can use to this purpose include, e.g., a stealthy exfiltration of the data or a manipulation of the records in the database – for simplicity we refer to them collectively using the `exfiltrate` action. Nevertheless, even these careful options still run into a small risk of being detected. Moreover these options run into the risk that the defender will avert the damage resulting from them by means of active deception and possibly even use the fact that the attacker uses the `exfiltrate` action for his benefit (e.g. to collect evidence). 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 $d_{\texttt{comp}} = 0.2$, 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 $d_{\texttt{exf}} = 0.1$. 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 get 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 we are advocating 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`. We cannot, however, contain the more damaging `take down` action by engaging the attacker – the only way to prevent that kind of damage is to `block` the attacker in time. Note that both of these actions of the defender can be only used once the attacker got detected – otherwise, the

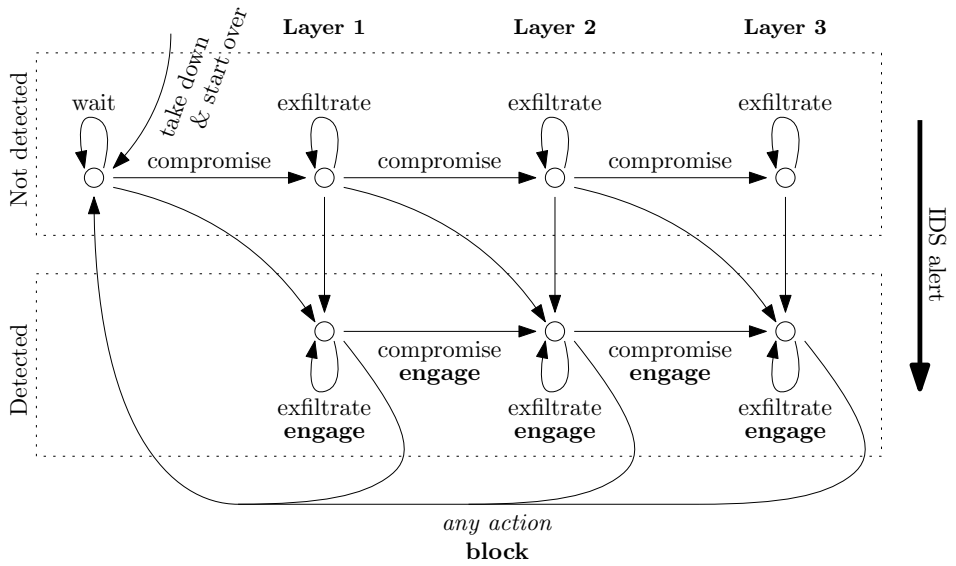


Figure 3.4: Transition system of a one-sided partially observable stochastic game representing attack on the network from Fig. 3.3. The attacker is uncertain whether he has been detected or not. The attacker can use the `take down` action in every layer. The `wait` action of the attacker has been omitted for clarity and is always applicable.

defender has to rely on the infrastructure of passive defensive systems as the attacker has to be detected first.

Game model We represent the above scenario as a one-sided POSG where the attacker is imperfectly informed about the state of the detection, i.e., he does not know whether he has been detected by the intrusion detection system (IDS) or not. The transition system of the game is shown in Figure 3.4. 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 decide between `engage` and `block` actions.

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 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 ($d_{\texttt{comp}}$ if the attacker uses `compromise` action, $d_{\texttt{exf}}$ 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

State ($s^{(t)}$)		Action		Defender's loss	
Position	Detected	Attacker ($a_A^{(t)}$)	Defender ($a_D^{(t)}$)	$L_D(s^{(t)}, a_A^{(t)}, a_D^{(t)})$	
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. 3.4. 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.

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. The game therefore belongs to the class of one-sided POSGs, and we can use techniques presented in this chapter to solve the game.

We consider discounted-sum utilities, and we use discount factor $\gamma = 0.95$ to account for the impatience of the attacker during the attack. In every stage of the game, the players choose their actions simultaneously, and the immediate loss of the defender $L_D(\cdot)$ (i.e., the reward of the attacker) follows Table 3.1. Here, we use a_A to denote actions of the attacker (i.e., the player 1 who maximizes the loss of the defender) and a_D to denote the actions of the defender who is aiming on minimizing the loss. The actual values for the case study have been devised based on a discussion with an expert, however, the model is general to capture any kind of preferences of the defender. We now present a brief intuition behind the game rewards.

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 ($L_1 = L_7 = -2$). Note that a negative loss is in fact a gain. 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 ($L_4 = -4$).

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.

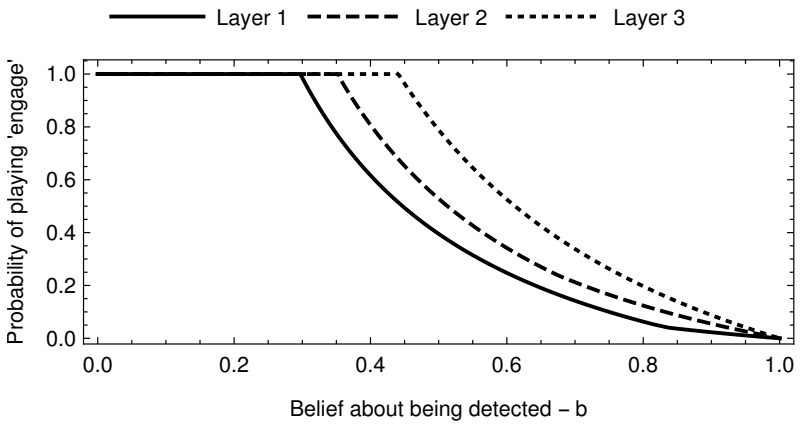


Figure 3.5: Optimal defense strategy for the network from Fig. 3.3. The optimal strategy of the defender is randomized and depends on the current position of the attacker (the layer he penetrated) and his belief about the detection state.

If the defender realizes that he is dealing with a malicious user, he can minimize or eliminate the risk of losing sensitive data, e.g. by presenting (partly) falsified data to the attacker, 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 $L_8 = 0$.

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 $L_3^i = L_6^i = 25i$ (when i represents the layer the attacker is in).

3.9.2 Results

In this section, we provide an overview of the numerical results obtained for the model introduced in Section 3.9.1 that we use in our case study. These results support the claim that the use of game-theory based active deception can significantly improve the security level of the network operation.

Optimal defense strategy 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 after detecting the attacker 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

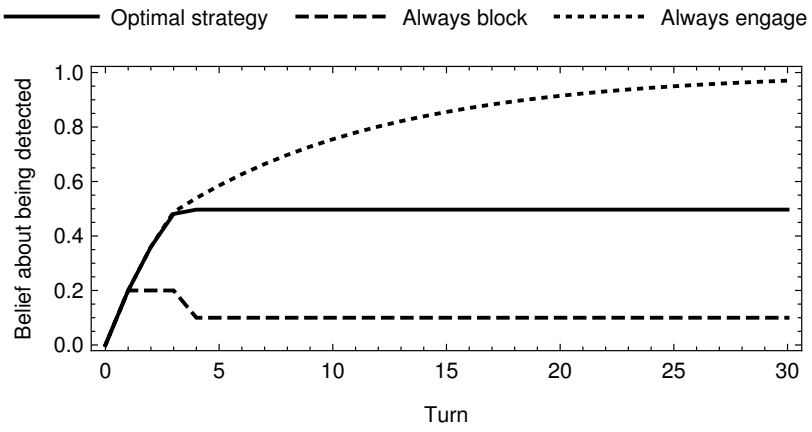


Figure 3.6: Evolution of attacker's belief over time. If we block the attacker immediately after detection, he remains highly confident that we cannot employ deceptive actions which allows him to perform long-term data exfiltration. If we always attempt to deceive the attacker by engaging him, he realizes that he likely faces a deception and decides to cause immediate damage – which is not prevented by the deceptive `engage` action.

make the deception believable, and the attacker would rather cause the damage by using the `takedown` action and forfeit his current attack attempt than battle the deception.

In Figure 3.5 we depict the optimal strategy of the defender based on the current belief of the attacker. Since the defender has only two actions available in the model, we express the probability of playing the `engage` action only. Observe that the more the attacker believes that he has been already detected, the less efficient the active deception is—and hence the defender is less likely to continue deceiving the attacker. Furthermore, observe that the closer the attacker is to his primary goals (or at least the closer he thinks to be), the less concerned he is about the fact that he might be deceived. Therefore, it is easier to deceive the attacker in deeper levels of the network—and the probability of playing the deceptive `engage` action in Layer 3 is thus higher than, e.g., in Layer 1.

To better understand the advantages of using the game-theoretic strategy to actively deceive the attacker compared to the baseline approaches, let us consider the evolution of the belief of the attacker in time shown in Figure 3.6. Observe that in case the defender uses `block` action immediately after he detects the attacker (strategy *Always block* in Figure 3.6), the attacker remains highly confident that he has not been detected yet, and thus is able to cause significant damage. Similarly, if the defender tries to `engage` the attacker at all times, the attacker can perform damaging actions `take down` when he becomes highly confident that the data that he exfiltrates is useless. In contrast, when employing the game-theoretic strategy, the belief of the attacker about being detected stabilizes at 0.4968. This is the right belief where the attacker still thinks that it is worth attempting to cause long-term damage by data exfiltration, despite being vulnerable to deceptive attempts of the defender.

Engaging the attacker 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 before we manage to detect him again.

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.

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 the time we decided to restrict attacker's access is in our case 4.577 time steps. In this time window, the defender gets additional alerts from the IDS which may help him decide about the credibility of the alert and thus assure that he is not about to block a legitimate user.

CHAPTER 4

Scaling Up

In Chapter 3, we have presented an algorithm for solving a class of one-sided partially observable stochastic games. This algorithm approximates the solution by approximating the optimal value function $V^* : \Delta(S) \to \mathbb{R}$ of the game, i.e., approximates the utility $V^*(b)$ player 1 is able to achieve in the game in case his belief is $b \in \Delta(S)$. The primary motivation for the class of one-sided POSGs comes from the domain of security. By assuming that the attacker is perfectly informed about the course of the game, we are able to come up with robust strategies that are guaranteed to work well even in cases where some (possibly unknown) information is leaking to the adversary.

Unfortunately, for many real-world large-scale security problems, the scalability of the basic version of the HSVI algorithm for solving one-sided POSGs (see Algorithm 3.1) is insufficient. In this chapter, we study an application of the algorithm to design dynamic defensive strategies against lateral movement, which is a well-known problem from the domain of cybersecurity [Noureddine et al., 2016; Kamdem et al., 2017]. In the model we consider, the state space of the game is exponential in the size of the analyzed network, and the basic algorithm can thus solve only the smallest instances. We first discuss the problem of lateral movement in Section 4.1 and provide an overview of the related work in Section 4.2. In Section 4.3, we introduce the idea of compact representation of the information the defender has and we extend the heuristic search value iteration algorithm for one-sided POSGs to use this representation. In Section 4.4, we discuss the idea of compact representation of the beliefs in the context of the lateral movement POSG. In Section 4.5, we further improve the scalability of the approach by using incremental strategy generation techniques. Finally, in Section 4.6, we conclude by experimental

---

This chapter is based on following publications:

[Horák et al., 2019b] Horák, K., Bošanský, B., Tomášek, P., Kiekintveld, C., and Kamhoua, C. (2019b). Optimizing honeypot strategies against dynamic lateral movement using partially observable stochastic games. *Computers & Security*, 87:101579 (40%)

[Horák et al., 2019a] Horák, K., Bošanský, B., Kiekintveld, C., and Kamhoua, C. (2019a). Compact Representation of Value Function in Partially Observable Stochastic Games. In *28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 350–356 (65%)

evaluation of our approach that demonstrates that the idea of compact representation can significantly improve the scalability of the solution algorithm.

4.1 Lateral Movement as Game Theoretic Problem

One of the most potent threats in cybersecurity are the Advanced Persistent Threat (APT) attackers. These are sophisticated (possibly state-sponsored) attackers who execute highly targeted, long-term, stealthy attacks against government, military, and corporate organizations. In many cases of known attacks, APTs have been successful at establishing a deep and persistent foothold in a target network and maintaining this presence for months or even years before being discovered [Mandiant Intelligence Center, 2013]. Another emerging challenge in cybersecurity is securing highly dynamic, diverse mobile networks against short-term but very intense attacks, as in the Internet of Battlefield Things (IoBT) [Abuzainab and Saad, 2018]. In both of these cases lateral movement plays a key role in the attack, as an attacker with a foothold needs to move to compromise additional network resources to achieve the goal of the attack. One of the methods a defender can use to detect and mitigate lateral movement is using honeypots and other deception technologies to detect, confuse, and redirect the attacker [Noureddine et al., 2016; Kamdem et al., 2017].

The resulting adversarial competition for control of a network between the defender and the attacker can be modeled as a game, where the defender is allocating defensive resources (e.g., honeypots), and the attacker is taking actions to gain greater control while remaining undetected. Game theory is becoming an increasingly important tool for optimizing cybersecurity resources, including strategic allocation of honeypots [Píbil et al., 2012; Kiekintveld et al., 2015; Durkota et al., 2015], allocating resources to perform the deep packet inspection [Vaněk et al., 2012], and modifying the structure or characteristics of the network [Jajodia et al., 2012; Cai et al., 2016; Nguyen et al., 2017].

Motivated by the scenarios of APT attackers and attacks on mobile IoBT infrastructure, we use partially observable stochastic games, where the interactions between the attacker and defender are *dynamic, involve uncertainty, and are long-lived*. First, in the realistic cases we want to model the interaction which is *dynamic*, in that the defender and attacker can observe and react to new information and the moves of the other player, allowing them to update their strategies over time. Second, the players have *imperfect information*; for example, the defender typically does not know exactly what set of resources an attacker may control at any given time. Finally, the interactions can be *very long* (as in the case of APTs), or *very frequent* (as in the case of IoBT networks under short but very intense attacks). Previous work using game theory to model lateral movement [Kamdem et al., 2017] and honeypot allocation [Píbil et al., 2012; Kiekintveld et al., 2015; Durkota et al., 2015; Wang et al., 2017] has not been able to address all of these problems.

In this chapter, we introduce a *lateral movement POSG*, a one-sided partially observable stochastic game that model an attacker moving laterally in a computer network. We assume that the attacker infiltrated some nodes in a network and aims to reach a certain

host (e.g., the main database). We model this problem as a two-player game on a graph between the attacker and the defender. The nodes of the graph correspond to hosts in the network, and the edges correspond to attacking another host using a particular vulnerability on the target host, which has an associated cost. The attacker sequentially chooses which hosts to attack to progress towards the goal. The defender chooses edges that will act as honeypots (i.e., fake vulnerabilities) which are able to detect certain moves by the attacker through the network and allow the defender to take additional mitigating actions. The defender can also change his honeypot deployment after any detection event to take into account the new piece of information about the attack.

There are two key challenges in solving the lateral movement POSG. First, the states in the game correspond to subsets of hosts/nodes the attacker has already managed to compromise. This means that the number of states, as well as the dimension of the belief space, is exponential in the size of the network. This means that representing, updating, and reasoning about the beliefs of the defender becomes computationally intractable for all but the smallest instances. The second challenge comes from the number of possible actions of the players. In the lateral movement POSG, the actions of the defender correspond to placing multiple honeypots on edges in a graph. Even if there are only 20 possible edges in a network and the defender can use 3 honeypots, the defender has up to $6.8 \cdot 10^3$ possible actions to choose from at a single decision point in the game. Since the game is dynamic with many decision points, this problem is exacerbated even further by the length of the game. We address both of these challenges and propose two key contributions for improving the scalability of the HSVI algorithm for solving the lateral movement POSG: (1) we replace the representation of beliefs over the exponential number of possible states with a *summarized abstraction* that captures key information but reduces the dimensionality of the beliefs; (2) we use an incremental strategy generation technique to iteratively expand the strategy space of the players in order to overcome large branching factor. We present general version of the algorithms and novel solution concepts, and then show how they can be applied specifically to the lateral movement POSG. We give theoretical justification and analysis of the key elements of the algorithm. In addition, we evaluate the performance of our algorithm empirically on realistic instances of the lateral movement POSG. Our algorithm is capable of finding strategies that are very near the optimal solutions for small cases where we can compute the true optimal value. It also scales dramatically better than the previous state of the art, allowing us to solve games with twenty or more nodes that are of practical significance. For example, we can find defense strategies in smaller computer networks or abstractions of larger network. With more aggressive approximation we could scale to even larger examples using the same algorithms.

4.2 Related Work

The concept of a honeypot has been around for more than 30 years in cybersecurity [Stoll, 1989]. They have been used for many different purposes, and have evolved to be much more

sophisticated with greater abilities to mimic real hosts and to capture useful information about attackers [Mairh et al., 2011; Nawrocki et al., 2016]. Recently, there have been a large number of game theory models developed to capture aspects of how honeypots and other deception methods can be strategically used to improve cybersecurity [Garg and Grosu, 2007; Píbil et al., 2012; Kiekintveld et al., 2015; Durkota et al., 2015; La et al., 2016; Wang et al., 2017].

While none of these previous models address the full scope of long-term dynamic games with uncertainty that we consider here, several have considered similar aspects of uncertainty or dynamic interactions. For example, Wang et al. [2017] investigate the use of honeypots in the smart grid to mitigate denial of service attacks through the lens of Bayesian games. Durkota et al. [2015] developed a model of honeypot allocation under uncertainty. La et al. [2016] model honeypots mitigating denial of service attacks in the Internet-of-Things domain. Du et al. [2017] tackle a similar "honeypots for denial of service attack" problem with Bayesian game modeling in the social networking domain. In addition, a number of works have consider deception as a signaling game (e.g., [Pawlick and Zhu, 2015]).

We use the idea of representing high-dimensional beliefs using low-dimensional characteristic vectors. Similar ideas has appeared in the literature focused on single-agent partially observable Markov decision processes (e.g., in [Roy et al., 2005; Li et al., 2010; Zhou et al., 2010]). The most general approach was to choose an abstraction based on Principal Components Analysis [Roy et al., 2005]. For our lateral movement POSG, we can exploit natural representation in marginal probabilities—i.e., reason about probability of each resource being infected as individual coordinates of the characteristic vector, instead of explicitly considering all possible subsets of infected resources.

Finally, we also use incremental strategy generation method that is often known as the double-oracle algorithm and often used for solving games with large combinatorial number of actions in security games (e.g., in [McMahan et al., 2003; Jain et al., 2011, 2013]), and more recently, the double oracle method is being also used in combination with reinforcement learning on domains where computing (approximate) best response is not possible [Lanctot et al., 2017; Oliehoek et al., 2017; Wang et al., 2019]. Double-oracle algorithm restricts the space of possible actions to choose from—the algorithm forms *a restricted stage game* that is iteratively expanded by adding new actions into the restricted game. These actions are often computed as *best responses* to the current strategy of the opponent in the restricted problem. In the worst case, all actions have to be added into the restricted problem. This, however, rarely happens in practice and double-oracle algorithms are often able to find an optimal strategy using only a fraction of all possible plans (see, for example, [Jain et al., 2011; Bošanský et al., 2014; Lanctot et al., 2017]). While the domain-independent idea of double-oracle algorithm is simple, our contribution is in (1) full integration of this methodology into the algorithm for solving stage games of lateral movement POSGs and (2) description of exact and heuristic oracle algorithms for determining which actions should be added into the restricted game.

4.3 Compact Representation of V^*

The dimension of the value function V^* of a one-sided POSG depends on the number of states, which can be very large in games like the lateral movement POSG where the number of states is exponential in the size of the network. We propose an abstraction scheme called *summarized abstraction* to decrease the dimensionality of the problem by creating a simplified representation of the beliefs over the state space.

We associate each belief $b \in \Delta(S)$ in the game with a *characteristic vector* $\chi^{(b)} = \mathbf{A} \cdot b$ (for some fixed matrix $\mathbf{A} \in \mathbb{R}^{k \times |S|}$ where $k \ll |S|$) and we define an (approximate) value function $\tilde{V}^* : \mathbb{R}^k \to \mathbb{R}$ over the space of characteristic vectors. We denote χ^{init} the characteristic vector corresponding to the initial belief b^{init}, i.e., $\chi^{\text{init}} = \mathbf{A} \cdot b^{\text{init}}$.

The main goal is to adapt algorithms based on value iteration to operate over the more compact space $\mathbb{R}^k$ instead of the original belief space $\Delta(S)$. First, we adapt the fixed point Equation (3.23c) for OS-POSGs to work with compact $\tilde{V}^*$ (instead of original V^*). We let the player 2 choose *any* belief that is consistent with the current characteristic vector χ (by adding an extra minimization term), and we replace the value of belief b, $V^*(b)$, by the value of its characterization $\tilde{V}^*(\mathbf{A} \cdot b)$. We also denote this compound function $\tilde{V}^* \circ \mathbf{A}$.

$$\tilde{V}^*(\chi) = \tilde{H}\tilde{V}^*(\chi) = \min_{b|\mathbf{A}b=\chi} \min_{\pi_2 \in \Pi_2} \max_{\pi_1 \in \Pi_1} \Big(\mathbb{E}_{b,\pi_1,\pi_2}[R(s,a_1,a_2)] + \\ + \gamma \sum_{a_1,o} \mathbb{P}_{b,\pi_1,\pi_2}[a_1,o] \cdot \tilde{V}^*(\mathbf{A} \cdot \tau(b,a_1,\pi_2,o)) \Big) \tag{4.1}$$

We will now prove that similarly to unabstracted one-sided POSGs, the operator $\tilde{H}$ defined by Equation (4.1) is a contraction mapping—and hence $\tilde{V}^*$ is the limit of repeated application of the operator $\tilde{H}$.

Lemma 4.1. *Operator $\tilde{H}$ is a contraction mapping with contractivity factor γ.*

Proof. Assume value functions $\tilde{V}_1, \tilde{V}_2 : \mathbb{R}^k \to \mathbb{R}$ in compact representation, and assume that $\|V_1 - V_2\|_\infty = \max_{\chi \in \mathbb{R}^k} |\tilde{V}_1(\chi) - \tilde{V}_2(\chi)| \leq C$. We will show that $|\tilde{H}\tilde{V}_1(\chi) - \tilde{H}\tilde{V}_2(\chi)| \leq \gamma C$ for arbitrary $\chi \in \mathbb{R}^k$. Without loss of generality, we will assume that $\tilde{H}\tilde{V}_1(\chi) \leq \tilde{H}\tilde{V}_2(\chi)$. Let (b, π_2) be the optimal solution of $\tilde{H}\tilde{V}_1(\chi)$. We will now show that by using (b, π_2) instead of the optimal minimizer for $\tilde{H}\tilde{V}_2(\chi)$ guarantees a solution with value at most $\tilde{H}\tilde{V}_1(\chi) + \gamma C$—and using the optimal minimizer instead of (b, π_2) can only decrease the value, and hence $\tilde{H}\tilde{V}_2(\chi) \leq \tilde{H}\tilde{V}_1(\chi) + \gamma C$.

Let us fix (b, π_2) in $\tilde{H}\tilde{V}_2(\chi)$, and let us consider the objective values $u_1(\pi_1)$ and $u_2(\pi_1)$ of using π_1 against (b, π_2) in $\tilde{H}\tilde{V}_1$ and $\tilde{H}\tilde{V}_2$, respectively. It holds

$$\begin{aligned} u_2(\pi_1) - u_1(\pi_1) &= \gamma \sum_{a_1,o} \mathbb{P}_{b,\pi_1,\pi_2}[a_1,o] \Big[\tilde{V}_2(\mathbf{A} \cdot \tau(b,a_1,\pi_2,o)) - \tilde{V}_1(\mathbf{A} \cdot \tau(b,a_1,\pi_2,o)) \Big] \\ &\leq \gamma \sum_{a_1,o} \mathbb{P}_{b,\pi_1,\pi_2}[a_1,o] \Big| \tilde{V}_2(\mathbf{A} \cdot \tau(b,a_1,\pi_2,o)) - \tilde{V}_1(\mathbf{A} \cdot \tau(b,a_1,\pi_2,o)) \Big| \end{aligned}$$

$$\leq \gamma \sum_{a_1,o} \mathbb{P}_{b,\pi_1,\pi_2}[a_1, o] \cdot C = \gamma C \ .$$

Now,

$$\tilde{H}\tilde{V}_2(\chi) \leq \max_{\pi_1} u_2(\pi_1) \leq \max_{\pi_1}[u_1(\pi_1) + \gamma C] = \max_{\pi_1} u_1(\pi_1) + \gamma C = \tilde{H}\tilde{V}_1(\chi) + \gamma C$$

which completes the proof. □

Theorem 4.2. *Value function $\tilde{V}^*$ is uniquely defined by the fixed point Equation* (4.1)*.*

Proof. The theorem is a direct consequence of Lemma 4.1 and Banach's fixed point theorem [Ciesielski et al., 2007]. □

From the perspective of security problems, an important aspect of defining value function $\tilde{V}^*$ using the fixed point Equation (4.1) is the ability to obtain worst-case type of guarantees. Namely, the utility represented by the abstracted value function $\tilde{V}^*(\mathbf{A} \cdot b)$ always lower-bounds the utility $V^*(b)$ of the unabstracted game.

Theorem 4.3. $\tilde{V}^*(\chi^{(b)}) \leq V^*(b)$ *for every* $b \in \Delta(S)$*.*

Proof. To prove this claim, we consider sequences of value functions $\{\tilde{V}_t\}_{t=1}^{\infty}$ and $\{V_t\}_{t=1}^{\infty}$, and we will use induction to show that $\tilde{V}_t(\chi^{(b)}) \leq V_t(b)$. Let $\tilde{V}_0$ be arbitrary and let V_0 of the original game be $V_0(b) = \tilde{V}_0(\chi^{(b)})$ (i.e., $\tilde{V}_0 \leq V_0$).

Assume that the induction hypothesis holds for t, i.e., $\tilde{V}_t(\chi^{(b)}) = [\tilde{V}_t \circ \mathbf{A}](b) \leq V_t(b)$ for every $b \in \Delta(S)$. Therefore, we have also that $H(\tilde{V}_t \circ \mathbf{A}) \leq H(V_t)$. The extra minimization over beliefs b in $\tilde{H}\tilde{V}$ can only decrease the utility of the stage game and hence

$$\tilde{V}_{t+1}(\chi^{(b)}) = \tilde{H}\tilde{V}_t(\chi^{(b)}) \leq H(\tilde{V}_t \circ \mathbf{A})(b) \leq HV_t(b) = V_{t+1}(b) \ . \tag{4.3}$$

The property holds even in the limit, and hence $\tilde{V}^*(\chi^{(b)}) \leq V^*(b)$ for every $b \in \Delta(S)$. □

4.3.1 Solving $\tilde{H}\tilde{V}$

Similarly to one-sided POSG, in case $\tilde{V}$ is a piecewise linear and convex (PWLC) function, the optimization problem $\tilde{H}\tilde{V}(\chi)$ defined in Equation (4.1) can also be solved using linear programming. Let us consider a value function $\tilde{V}$ represented as a point-wise maximum over a set Γ of affine functions $\alpha_i(\chi) = (\mathbf{a}^{(i)})^T\chi + z^{(i)}$.[1] We modify the linear

[1] Note that this representation will later be used to represent a lower bound $\tilde{V}_{\text{LB}}$ on $\tilde{V}^*$, similarly to unabstracted one-sided POSGs presented in Chapter 3.

program (3.38) by considering that the minimizing player 2 can choose the belief, and hence the belief b is a variable (constrained by χ),

$$\min_{V,\pi_2,\hat{\tau},b,\hat{\chi},q} \quad V \tag{4.4a}$$

$$\text{s.t.} \quad \text{constraints (3.38b), (3.38d), (3.38e), (3.38f)} \tag{4.4b}$$

$$\sum_s b(s) = 1 \tag{4.4c}$$

$$\mathbf{A} \cdot b = \chi \tag{4.4d}$$

$$b(s) \geq 0 \qquad \forall s\ , \tag{4.4e}$$

and we replace the constraint (3.38c) to account for different representation of $\tilde{V}$ by

$$\hat{\chi}^{a_1 o} = \mathbf{A} \cdot \hat{\tau}(b, a_1, \pi_2, o) \qquad \forall a_1, o \tag{4.4f}$$

$$q^{a_1 o} = \mathbf{1}^T \cdot \hat{\tau}(b, a_1, \pi_2, o) \qquad \forall a_1, o \tag{4.4g}$$

$$\hat{V}(a_1, o) \geq (\mathbf{a}^{(i)})^T \cdot \hat{\chi}^{a_1 o} + z^{(i)} \cdot q^{a_1 o} \qquad \forall a_1, o, \alpha_i\ , \tag{4.4h}$$

where $q^{a_1 o} = \mathbf{1}^T \cdot \hat{\tau}(b, a_1, \pi_2, o)$ is the probability that o is generated when player 1 uses action a_1. Recall that $\hat{\tau}(b, a_1, \pi_2, o)$ from constraint (3.38d) corresponds to the updated belief $\tau(b, a_1, \pi_2, o)$ multiplied by the probability $\mathbb{P}_b[o \mid a_1] = q^{a_1, o}$. Similarly, $\hat{\chi}^{a_1 o}$ is multiplied by $\mathbb{P}_b[o \mid a_1]$.

4.3.2 Properties of $\tilde{V}^*$

Observe that the only constraints in the linear program (4.4) where non-zero constant terms are involved are constraints (4.4c) and (4.4d). This allows us to prove that in case $\tilde{V}$ is PWLC, $\tilde{H}\tilde{V}$ is also PWLC.

Lemma 4.4. *Let $\tilde{V}$ be a piecewise linear and convex function. Then $\tilde{H}\tilde{V}$ is piecewise linear and convex function as well.*

Proof. Consider a dual formulation of the linear program (4.4). Since the only non-zero constant terms within constraints of the primal are 1 (constraint (4.4c)) and χ (constraints (4.4d)), the objective of the dual formulation is $o(\chi) = \chi^T \cdot \mathbf{a} + z$. Moreover, this is the only place where the characteristic vector χ occurs. Hence the polytope of the feasible solutions of the dual problem is identical for every characteristic vector $\chi \in \mathbb{R}^k$ and $o(\chi)$ (after fixing variables $\mathbf{a}$ and z to feasible values) forms a lower bound on the objective value of the solution for *arbitrary* χ. Since we maximize over all possible $o(\chi)$ in the dual, the objective value of the linear program (and also $\tilde{H}\tilde{V}$) is convex in the parameter χ.

The solution of a dual linear program to (4.4) can be found within finitely many vertices of the corresponding polytope induced by its constraints [Vanderbei, 2015]—corresponding to a finite number of possible objective functions $o(\chi)$. Hence the value

function $\tilde{H}\tilde{V}$ can be represented as a point-wise maximum over a finite set of linear functions $o(\chi)$, which is piecewise linear and convex. □

Since the optimal value function $\tilde{V}^*$ of compact OS-POSGs, is the limit point of applying $\tilde{H}$ iteratively, the Lemma 4.4 also implies that $\tilde{V}^*$ is convex.

Theorem 4.5. *The optimal value function $\tilde{V}^*$ of compact OS-POSGs is convex.*

Proof. Due to the contractivity of operator $\tilde{H}$ and Banach's fixed point theorem [Ciesielski et al., 2007], $\tilde{V}^*$ is the limit point of the repeated application of operator $\tilde{H}$. Let $\tilde{V}_0 : \mathbb{R}^k \to \mathbb{R}$ be an arbitrary piecewise linear and convex value function and let $\{\tilde{V}_i\}_{i=0}^{\infty}$ be a sequence of value functions such that $\tilde{V}_{i+1} = \tilde{H}\tilde{V}_i$. Lemma 4.4 implies that every $\tilde{V}_i$ is convex—and therefore also the limit point $\tilde{V}^*$ is convex. □

The convexity of the optimal value function $\tilde{V}^*$ of a compactly represented OS-POSG allows us to focus on piecewise linear and convex functions $\tilde{V}_{\mathrm{LB}}$ and $\tilde{V}_{\mathrm{UB}}$ to approximate $\tilde{V}^*$. This is again similar to the approach used to solve OS-POSGs in Chapter 3.

4.3.3 Stage Games

In the optimization problem (4.1), the player 2 first chooses the belief $b \in \Delta(S)$ satisfying $\mathbf{A}b = \chi$, and then chooses the stage strategy $\pi_2 \in \Pi_2$. This two-level optimization can, however, be replaced by letting the player 2 choose a *joint* probability distribution $\hat{\pi}_2 \in \Delta(S \times A_2)$ over states and actions of player 2 which satisfies the constraint induced by the characteristic vector χ, i.e.,

$$b_{\hat{\pi}_2}(s) = \sum_{a_2 \in A_2} \hat{\pi}_2(s \wedge a_2) \;\; \forall s \in S \qquad \text{and} \qquad \mathbf{A}b_{\hat{\pi}_2} = \chi \tag{4.5}$$

where $b_{\hat{\pi}_2}$ denotes the belief induced by the joint probability distribution $\hat{\pi}_2$. For simplicity, we use $\hat{\Pi}_2^{\chi}$ to denote the set of all joint probability distributions that satisfy constraint (4.5). We can use this representation of strategies of player 2 to define a stage game corresponding to the optimization problem $\tilde{H}\tilde{V}(\chi)$.

Definition 4.1 (Stage game in compact representation)**.** Let $\tilde{V}$ be a continuous convex value function in compact representation. A *stage game* with respect to the $\tilde{V}$ and characteristic vector $\chi \in \mathbb{R}^k$ is a two-player zero-sum game with strategy space Π_1 and $\hat{\Pi}_2^{\chi}$ for player 1 and player 2, respectively, and utility function

$$u^{\tilde{V},\chi}(\pi_1, \hat{\pi}_2) = \mathbb{E}_{\pi_1,\hat{\pi}_2}[R(s, a_1, a_2)] + \gamma \sum_{a_1,o} \mathbb{P}_{\pi_1,\hat{\pi}_2}[a_1, o] \cdot \tilde{V}(\mathbf{A} \cdot \tau(b_{\hat{\pi}_2}, a_1, \hat{\pi}_2, o)) \; . \tag{4.6}$$

We overload the notation and use $\tilde{H}\tilde{V}(\chi)$ to denote this stage game.

Algorithm 4.1: HSVI algorithm for one-sided POSGs when summarized abstraction is used.

1 Initialize $\tilde{V}_{\mathrm{LB}}$ and $\tilde{V}_{\mathrm{UB}}$ to lower and upper bound on $\tilde{V}^*$
2 **while** $\mathrm{excess}_0(\chi^{\mathrm{init}}) > 0$ **do**
3 `Explore`$(\chi^{\mathrm{init}}, 0)$
4 **return** $\tilde{V}_{\mathrm{LB}}$ *and* $\tilde{V}_{\mathrm{UB}}$
5 **procedure** `Explore`(χ_t, t)
6 $(b, \pi_2) \leftarrow$ optimal belief and stage strategy of player 2 in $\tilde{H}\tilde{V}_{\mathrm{LB}}(\chi)$
7 $\pi_1 \leftarrow$ optimal stage strategy of player 1 in $\tilde{H}\tilde{V}_{\mathrm{UB}}(\chi)$
8 Perform point-based updates of $\tilde{V}_{\mathrm{LB}}$ and $\tilde{V}_{\mathrm{UB}}$ at characteristic vector χ
9 $(a_1^*, o^*) \leftarrow$ select according to forward exploration heuristic
10 $\chi' \leftarrow \mathbf{A} \cdot \tau(b, a_1^*, \pi_2, o^*)$
11 **if** $\mathbb{P}_{b,\pi_1,\pi_2}[a_1^*, o^*] \cdot \mathrm{excess}_{t+1}(\chi') > 0$ **then**
12 `Explore`$(\chi', t+1)$
13 Perform point-based updates of $\tilde{V}_{\mathrm{LB}}$ and $\tilde{V}_{\mathrm{UB}}$ at characteristic vector χ

Following the proof of Theorem 3.15, it can be shown that the utility function $u^{\tilde{V},\chi}$ is convex in the strategies $\hat{\Pi}_2^\chi$ of the minimizing player 2 and concave in the strategies Π_1 of player 1. Therefore, the assumptions of the von Neumann's minimax theorem [von Neumann, 1928; Nikaido, 1953] apply, and minimax and maximin values of the stage games in compact representation coincide (and the optimal strategies form Nash equilibrium of the stage game).

4.3.4 HSVI Algorithm for Compact POSGs

The Algorithm 4.1 we propose for solving abstracted games is a modified version of the original heuristic search value iteration algorithm (HSVI) for solving unabstracted one-sided POSGs (see Algorithm 3.1). The key difference here is that we use the value functions $\tilde{V}_{\mathrm{LB}}$ and $\tilde{V}_{\mathrm{UB}}$ that are defined over $\mathbb{R}^k$ (instead of V_{LB}^Γ and V_{UB}^Υ defined over $\Delta(S)$) and we must have modified all parts of the algorithm to use the abstracted representation of the beliefs. We provide a high-level overview of the algorithm, and we provide details specifically for the case of the lateral movement POSG in Section 4.4.

First, we initialize bounds $\tilde{V}_{\mathrm{LB}}$ and $\tilde{V}_{\mathrm{UB}}$ (line 1) to valid piecewise linear and convex lower and upper bounds on $\tilde{V}^*$. Then, we perform a sequence of trials (lines 2–3) from the initial characteristic vector $\chi^{\mathrm{init}} = \mathbf{A} \cdot b^{\mathrm{init}}$ until the desired precision $\varepsilon > 0$ is reached.

Similarly to unabstracted one-sided POSGs, in each stage (i.e., call to the procedure `Explore`) we first compute optimal *optimistic* strategies of player 1 and player 2. The strategy of the minimizing player 2 is represented as the optimal choice of belief b and stage strategy π_2 when evaluated with respect to the lower bound $\tilde{V}_{\mathrm{LB}}$ on $\tilde{V}^*$. Conversely, the strategy of player 1 originate from solving the stage game $\tilde{H}\tilde{V}_{\mathrm{UB}}(\chi)$ with respect to the upper bound $\tilde{V}_{\mathrm{UB}}$. Next, we choose the action a_1^* of player 1 and the observation o^* (lines 9–10) so that the excess approximation error $\mathrm{excess}_{t+1}(\chi') =$

$\tilde{V}_{\mathrm{UB}}(\chi') - \tilde{V}_{\mathrm{LB}}(\chi') - \varepsilon\gamma^{-(t+1)}$ in the subsequent stage (where the belief is described by a characteristic vector $\chi' = \tau(\chi, a_1, \pi_2, o)$) multiplied by the probability $\mathbb{P}_{b,\pi_1,\pi_2}[a_1^*, o^*]$ of witnessing (a_1^*, o^*) is maximized. If this excess approximation error is positive, we recurse to the characteristic vector χ' (line 12).

Before and after the recursion the bounds $\tilde{V}_{\mathrm{LB}}(\chi)$ and $\tilde{V}_{\mathrm{UB}}(\chi)$ are improved using the solution of $\tilde{H}\tilde{V}_{\mathrm{LB}}(\chi)$ and $\tilde{H}\tilde{V}_{\mathrm{UB}}(\chi)$ (lines 8 and 13). The update of $\tilde{V}_{\mathrm{UB}}$ is straightforward and a new point $(\chi, \tilde{H}\tilde{V}_{\mathrm{UB}}(\chi))$ is added to the set Υ used to represent upper bound $\tilde{V}_{\mathrm{UB}}$. To obtain a new linear function to add to the set Γ that represents $\tilde{V}_{\mathrm{LB}}$, we use the objective function $o(\chi) = \chi^T \cdot \mathbf{a} + z$ (after fixing variables $\mathbf{a}$ and z) of the dual linear program to (4.4) that forms a lower bound on $\tilde{H}\tilde{V}_{\mathrm{LB}}$ and $\tilde{V}^*$ (see the proof of Theorem 4.4 for more discussion). We provide more details on the representation of value functions $\tilde{V}_{\mathrm{LB}}$ and $\tilde{V}_{\mathrm{UB}}$ in Section 4.4.2.

4.3.5 Extracting Strategy of the Player 1

In Section 4.3, we have presented linear programs to solve stage game $\tilde{H}\tilde{V}(\chi)$ from the perspective of the player 2. These linear programs solved a minimax problem of player 2 by explicitly reasoning about her strategies. The actions of the player 1, however, were only represented as best-response constraints which does not provide the strategy of player 1 directly. In this section, we discuss the dual formulation of the linear program (4.4) and show the way the strategy of the player 1 can be extracted from dual variables. In Section 4.5.2, we use this representation of strategy of player 1 to devise an exact oracle to compute missing actions of the attacker.

Consider constraints (3.38b), (4.4f), (4.4g) and (4.4h) and let us establish the following mapping to dual variables.

Constraints (3.38b)	Dual variables $\pi_1(a_1) \geq 0$
Constraints (4.4h)	Dual variables $\beta^{a_1 o}(\alpha_i) \geq 0$
Constraints (4.4f)	Dual variables $\hat{\mathbf{a}}^{a_1 o} \in \mathbb{R}^k$
Constraints (4.4g)	Dual variables $\hat{z}^{a_1 o} \in \mathbb{R}$

The dual formulation of (4.4) then contains the following constraints (corresponding to variables V, $\hat{V}(a_1, o)$, $\hat{\chi}^{a_1 o}$ and $q^{a_1 o}$ in the primal):

$$\sum_{a_1} \pi_1(a_1) = 1 \tag{4.7a}$$

$$\sum_{\alpha_i} \beta^{a_1 o}(\alpha_i) = \gamma \pi_1(a_1) \qquad \forall a_1, o \tag{4.7b}$$

$$\hat{\mathbf{a}}_j^{a_1 o} = \sum_{\alpha_i} \mathbf{a}_j^{(i)} \beta^{a_1 o}(\alpha_i) \qquad \forall a_1, o, 1 \leq j \leq k \tag{4.7c}$$

$$\hat{z}^{a_1 o} = \sum_{\alpha_i} z^{(i)} \beta^{a_1 o}(\alpha_i) \qquad \forall a_1, o \ . \tag{4.7d}$$

Constraints (4.7) give the following interpretation to variables π_1, β^{a_1o}, $\hat{\mathbf{a}}^{a_1o}$ and $\hat{z}^{a_1o}$. Variables π_1 directly represent the strategy of player 1 for the current stage, $\pi_1 \in \Delta(A_1)$. Variables $\beta^{a_1o}(\alpha_i)$ are the probabilities of playing strategy the value of which is represented by the linear function $\alpha_i(\chi) = (\mathbf{a}^{(i)})^T\chi + z^{(i)}$ in a subgame where player 1 played action a_1 and received observation o. These probabilities are represented in the form of realization plans, i.e., they sum to $\gamma\pi_1(a_1)$ instead of one. We can, however, obtain the true probabilities by dividing β^{a_1o} by $\gamma\pi_1(a_1)$ where applicable. Variables $\hat{\mathbf{a}}^{a_1o}$ and $\hat{z}^{a_1o}$ then define a linear function $\hat{\zeta}^{a_1o}(\chi) = (\hat{\mathbf{a}}^{a_1o})^T\chi + \hat{z}^{a_1o}$ representing the value of playing the given mix of strategies in the subgame and are used to express the value of the game in the rest of the dual formulation.

By normalizing variables β^{a_1o} we obtain a linear function $\zeta^{a_1o} : \mathbb{R}^k \to \mathbb{R}$, called *gadget*, which represent the value in the subgame *given* that subgame has been reached,

$$\zeta^{a_1o}(\chi) = (\mathbf{a}^{a_1o})^T\chi + z^{a_1o} \qquad \text{where} \qquad \begin{aligned} \mathbf{a}^{a_1o} &= \sum_{\alpha_i} (\beta^{a_1o}(\alpha_i)/\gamma\pi_1(a_1)) \cdot \mathbf{a}^{(i)} \\ z^{a_1o} &= \sum_{\alpha_i} (\beta^{a_1o}(\alpha_i)/\gamma\pi_1(a_1)) \cdot z^{(i)} \quad . \end{aligned} \tag{4.8}$$

Note that the gadgets ζ^{a_1o} are only defined for subgames that can be reached, i.e., where $\pi_1(a_1) > 0$.

The representation of a strategy of player 1 using variables π_1 and ζ can then be used to play the game following the similar ideas that are used in continual resolving in the domain of extensive-form games [Burch et al., 2014; Moravčík et al., 2017] and is analogous to the way strategy of the imperfectly informed player is constructed in OS-POSGs (see Section 3.7.2).

Algorithm 4.2: Playing a compact OS-POSG using $\tilde{V}_{\text{LB}}$.

1 $\chi \leftarrow \chi^0$, $\underline{\zeta} \leftarrow$ initial gadget $\underline{\zeta}(\chi) = -\infty$
2 **while** player 1 is required to act **do**
3 Solve $\tilde{H}\tilde{V}_{\text{LB}}(\chi)$ with additional constraint $o(\chi) \geq \underline{\zeta}(\chi)$ in the dual formulation and extract strategy π_1, ζ of player 1
4 Sample action $a_1 \sim \pi_1$, execute it and observe an observation o
5 $\chi \leftarrow \hat{\chi}^{a_1o}/q^{a_1o}$, $\underline{\zeta} \leftarrow \zeta^{a_1o}$

Algorithm 4.2 describes the algorithmic scheme to play a game in compact representation. Whenever player 1 is required to act, he solves a stage game $\tilde{H}\tilde{V}_{\text{LB}}(\chi)$. In the first stage, he solves the game without any additional constraints (since initially $\underline{\zeta}(\chi) = -\infty$) to find the near-optimal strategy for the entire game. In the subsequent stages, player 1 has to ensure that the strategy he is about to play is consistent with the strategy assumed in the first stage, i.e., it provides (at least) the value represented by the gadget ζ^{a_1o} from the previous stage. To this end, a constraint $o(\chi) \geq \underline{\zeta}$ is added to the dual formulation of (4.4) (recall that $o(\chi)$ is the objective of the dual linear program).

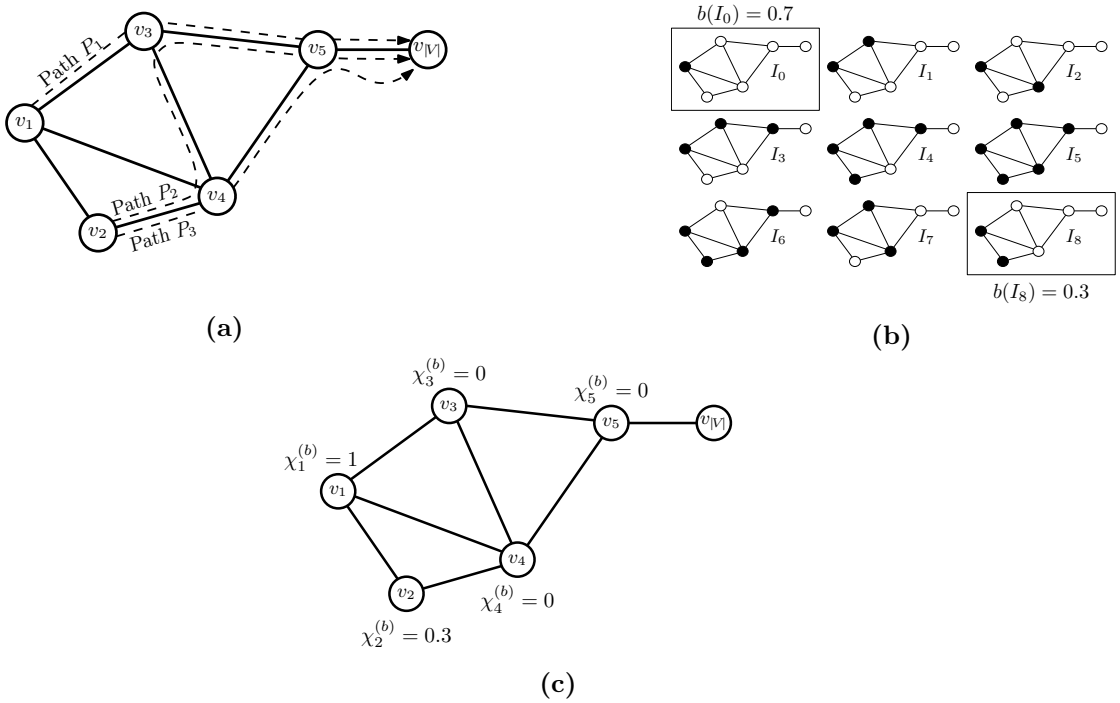


Figure 4.1: Example of a lateral movement POSG: (a) Example of a network with 6 vertices. Attacker starts in vertex v_1 and is trying to reach vertex $v_{|V|}$ while minimizing the cost. (b) Subset of states of the game (in total, 14 non-terminal states are reachable). Initial state of the game is I_0 (only v_1 is infected). Further states are reached by infecting adjacent vertices to an already infected vertex. OS-POSGs define the belief as a probability distribution over states—b is an example of such belief. (c) Our approach *summarizes* the belief b using a characteristic vector $\chi^{(b)}$. In this case, $\chi_i^{(b)}$ represents the probability that vertex v_i is infected when belief b from Figure 4.1b is considered (here, v_2 is infected in the states of infection I_4, I_5, I_6 and I_8, i.e., $\chi_2^{(b)} = b(I_4) + b(I_5) + b(I_6) + b(I_8) = 0.3$).

4.4 The Lateral Movement POSG

We now formally define the lateral movement POSG and describe the key steps of the algorithm specifically for this domain. The game is played on a directed acyclic graph $G = (V, E)$ representing a computer network. We assume that $V = \{v_1, \ldots, v_{|V|}\}$. The goal of the attacker is to reach vertex $v_{|V|}$ from the initial source of the infection v_1 by traversing the edges of the graph, while minimizing the cost to do so, see Figure 4.1a.

Initially, the attacker controls only the vertex v_1, i.e., the initial state of *infection* is $I_0 = \{v_1\}$. Then, in each stage of the game, the attacker chooses a directed path $P = \{P(i)\}_{i=1}^{k}$ (where k is the length of the path) from any of the infected vertices to the target vertex $v_{|V|}$. Figure 4.1a provides examples of paths the attacker may consider, however, he can only use paths P_2 and P_3 if vertex v_2 has been previously infected.

Unless the defender takes countermeasures, the attacker infects all the vertices on the path, including the target vertex $v_{|V|}$, and pays the cost of traversing each of the edges

on the path,

$$c_{-,P} = \sum_{P(i) \in P} C(P(i)) \ , \tag{4.9}$$

where $C(P(i))$ is a cost associated with taking edge $P(i)$.

The defender tries to discover the attacker and increase his cost to reach the target (and thus possibly discourage the attack) by deploying honeypots into the network. In each stage, the defender can decide a subset $H \in E^{[N_H]}$ of edges E (where $E^{[N_H]}$ denotes all subsets of E with cardinality N_H) to be honeypots. A honeypot is then deployed on every edge $h \in H$ and is able to detect if the attacker traverses that specific edge. Furthermore, it also increases the cost for using the edge h to $\overline{C}(h)$. If the attacker observes that he has traversed any of the honeypot edges, he may decide to change his plan and therefore he does not execute the rest of his originally intended path P. The cost of playing a path P against a honeypot placement H is therefore

$$c_{H,P} = \sum_{P(i) \in P_{\leq H}} C(P(i)) + \sum_{h \in H} \mathbf{1}_{h \in P_{\leq H}} \cdot [\overline{C}(h) - C(h)] \tag{4.10}$$

where $P_{\leq H}$ is the prefix of the path P until the interaction with any honeypot edge $h \in H$,

$$P_{\leq H} = \begin{cases} P & P \cap H = \varnothing \\ \{P(i)\}_{i=1}^{\min\{j | P(j) \in H\}} & \text{otherwise} \ . \end{cases} \tag{4.11}$$

For the reasons of notational simplicity, we write $P_{\leq h}$ instead of $P_{\leq \{h\}}$ when it is clear that h is a singleton.

Since the attacker does not need to continue to execute his selected path P (since the defender can update his belief over the possible subsets of infected nodes and thus reconfigure the honeypots), the new state of infection $I_{H,P}$ (i.e., the subset of vertices that are infected at the beginning of the next stage) becomes

$$I_{H,P} = I \cup \{v \,|\, (u, v) \in P_{\leq H}\} \ . \tag{4.12}$$

We assume the worst case scenario from the perspective of the defender where the attacker is assumed to be able to infer the position of *all* honeypots upon interacting with any of them.

The above problem can be formalized as a one-sided partially observable stochastic game:

- states are possible subsets of infected vertices (or *infections*), i.e., $S \subseteq 2^V$ (Figure 4.1b shows an example of a state space where filled dots represent infected vertices),
- actions of the defender (player 1) are honeypot allocations, i.e., $A_1 = E^{[N_H]}$ where $E^{[N_H]}$ denotes N_H-element subsets of E,
- actions A_2 of the attacker (player 2) are directed paths in G reaching $v_{|V|}$,
- observations denote whether and where the defender detected the attacker (observations $\det(h)$) or the attacker reached the target $v_{|V|}$ undetected (observation $\neg\det$),

- transitions follow the Equation (4.12) and an observation $\det(h)$ is generated if and only if the honeypot edge h is the first honeypot edge traversed on the path chosen by the attacker,
- reward of the defender is the cost of the attacker, i.e., $R(H, P) = c_{H,P}$,
- discount factor of the game is $\gamma = 1$, and
- initial belief b^{init} of the game satisfies $b^{\text{init}}(I_0) = 1$.

Note that the original HSVI algorithm for one-sided POSGs has been defined and proved for discounted problems with $\gamma < 1$. In this particular case, however, we expect that the convergence properties translate even to this undiscounted case since the game is essentially finite (in a finite number of steps, *all* vertices, including $v_{|V|}$, get infected and the game ends).

4.4.1 Characteristic Vectors

The number of states in the game is exponential in the number of vertices of the graph, up to $|S| = 2^{|V|-2} + 1$ (we consider that v_1 is always infected and we treat all states where $v_{|V|}$ is infected as a single terminal state of the game). We use marginal probabilities of a vertex being infected as characteristic vectors $\chi \in \mathbb{R}^{|V|}$, i.e.

$$\chi_i^{(b)} = \sum_{I \in S \mid v_i \in I} b(I) \ . \tag{4.13}$$

Consider the example of a belief from Figure 4.1b. The belief in the original OS-POSG model is a probability distribution over states, i.e., $|S|$-dimensional vector with $|S| - 1$ degrees of freedom. In this game, the number of non-terminal reachable states (i.e., states where $v_{|V|}$ is not yet infected) is 14. In comparison, the characteristic vector (see Figure 4.1c) has only $|V|$ dimensions. Moreover, observe that in this case the set $\{b' \mid \mathbf{A}b' = \chi^{(b)}\}$ of possible beliefs that get projected to $\chi^{(b)}$ is a singleton and contains only the belief b from Figure 4.1b.

4.4.2 Value Function Representation

The algorithm from Section 4.3.4 approximates $\tilde{V}^*$ using a pair of value functions, the lower bound $\tilde{V}_{\text{LB}}$ and the upper bound $\tilde{V}_{\text{UB}}$. Similarly to one-sided POSGs, we use a pointwise maximum over a finite set $\Gamma = \{\alpha_1, \ldots, \alpha_k\}$ of affine functions $\alpha_i(\chi) = (\mathbf{a}^{(i)})^T \chi + z^{(i)}$ to represent $\tilde{V}_{\text{LB}}$, i.e.,

$$\tilde{V}_{\text{LB}}(\chi) = \max_{\alpha \in \Gamma} \alpha(\chi) \ . \tag{4.14}$$

The representation of the upper bound $\tilde{V}_{\text{UB}}$ is, however, more challenging.

In the original algorithm, the value function V_{UB}^{Υ} is defined (see Equation (3.42)) over the probability simplex $\Delta(S)$. In that case, it suffices to consider $|S|$ points in Υ to define V_{UB}^{Υ} for every belief. In contrary, the space of characteristic vectors (i.e., marginal probabilities) is formed by a hypercube $[0, 1]^{|V|}$ with $2^{|V|}$ vertices, which would make the straightforward point representation impractical.

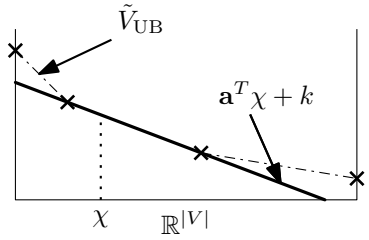


Figure 4.2: Dual interpretation of the projection on the convex hull.

We can, however, leverage the fact that in this domain infecting an additional node can only decrease the cost to the target (and hence $\tilde{V}^*$ is decreasing). Consider the dual formulation of the optimization problem (3.42). In this formulation, the projection of χ to the lower convex hull of a set of points is represented by the *optimal* linear function $\mathbf{a}^T\chi + z$ defining a facet of the convex hull (see Figure 4.2). Since $\tilde{V}^*$ is decreasing in χ, we can also enforce that $\mathbf{a}^T\chi + z$ is decreasing in χ (i.e., add the constraint $\mathbf{a} \leq 0$ to the dual formulation). This additional constraint translates to a change of the equality $\sum_{1\leq i\leq|\Upsilon|} \lambda_i \chi^{(i)} = \chi$ in the primal problem to an inequality.

$$\tilde{V}_{\mathrm{UB}}(b) = \min_{\lambda\in\mathbb{R}^{|\Upsilon|}_{\geq 0}} \left\{ \sum_{1\leq i\leq|\Upsilon|} \lambda_i y^{(i)} \mid \mathbf{1}^T\lambda = 1, \sum_{1\leq i\leq|\Upsilon|} \lambda_i \chi^{(i)} \leq \chi \right\} \tag{4.15}$$

Now it is sufficient that the set Υ contains just one point $(\chi^{(i)}, y^{(i)})$ where $\chi^{(i)} = \mathbf{0}^{|V|}$ (instead of $2^{|V|}$ points) to make the constraint $\sum_{1\leq i\leq|\Upsilon|} \lambda_i \chi^{(i)} \leq \chi$ satisfiable.

It is possible to adapt the constraint (4.4h) to use the representation from (4.15) using similar ideas used to derive (3.50)—and thus obtain a linear program for solving $\tilde{H}\tilde{V}_{\mathrm{UB}}(\chi)$.

4.4.3 Using Marginalized Strategies in Stage Games

The linear program formed by modifications from Equations (4.4) still requires solving the stage game for the original, unabstracted problem. In this section, we show that it is possible to avoid expressing the belief b explicitly, and to compute the stage game directly using the characteristic vectors and marginalized strategies of the attacker.

First, we present the representation of the stage-game strategies of the attacker. Instead of representing joint probabilities $\pi_2(I \wedge P)$ of choosing path P in state I, we only model the probability $\tilde{\pi}_2(P)$ of choosing path P aggregated over all states $I \in S$. Furthermore, we allow the attacker to choose the probability $\xi(P \wedge v_i)$ that vertex v_i is infected while he opts to follow path P.

$$\sum_P \tilde{\pi}_2(P) = 1 \tag{4.16a}$$

$$0 \leq \xi(P \wedge v_i) \leq \tilde{\pi}_2(P) \qquad \forall P, v_i \tag{4.16b}$$

$$\tilde{\pi}_2(P) \geq 0 \qquad \forall P \tag{4.16c}$$

To ensure that the strategy represented by variables $\tilde{\pi}_2$ and ξ is feasible it must be consistent with the characteristic vector χ, where χ_i is the probability that the vertex v_i is infected at the beginning of the stage.

$$\sum_P \xi(P \wedge v_i) = \chi_i \qquad \forall v_i \tag{4.16d}$$

Furthermore, the path P must start in an already infected vertex (denoted as $\mathrm{Pre}(P)$), i.e., the conditional probability $\mathbb{P}[\mathrm{Pre}(P) \in I \,|\, P]$ of $\mathrm{Pre}(P)$ being infected when path P is chosen has to be 1. Now, since $\xi(P \wedge v)$ is the joint probability, $\xi(P \wedge v) = \mathbb{P}[\mathrm{Pre}(P) \in I \,|\, P] \cdot \tilde{\pi}_2(P)$, we get that

$$\xi(P \wedge v) = \tilde{\pi}_2(P) \qquad \forall P, v = \mathrm{Pre}(P)\ . \tag{4.16e}$$

Example 4.1. Consider the example from Figure 4.1a and assume that the attacker wanted to play a strategy π_2 in the original game, where

$$\begin{aligned} &\pi_2(I_0 \wedge P_1) = 0.7 \quad \pi_2(I_8 \wedge P_1) = 0.1 \\ &\pi_2(I_8 \wedge P_2) = 0.1 \quad \pi_2(I_8 \wedge P_3) = 0.1\ . \end{aligned}$$

The same strategy can be described using the marginalized representation $\tilde{\pi}_2$ and ξ:

$$\begin{aligned} &\tilde{\pi}_2(P_1) = \pi_2(I_0 \wedge P_1) + \pi_2(I_8 \wedge P_1) = 0.8 \\ &\tilde{\pi}_2(P_2) = \pi_2(I_8 \wedge P_2) = 0.1 \\ &\tilde{\pi}_2(P_3) = \pi_2(I_8 \wedge P_3) = 0.1 \\ &\xi(P_1 \wedge v_1) = 0.8 \qquad \xi(P_1 \wedge v_2) = 0.1 \\ &\xi(P_2 \wedge v_1) = 0.1 \qquad \xi(P_2 \wedge v_2) = 0.1 \\ &\xi(P_3 \wedge v_1) = 0.1 \qquad \xi(P_3 \wedge v_2) = 0.1 \\ &\text{all other variables } \tilde{\pi}_2 \text{ and } \xi \text{ are zero} \end{aligned}$$

It is straightforward to verify that variables $\tilde{\pi}_2$ and ξ satisfy Equations (4.16a)–(4.16e) when the characteristic vector $\chi^{(b)}$ from Figure 4.1c is considered.

The representation of strategies of the attacker using variables $\tilde{\pi}_2$ and ξ is sufficient to express the expected immediate reward of the strategy $\tilde{\pi}_2$, hence the constraint (3.38b) can be changed to use the marginalized strategies,

$$V \geq \sum_P \tilde{\pi}_2(P) c_{H,P} + \sum_{h \in H} \hat{V}(H, \det(h)) \qquad \forall H \in E^{[N_H]}\ . \tag{4.16f}$$

Importantly, we can also skip the computation of the belief $b^{H,\det(h)}$ and compute the characteristic vector formed by the marginals $\chi^{H,\det(h)}$ directly from the variables $\tilde{\pi}_2$

and ξ. We now present the equation to compute the updated marginal $\chi^{H,\det(h)}$ given that the attacker has been detected while traversing the honeypot edge h (and this has been the first honeypot edge he traversed). Denote the set of all paths satisfying this condition $\mathcal{P}_H^h = \{P \mid h \in P \wedge P_{\leq h} \subseteq P_{\leq H}\}$.

$$\hat{\chi}_i^{H,\det(h)} = \sum_{P \in \mathcal{P}_H^h \mid P_{\leq(\cdot,v_i)} \subseteq P_{\leq h}} \tilde{\pi}_2(P) \qquad + \sum_{P \in \mathcal{P}_H^h \mid P_{\leq(\cdot,v_i)} \not\subseteq P_{\leq h}} \xi(P \wedge v_i) \tag{4.16g}$$

The first sum stands for the probability that the attacker is detected while traversing edge h, but he managed to infect v_i on his chosen path before the detection. The second sum represents the probability that the attacker was detected on the edge h as well, but this time he has not infected v_i using path P, however, the vertex v_i has already been infected before he started to execute path P.

Analogously, we can obtain the probability $q^{H,\det(h)}$ that the attacker got detected while traversing edge h as

$$q^{H,\det(h)} = \sum_{P \in \mathcal{P}_H^h} \tilde{\pi}_2(P) \ . \tag{4.16h}$$

We need not consider the subsequent stages where the attacker has not been detected (i.e., $\neg$det observation has been generated) or the honeypot edge h reaches the target vertex $v_{|V|}$. In each of these cases, the target vertex has been reached and thus the value of the subsequent stage is zero.

Example 4.2. Consider once again the example from Figure 4.1 and the strategy of the attacker π_2 from Example 4.1 and assume that a honeypot is deployed on the edge $h = (v_3, v_5)$ only (i.e., $H = \{h\}$) and the attacker has been detected there. In such case, the attacker could have either used path P_1 or P_2, but not P_3 as it does not reach h.

If the attacker used P_1 in I_0 in his strategy π_2, he infects v_3 and v_5 up to the point of the detection and thus reaches a new state of infection I_3 (this might have happened with probability 0.7). Similarly, using P_1 in I_8 reaches I_4 (with probability 0.1) and using P_2 in I_8 reaches I_5 (with probability 0.1). The (denormalized) belief $\tau(b, H, \pi_2, \det(h))$ (corresponding to Equation (3.38d)) is thus

$$\tau(b, H, \pi_2, \det(h))(I_3) = 0.7 \quad \tau(b, H, \pi_2, \det(h))(I_4) = \tau(b, H, \pi_2, \det(h))(I_5) = 0.1 \ .$$

Computing marginal probabilities that each of the vertices is infected in the belief $\tau(b, H, \pi_2, \det(h))$ yields

$$\hat{\chi}_1^{H,\det(h)} = \hat{\chi}_3^{H,\det(h)} = \hat{\chi}_5^{H,\det(h)} = 1$$
$$\hat{\chi}_2^{H,\det(h)} = 0.2 \qquad \hat{\chi}_4^{H,\det(h)} = 0.1 \ .$$

We can obtain exactly the same marginal probabilities when using marginalized strategy $\tilde{\pi}_2$ and ξ from Example 4.1. For example, neither P_1 nor P_2 infects vertex v_2 (i.e., no edge $(\cdot, v_2)$ is present). Therefore, the first sum in Equation (4.16g) is empty and

$\hat{\chi}_2^{H,\det(h)} = \xi(P_1 \wedge v_2) + \xi(P_2 \wedge v_2) = 0.2$. Similarly, only P_2 infects v_4 before reaching edge h, hence is contained in the first sum in (4.16g), and $\hat{\chi}_4^{H,\det(h)} = \tilde{\pi}_2(P_2) + \xi(P_1 \wedge v_4) = 0.1$.

4.4.4 Initializing Bounds

We now describe our approach to initialize bounds $\tilde{V}_{\mathrm{LB}}$ and $\tilde{V}_{\mathrm{UB}}$ (line 1 of Algorithm 4.1). We start with the former. To obtain a lower bound $\tilde{V}_{\mathrm{LB}}$, we assume that no honeypots can be deployed in the network. The attacker then only needs to find the cheapest (i.e., shortest) path in the graph when costs C are considered. Denote $C^*(v_i)$ the cost of the cheapest path from vertex v_i to the target $v_{|V|}$. We initialize the lower bound $\tilde{V}_{\mathrm{LB}}$ using two linear functions. Firstly, the cost cannot be negative and we use a linear function $\alpha_1(\chi) = 0$. Next, we consider linear function $\alpha_2(\chi) = (\mathbf{a}^{(2)})^T\chi + z^{(2)}$ where $z^{(2)} = C^*(v_1)$ and $\mathbf{a}_i^{(2)} = \min\{C^*(v_i) - C^*(v_1), 0\}$ which is a tighter lower bound for some χ (especially close to the initial characteristic vector χ^{init} where $\chi_1^{\mathrm{init}} = 1$ and other vertices are not infected, i.e., $\chi_i^{\mathrm{init}} = 0$ for $i \geq 2$).

To initialize the upper bound $\tilde{V}_{\mathrm{UB}}$, we consider a perfect information variant of the game (similarly to Section 3.6.1). However, since the number of states (i.e., possible subsets of infected vertices) in such a game can be exponential, we consider a simpler version of the game where the attacker is only in charge of the vertex he visited the last (instead of a subset of all vertices visited in the past) which can only increase the cost for the attacker. Denote $\overline{C}^*(v_i)$ the expected cost of the attacker in the perfect information variant of the game where the attacker controls vertex v_i only. We then use the set $\Upsilon = \{(\chi^{(j)}, y^{(j)}) \,|\, 1 \leq j \leq |V|\}$ of points to initialize $\tilde{V}_{\mathrm{UB}}$ where

$$\chi_i^{(j)} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \qquad y^{(j)} = \overline{C}^*(v_j)\ . \tag{4.17}$$

4.5 Incremental Strategy Generation

In the previous section, we introduced marginalized beliefs and strategies to deal with the large number of game states. However, the number of actions grows fast as well which makes the game computationally challenging and may cause memory consumption issues in practical implementation of the approach. In this section, we address this issue by adopting a double-oracle approach to generate actions of the players incrementally.

4.5.1 Double-Oracle Scheme for Solving Stage Games

Before describing the oracle algorithms used in the experiments, we provide a generic scheme for using double-oracle approach to solving stage games $\tilde{HV}(\chi)$, see Algorithm 4.3. The algorithm keeps global sets of actions $\tilde{A}_1$ and $\tilde{A}_2$ defining a restricted game where only these actions are considered (instead of entire sets of actions A_1 and A_2). Set A_1 is initialized to a single arbitrary honeypot allocation, set A_2 is initialized to an arbitrary set of $|V|$ paths $P_1, \ldots, P_{|V|}$ from each of the vertices v_i to $v_{|V|}$. When solving a stage game,

we first query the oracle of the defender before querying the oracle of the attacker. This decision is motivated by the possibility to use a heuristic oracle for generating actions of the defender (see Section 4.5.4) which is less computationally demanding compared to the exact computation of actions of the attacker.

Algorithm 4.3: Generic double-oracle scheme for solving stage games.

```
1  Ã_1, Ã_2 ← initial subsets of actions of player 1 and player 2 defining a restricted
     game
2  When solving a stage game H̃Ṽ(χ):
3  do
4      Solve stage game H̃Ṽ(χ) considering actions Ã_1 and Ã_2 only
5      if improving allocation H for the defender is found then
6          Ã_1 ← Ã_1 ∪ {H}
7          continue
8      else if improving path P for the attacker is found then
9          Ã_2 ← Ã_2 ∪ {P}
10         continue
11     break
12 while an improving action is found for one of the players
```

4.5.2 Exact Oracle of the Attacker

In Section 4.3.5, we have described the way the strategy of player 1 can be extracted in the form of variables π_1 and gadgets $\zeta^{H,\det(h)}(\chi) = (\mathbf{a}^{H,\det(h)})^T\chi + z^{H,\det(h)}$. The oracle of the attacker inspects this strategy and its lower bound $o(\chi)$ (extracted from the objective of the dual formulation of the stage game) and tries to *prove* or *disprove* that the value $o(\chi)$ is a valid lower bound on the solution.

To this end, the attacker selects a *single* path from any of the vertices of the graph towards the target vertex $v_{|V|}$ and selects a characteristic vector χ where this path of the attacker is applicable.

$$\delta(v_j) + \sum_{(v_i,v_j)\in E} P_{ij} = \sum_{(v_j,v_k)\in E} P_{jk} \qquad \forall v_j \neq v_{|V|} \tag{4.18a}$$

$$\sum_{v_i \neq v_{|V|}} \delta(v_i) = 1 \tag{4.18b}$$

$$P_{ij} \in \{0,1\} \qquad \forall (v_i,v_j) \in E \tag{4.18c}$$

$$\delta(v_i) \geq 0 \qquad v_i \neq v_{|V|} \tag{4.18d}$$

(4.18e)

The single path of the attacker is represented as an integral flow in the graph of capacity 1. Constraint (4.18a) represents the Kirchoff's law which has to hold for any but two

vertices—the source vertex of the path which is indicated by $\delta(v_j) = 1$, and the target vertex $v_{|V|}$. Furthermore, constraint (4.18b) ensures that only one path is considered.

The characteristic vector χ must make the path represented by the flow P applicable, i.e. $\chi_i = 1$ for the source vertex v_i where $\delta(v_i) = 1$.

$$\chi_i \geq \delta(v_i) \qquad \forall v_i \neq v_{|V|} \tag{4.18f}$$

$$0 \leq \chi_i \leq 1 \qquad \forall v_i \in V \tag{4.18g}$$

We now express the cost of playing a path represented by the flow P against the honeypot allocation H. To this end, we first define an auxiliary flow P^H which is identical to P, except that it gets blocked by the first honeypot edge $(v_i, v_j) \in H$.

$$\delta(v_j) + \sum_{(v_i,v_j)\in E\setminus H} P^H_{ij} = \sum_{(v_j,v_k)\in E} P^H_{jk} \qquad \forall v_j \neq v_{|V|} \tag{4.18h}$$

$$P^H_{ij} \leq P_{ij} \qquad \forall (v_i, v_j) \in E \tag{4.18i}$$

$$0 \leq P^H_{ij} \leq 1 \qquad \forall (v_i, v_j) \in E \tag{4.18j}$$

Note that the fact that flow P^H is identical to P (except for the blocking on honeypot edges H) is ensured by the constraint (4.18i)—if the flow has to continue from any vertex, it must follow the edge contained in P. Now, we can express the immediate cost of playing P against honeypot allocation H as

$$c_{H,P} = \sum_{(v_i,v_j)\in E\setminus H} C(v_i, v_j) P^H_{ij} + \sum_{(v_i,v_j)\in H} \overline{C}(v_i, v_j) P^H_{ij} \quad . \tag{4.18k}$$

Next, we express the characteristic vector χ^H in the subgame given that the attacker got detected on *any* edge from H (here, $\sum_{(v_i,v_j)\in E} P^H_{ij}$ is a binary value indicating whether the attacker reached v_j by the time he got detected).

$$\chi^H_j = \sum_{(v_i,v_j)\in E} P^H_{ij} + \chi_j \left(1 - \sum_{(v_i,v_j)\in E} P^H_{ij}\right) \tag{4.18l}$$

This allows us to derive the expected cost in the subgame given that the attacker got detected on a honeypot edge $h = (v_x, v_y) \in H$.

$$V^{H,\det(h)} = \begin{cases} 0 & v_y = v_{|V|} \\ \sum_{v_i\in V} \mathbf{a}_i^{H,\det(h)} \chi^H_i P^H_{xy} + z^{H,\det(h)} P^H_{xy} & \text{otherwise} \end{cases} \tag{4.18m}$$

Note that $V^{H,\det(h)}$ is zero if the attacker does not reach edge h (in that case $P^H_{xy} = 0$) or the attacker reached his target $v_{|V|}$. This gives us the expected cost V^H of the path represented by the flow P against honeypot allocation H,

$$V^H = c_{H,P} + \sum_{h\in H} V^{H,\det(h)} \ , \tag{4.18n}$$

and the expected cost of using path P is then

$$V = \sum_{H} \pi_1(H) V^H \ . \tag{4.18o}$$

The optimization objective of the attacker is to disprove that $o(\chi)$ is the lower bound. To this end, he attempts to find abstracted belief χ and path represented by P where the difference $o(\chi) - V$ is maximized, i.e.,

$$\begin{aligned} &\max\ o(\chi) - V \\ &\text{s.t. constraints (4.18a)–(4.18o)} \ . \end{aligned}$$

If $o(\chi) - V > 0$, the function $o(\chi)$ is provably not a lower bound on the solution of the stage game. Hence, the path of the attacker represented by P can be added into the restricted game.

Note that all products $z = \text{term}_1 \cdot \text{term}_2$ of variable expressions in the above mathematical program only involves expressions $\text{term}_1 \in [0, 1]$ and a binary expression $\text{term}_2 \in \{0, 1\}$. Such products can be rewritten using a set of linear constraints

$$z \leq \text{term}_1 \qquad z \leq \text{term}_2 \qquad z \geq \text{term}_1 - (\text{term}_2 - 1) \qquad z \geq 0 \tag{4.19}$$

and we thus obtain a mixed integer linear program formulation.

4.5.3 Exact Oracle of the Defender

The oracle of the defender is aimed on computing a best response of the defender (i.e., the best possible placement of honeypots on edges) against a fixed strategy of the attacker (described by variables $\tilde{\pi}_2$ and ξ). To this end, we propose a branch-and-bound approach.

Each node of a branch-and-bound search tree corresponds to a partial allocation of the honeypots H. We can obtain a lower bound on the cost of the attacker induced by all solutions $H' \supseteq H$ by assuming that only honeypot edges in H are used,

$$LB(H) = \sum_{P} \tilde{\pi}_2(P) c_{H,P} + \sum_{h \in H} q^{H,\det(h)} \tilde{V}(\hat{\chi}^{H,\det(h)} / q^{H,\det(h)}) \ . \tag{4.20}$$

Note that the division by $q^{H,\det(h)}$ is used to account for the fact that $\hat{\chi}^{H,\det(h)}$ is multiplied by $q^{H,\det(h)} = \mathbb{P}_b[\det(h) \mid H]$ (i.e., by the probability that observation $\det(h)$ is generated when honeypot allocation H is used).

An upper bound on the solutions $H' \supseteq H$ is obtained by neglecting the sequential interaction of remaining honeypots (i.e., the attacker can be possibly caught more than once). By placing a honeypot on an edge $h \notin H$, the cost can be increased by at most

$$\Delta_h = \sum_{P \mid h \in P \wedge P_{\leq h} \subseteq P_{\leq H \cup \{h\}}} \tilde{\pi}_2(P) \cdot (\overline{C}(h) - C(h)) + q^{H \cup \{h\}, \det(h)} \tilde{V}(\chi^{H \cup \{h\}, \det(h)} / q^{H \cup \{h\}, \det(h)}) \ . \tag{4.21}$$

Given that the attacker reaches edge h, he pays $\overline{C}(h) - C(h)$ extra cost and furthermore he enters a subgame where he got detected on edge h from a honeypot set $H \cup \{h\}$. We can then obtain the upper bound using

$$UB(H) = LB(H) + \sum_{i=1}^{N_H - |H|} \Delta_i^* \ , \tag{4.22}$$

where $\Delta_1^*, \ldots, \Delta_{N_H-|H|}^*$ are the $N_H - |H|$ highest values among Δ_h.

4.5.4 Heuristic Oracles

In practice, finding exact solutions of oracles from Sections 4.5.2 and 4.5.3 can be computationally expensive. To mitigate the impact of the oracle computation times on the performance of the algorithm, we present a heuristic version of the oracle of the defender. The heuristic oracle is a greedy variant of the exact oracle presented in Section 4.5.3.

Algorithm 4.4: Heuristic version of the oracle of the defender.

```
1 H ← ∅
2 while |H| < N_H do
3 |   h* ← arg max_{h∈E\H} Δ_h
4 |   H ← H ∪ {h*}
```

The use of the heuristic oracle of the defender from the Algorithm 4.4 can lead to a degradation of the quality of the solution found by the HSVI algorithm. By omitting some actions of the defender from the support of the equilibrium, the cost of the attacker can be lowered. Nevertheless, the lower-bound guarantees on the quality of the solution found by the algorithm (Theorem 4.3) are retained.

4.6 Experimental Evaluation

In this section we focus on the experimental evaluation of the proposed techniques on the lateral movement POSG introduced in Section 4.4. We consider three variants of the HSVI algorithm for compact POSGs depending on the oracle algorithm used: (1) the version where no oracles are used (i.e., $\tilde{A}_i = A_i$), (2) the variant using exact oracles for both of the players (described in Sections 4.5.2 and 4.5.3), (3) and finally the variant where heuristic oracle for the defender (described in Section 4.5.4) is used instead of the exact computation of the best response. We also compare these approaches with the original HSVI algorithm presented in Chapter 3 where the summarized abstraction is not used and the computation is performed based on the entire (exponential) state space.

4.6.1 Experiments Setting

The evaluation has been performed on a set of randomly generated directed acyclic graphs with varying parameters—number of vertices in the network $|V|$, number of honeypots N_H and the density of the graph.

We use the following algorithm to generate mesh-like networks. First, positions of $|V|$ nodes are randomly generated from a 2-dimensional interval $[0, 20] \times [0, 20]$ with v_1 positioned at $(0, 0)$ and $v_{|V|}$ positioned at $(20, 20)$ ($\mathbf{p}_i \in [0, 20]^2$ denotes the position of v_i). For each vertex v_i a set of k nearest neighbors $N_k(v_i)$ (with respect to positions $\mathbf{p}_j$) is determined and an edge connecting vertices v_i and v_j is created for every $v_j \in N_k(v_i)$. The orientation of the edges is set to always lead towards the vertex closer to the target $v_{|V|}$. Note that increasing the parameter k leads to an increase in the number of edges in the graph.

The graph generated by the above method may, however, contain multiple vertices with zero in-degree or out-degree (apart of vertices v_1 and $v_{|V|}$). We fix this by regenerating positions of such nodes until finding a directed acyclic graphs where only v_1 and $v_{|V|}$ have zero in-degree and out-degree, respectively.

Next, we define a cost function $f(x, y) \to \mathbb{R}_{>0}$. This function indicates the hardness to operate in point (x, y). In our experiments, we use $f(x, y) = x + y$ which attains its maximum in $(20, 20)$ where the target vertex $v_{|V|}$ is located (i.e., the network is the most secured around the target vertex). To obtain the cost $C(v_i, v_j)$ of traversing an edge (v_i, v_j) given that the honeypot is not deployed there, we integrate the cost function f along the line connecting points $\mathbf{p}_i$ and $\mathbf{p}_j$,

$$C(v_i, v_j) = \|\mathbf{p}_i - \mathbf{p}_j\|_2 \int_0^1 f(\lambda \mathbf{p}_i + (1 - \lambda)\mathbf{p}_j)\, d\lambda \ . \tag{4.23}$$

Furthermore, we partition the vertices into 3 security perimeters (with multiplicative constants κ_1, κ_2 and κ_3) based on the distance from the position of the target vertex $v_{|V|}$. The vertices closest to the target vertex $v_{|V|}$ are the most secured, hence the attacker has to use a potentially costly zero-day exploit to proceed (and risk its revelation). Thus the cost $\overline{C}(v_i, v_j)$ of traversing an edge entering vertex v_j that is present in the most secured zone is high,

$$\overline{C}(v_i, v_j) = \kappa_3 \cdot C(v_i, v_j) \qquad \text{for} \qquad \kappa_3 = 4 \ . \tag{4.24}$$

For security perimeters further away from the target vertex $v_{|V|}$ we use $\kappa_2 = 1.5$ and $\kappa_1 = 1$. Figure 4.3 provides an example of such a network and its accompanying cost function f.

All computational results have been obtained on computers equipped with Intel Xeon Scalable Gold 6146 processors and 16GB of available RAM while limiting the runtime of the algorithms to 2 hours. CPLEX 12.9 has been used to solve (mixed integer) linear programs. The algorithms were required to find an ε-optimal solution where ε is set to 1% of the error $\tilde{V}_{\mathrm{UB}}(\chi^{\mathrm{init}}) - \tilde{V}_{\mathrm{LB}}(\chi^{\mathrm{init}})$ in the initial characteristic vector χ^{init} after the initialization phase described in Section 4.4.4 is completed. If the algorithm failed

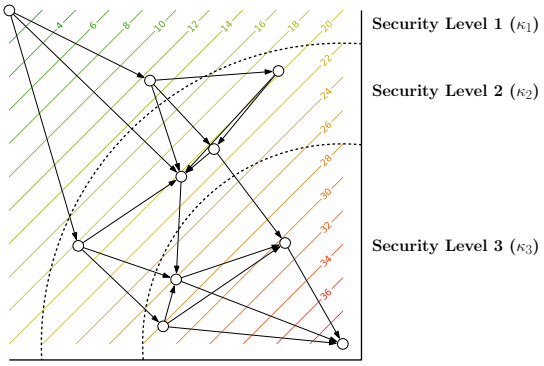


Figure 4.3: Example of a mesh network used in experiments. Contour lines depict the cost function f used to derive the costs C and $\overline{C}$ of the edges. The vertices are partitioned into 3 security levels.

to reach this level of precision within 2 hours, we report an instance as unsolved. The results are based on 100 randomly generated networks for each parameter set.

4.6.2 Scalability in the Size of Graphs

First, we focus on the scalability of the algorithms in size of the network—in the number of vertices $|V|$, and in the density of the graph (controlled by the parameter k).

Figure 4.4 depict the scalability of the algorithms in the number of vertices. First of all, observe that the original HSVI algorithm for one-sided POSGs has been unable to solve all but the smallest instances with 8–12 vertices and even with only $N_H = 1$ honeypots (while exhausting the memory on larger instances). To this end, we did not evaluate this algorithm on more complicated instances with $N_H > 1$. In contrary, the algorithm relying on the technique of summarized abstractions have been able to solve most of the instances with $N_H = 1$ even without the use of the incremental strategy generation technique described in Section 4.5. The versions of the algorithm relying on the incremental strategy generation were then able to solve all 100 randomly generated instances for each parameterization when $N_H = 1$ honeypot is considered. Importantly, the bounds computed by the versions using incremental strategy generation overlap with the bounds computed by the remaining variants. This demonstrates that in spite of improving the runtimes by using the incremental strategy generation technique, the quality of the solution does not deteriorate.

Increasing the number of honeypots to $N_H = 2$ highlights the advantages of using the oracles to generate actions of the players incrementally. While the algorithm that does not use the oracles failed to solve most of the instances with $|V| \geq 18$ in the time limit of 2 hours, both of the variants relying on the incremental strategy generation technique were able to solve even most of the largest instances with 24 vertices and $k = 4$ in less than 20 minutes on average (considering only the instances solved by the algorithms).

Increasing the number of honeypots to $N_H = 3$ highlights the computational advantages of using the heuristic version of the oracle of the defender presented in Section 4.5.4. This version of the algorithm was able to solve significantly more instances compared to

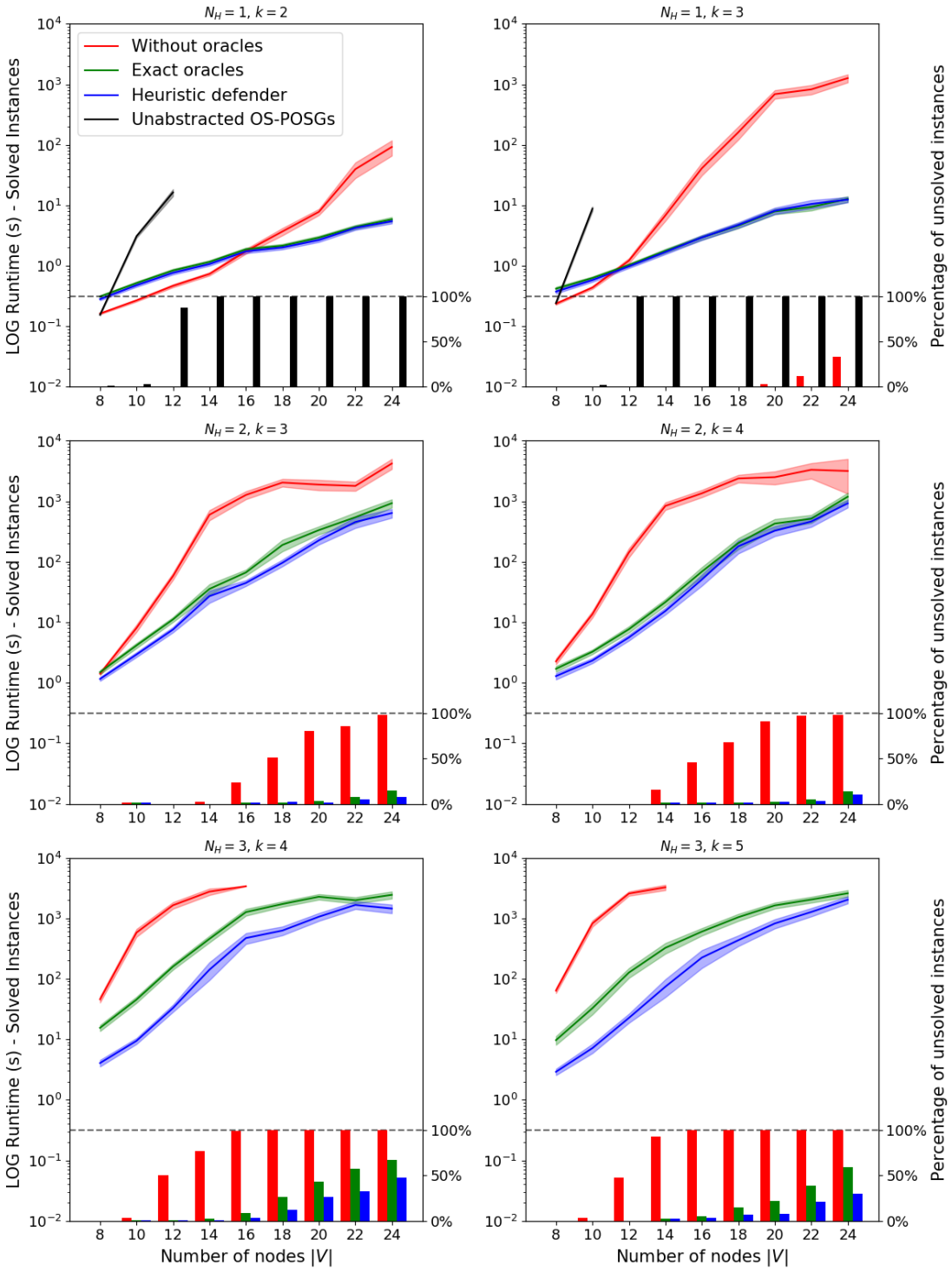


Figure 4.4: Scalability in the number of vertices in the network $|V|$ (averages based on 100 instances for each parameter set). Confidence intervals mark the standard error. The reported runtimes include only instances solved by the algorithms. The percentage of instances where the algorithms failed to terminate within 2 hours are reported separately.

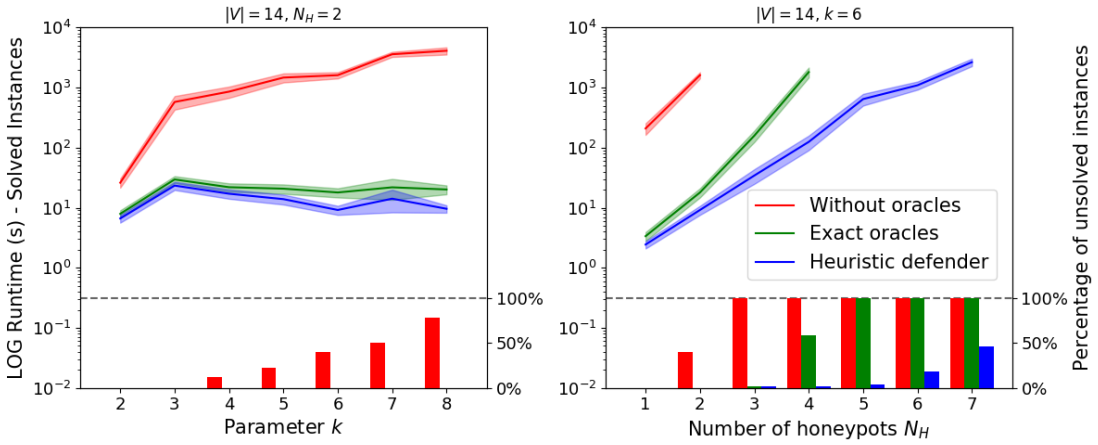


Figure 4.5: Scalability of the algorithms: (Left) in the density (i.e., the number k of nearest neighbors considered when forming the edges) of the network with 14 vertices and $N_H = 2$ honeypots, (Right) in the number of honeypots N_H in a network with 14 vertices and $k = 6$. Confidence intervals mark the standard error. The reported runtimes include only instances solved by the algorithms. The percentage of instances where the algorithms failed to terminate within 2 hours are reported separately.

the version using the exact version of the oracle of the defender (Section 4.5.3). Note that the version that does not use the strategy generation technique was able to solve only a few instances with $|V| \leq 14$. This is mainly attributed to the size of the linear programs to solve the stage games—e.g., the number of possible combinations of actions of players in the games with 14 vertices and $k = 5$ we considered is up to $|A_1| \times |A_2| \approx 5.2 \cdot 10^7$. Note that the algorithms using the oracles were able to solve nearly all instances in this setting ($|V| = 14$ and $k = 5$).

In Figure 4.5 (left), we focus on the scalability of the three variants of the algorithm using the summarized abstraction based on the density of the network (controlled by the parameter k). The number of actions in the game can grow fast with the addition of new edges to the network which makes solving the game challenging for the variant without the use of oracles. However, the addition of the edges need not increase the support of the equilibrium and hence the runtime of the variants relying on incremental strategy generation is not affected by the increasing value of the parameter k.

4.6.3 Scalability in the Number of Honeypots

In Figure 4.5 (right) we focus on the scalability of the algorithm in the number of honeypots. The number of actions of the defender is exponential in the number of honeypots N_H. Hence, it is not surprising that the algorithm that does not use the incremental strategy generation to form defender's actions can solve only the smallest instances in terms of the number of honeypots N_H. The variants of the algorithm relying on the oracles to incrementally generate the actions perform significantly better, however, when the number of honeypots is large ($N_H \geq 4$), computing the best response of the

defender exactly (using the approach from Section 4.5.3) is prohibitively expensive. In contrary, the heuristic variant of the oracle of the defender allows the algorithm to scale better and solve most of the instances up to $N_H = 6$.

Interestingly, we observed that using the heuristic oracle did not impact the quality of the solution in a negative way. Considering the instances from Figure 4.5 (right) that were solved by the variant using the exact oracle of the defender, we observed that the lower bounds $\tilde{V}_{\mathrm{LB}}(\chi^{\mathrm{init}})$ computed by these two variants of the algorithm were within 0.6% of each other.

4.6.4 Applicability to Computer Networks

Previous experiments have been focused on randomly generated mesh networks. The physical properties of mesh networks typically disallow direct communication between the devices (e.g., due to the presence of obstacles, or low signal strength caused by large distances between the devices). This leads to a significantly lower number of edges in the corresponding game graph compared to regular computer network architectures. To demonstrate the versatility of our approach, we evaluate our algorithms on a graph originating from a fundamentally different network topology given as an example by a network security expert.[2]

In Figure 4.6a, the physical topology of the considered computer network is shown. Based on this topology, we derived logical communication links, shown in Figure 4.6b, that the attacker can use to perform lateral movement and that do not directly correspond to the physical links. For example, every device can directly communicate with the web server and thus the attacker can use this communication channel to carry on his attack. Moreover, compromising the web server allows an attack on the database server. We also reflect potential side channels in the networks such as social engineering allowing the attacker to compromise end-user machines by publishing malicious files on the web presentation of the company.

Each logical link can be used to infect an adjacent machine. The links, however, differ in the difficulty of carrying such an attack along the edge, e.g., it is substantially easier to infiltrate web server via an exploitable web presentation than attempting to compromise the domain controller running the latest versions of the services. We distinguish three difficulty levels (with three different associated costs representing the time needed to successfully traverse the edge): *easy* ($C(e) = 10$ units of time), *medium* ($C(e) = 20$) and *hard* ($C(e) = 40$). In the presence of a honeypot, the costs increase to $\overline{C}(e) = 40$, $\overline{C}(e) = 80$ and $\overline{C}(e) = 160$, respectively[3].

In our example, we assume that the attacker uses a computer outside of the network (i.e., in the internet zone) to infiltrate the network, with the aim of compromising the `target` node within the internal part of the network (assume that sensitive data are

[2]We thank Sridhar Venkatesan for providing the expertise on experiments with the business network.

[3]Note that the exact values of costs do not affect usability of our algorithm and hence can be set specifically for each network.

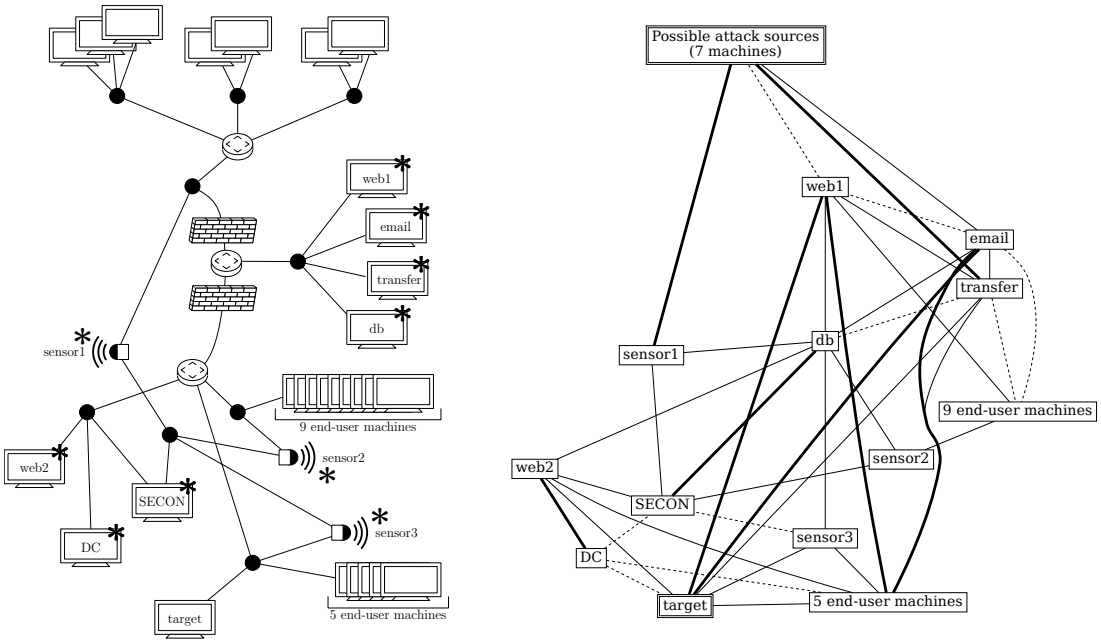


Figure 4.6: Computer network topology considered in the experiments: (a) Physical topology of the network. Nodes marked with asterisk exist in multiple instances and can be used as honeypots. (b) Logical communication links in the network. Dashed lines correspond to *easy* difficulty levels, solid lines to *medium* difficulty and finally bold solid lines represent *hard* difficulty.

located at this host). The defender attempts to slow down attacker's progress by deploying a honeypot on one of the hosts[4] within the network (hosts that can be selected are denoted by an asterisk in Figure 4.6a).

We assume that each such host is present in d instances within the network (e.g., one being the production server and the remaining instances serving for backup purposes). In order to serve the legitimate users, the defender is allowed to operate a honeypot on one of the instances only. For $d = 2$ instances of each server, the algorithm (using the exact versions of the oracles) took 982s to find a solution with the lower bound on attacker's cost of 69.4, while for $d = 3$ instances of each server a solution with quality 63.3 has been found in 5950s. The shortest attack path in the graph has a cost of 50, which marks an increase in the attacker's cost by 38.8% and 26.6%, respectively. The larger of the two instances (with $d = 3$) is represented by a graph with 52 vertices and 504 edges. Note that the decreasing value of the game with increasing value of d is caused by the fact that the defender is able to cover only one of the instances of a server—thus the probability that the attacker manages to choose an unprotected instance increases.

While we have demonstrated the capabilities of the algorithm to solve problems related to computer networks, we believe that its scalability on this type of networks can be significantly improved and we discuss this in the next section.

4.7 Discussion and Future Extensions

The model introduced in Section 4.4 represents the interaction with an attacker who is determined to reach his goal (in our case reach the target vertex $v_{|V|}$). The attacker accepts the risk of getting detected in the course of the attack, and understands the implications of a successful detection. First, the cost of traversing a honeypot edge is higher. More importantly, the defender gets vital information about the progress of the attack which the defender can use to harden further progress of the attacker.

In Section 4.6.4, we presented an application of the proposed algorithm to improve security of computer networks. Real-world applications, however, require that the legitimate services do not get replaced by honeypots to sustain the operation of the network. We reflected this by operating each server in multiple instances (essentially by duplicating vertices and their incident logical communication links in the network) and allowing the defender to use only one of the instances as a honeypot (thus confusing the attacker without compromising the experience of legitimate users).

Such an approach inherently introduces symmetries to the problem—the defender can decide to choose any of the instances of a single service as a honeypot, while the attacker can choose any instance (not knowing which one is a decoy) to carry on the attack. Symmetries, however, have negative impact on the time needed to solve the game as the experiments in Section 4.6.4 show.

[4]If a honeypot is deployed in vertex v_i, all edges adjacent to v_i are honeypots.

We believe that our approach is well suited to deal with such symmetries. Since all instances $v_{i_1}, \ldots, v_{i_d}$ of a service are indistinguishable within the graph, it is possible to replace coordinates $\chi_{i_1}, \ldots, \chi_{i_d}$ of a characteristic vector χ by a single value $\chi_{i_1} + \cdots + \chi_{i_d}$ without compromising the quality of the solution.

The location of the honeypots is typically assumed to be known to legitimate users, and the interaction with a honeypot can thus be considered as a reliable indication of an ongoing attack. Real-world computer networks rely also on intrusion detection systems (IDS) to discover potential malicious actions of the attackers. The detection of IDS is, however, not reliable. Another natural direction is thus to extend the algorithm (especially the Equations (4.16)) to account for unreliable detection represented by stochastic transitions in the game.

CHAPTER 5

Towards Two-Sided POSGs

In Chapters 3 and 4, we have studied partially observable stochastic games where only one side of the game has the imperfect information (hence called one-sided POSGs). This approach is well-motivated by security applications. Here, we often do not know what pieces of information are available to the adversary, and hence it is sensible to assume the worst-case scenario, i.e., that the adversary knows everything. By doing so, we are able to obtain robust strategies and obtain strong security guarantees.

Although it is possible to apply similar ideas to every two-player game and thus assume that our opponent has perfect information about the game, such an approach can have undesirable effects. In many real-world games (e.g., card games), the only way to win is to reason about the uncertainty of the adversary and leverage this uncertainty to our advantage. In case we approximate a two-sided game using the one-sided POSG model, we lose this possibility—and the computed strategies can be thus overly conservative. As an example, consider a variant of poker where each player can pay the opponent to reveal their private cards. One of the key characteristics of poker is that the game involves strategic betting, and each player needs to reason about possible cards of the players. In most situations, accepting the payment and revealing the cards to the adversary thus significantly degrades the chances of the player to win the game. If we used the one-sided POSG model to reason about such variant of poker, we would have had to assume that the adversary already knows our card. However, within this assumption, there is no reason to decline the payment. The adversary is assumed to already know our cards, and hence the model of one-sided POSGs does not allow us to understand the negative consequences of revealing the cards to the adversary, and the optimal strategy computed within the one-sided POSG framework would therefore suggest to accept the payment and show the cards.

This chapter is based on following publications:

[Horák and Bošanský, 2019] Horák, K. and Bošanský, B. (2019). Solving partially observable stochastic games with public observations. In *33rd AAAI Conference on Artificial Intelligence*, pages 2029–2036 (70%, 1 citation)

Unfortunately, solving general POSGs with two-sided imperfect information is challenging. The players need not see each other's actions and observations, and hence they form different beliefs [Hansen et al., 2004]. Furthermore, since the decision-making of the agents depends on their beliefs, they need to reason about what is the belief of the opponent, what is the belief of the opponent about our belief, and so on. This reasoning can go on indefinitely and results in so-called *nested beliefs* [MacDermed, 2013].

The approach we presented in Chapter 3 for solving games with one-sided imperfect information relied on the fact that we can characterize the information concisely without the need for the infinite nesting of beliefs. Since this seems not to be possible for general two-sided POSGs with infinite horizon, designing a scalable and optimal value-iteration algorithm for such games does not seem to be possible.

In this chapter, we take a step towards solving games with two-sided partial information by introducing a model of *partially observable stochastic games with public observations* (PO-POSGs) that is inspired by the works of Cole and Kocherlakota [2001] and MacDermed [2013]. However, unlike these works, we focus on a two-player zero-sum setting where randomized strategies are used. PO-POSGs generalize one-sided POSGs by allowing both players to have imperfect information about the private state and actions of the adversary. Although the players form different beliefs, each one of them can infer the belief of his adversary—thus making the reasoning about the nested beliefs unnecessary. We discuss the structural properties of the solution of PO-POSGs, and we show that the algorithm for one-sided POSGs can be adapted to work on PO-POSGs.

5.1 Game Model

Similarly to [Cole and Kocherlakota, 2001], we employ factorization of the state space of the game. Here, a state of the game $s = (s_1, s_2)$ consists of private states s_1 and s_2 of player 1 and player 2, respectively. Furthermore, the dynamics of the game ensures that the beliefs are Markov, i.e., the players can infer each other's belief when the strategies are fixed.

Definition 5.1 (Partially observable stochastic game with public observations)**.** A partially observable stochastic game with public observations (PO-POSG) is a two-player zero-sum game between players $i \in \{1, 2\}^a$ represented by a tuple $\langle S_i, A_i, O_i, Z_i, T_i, R, b^{\text{init}}, \gamma \rangle$, where

- S_i is a finite set of (private) states of player i
- A_i is a finite set of actions available to player i
- O_i is a finite set of observations for player i
- $Z_i(o_i|s_{-i}, a_{-i})$ is the probability to generate observation o_i for player i, given that his opponent $-i$ played action a_{-i} in private state s_{-i}
- $T_i(s'_i|s_i, a_i, o_i, o_{-i})$ is the probability to transition from s_i to s'_i when player i played a_i and public observations o_i and o_{-i} have been generated

- $R(s_1, s_2, a_1, a_2)$ is the reward of player 1 when actions (a_1, a_2) have been jointly played in the joint state (s_1, s_2)
- $b^{\text{init}} = (b_1^{\text{init}}, b_2^{\text{init}})$ is the initial beliefs of the players, where $b_i^{\text{init}} \in \Delta(S_{-i})$ is the belief player i has about the states S_{-i} of the opponent, and
- γ is the discount factor.

[a] As it is commonly used, $-i$ denotes opponent of player i.

A play in a PO-POSG proceeds as follows. First, the initial joint state $(s_1^{(1)}, s_2^{(1)})$ is drawn with probability $b_2^{\text{init}}(s_1^{(1)}) \cdot b_1^{\text{init}}(s_2^{(1)})$. Then, in each stage t, players observe their current private state (player i observes $s_i^{(t)}$, but not $s_{-i}^{(t)}$ of his opponent). Based on this information (and history), each player i chooses actions $a_i^{(t)} \in A_i$ independently of the decision of his opponent $-i$. As a consequence of this choice, player 1 receives reward $r^{(t)} = R(s_1^{(t)}, s_2^{(t)}, a_1^{(t)}, a_2^{(t)})$ and player 2 receives negated reward $-R(s_1^{(t)}, s_2^{(t)}, a_1^{(t)}, a_2^{(t)})$. Furthermore, observation $o_i^{(t)}$ for each player is generated and made publicly known to both players with probability $Z_i(o_i^{(t)} | s_{-i}^{(t)}, a_{-i}^{(t)})$ and a new private state $s_i^{(t+1)}$ of each player is drawn from $T_i(\cdot | s_i^{(t)}, a_i^{(t)}, o_i^{(t)}, o_{-i}^{(t)})$. We consider infinite-horizon discounted setting and the utility of player 1, denoted Disc^γ, is thus $\sum_{t=1}^{\infty} \gamma^{t-1} r^{(t)}$. Player 1 is aiming on maximizing this quantity, while the player 2 is trying to minimize it.

Apart from observing the public observations $(o_1^{(t)}, o_2^{(t)})$, the players are able to recall their own actions and private states only. Player i hence cannot observe the states $s_{-i}^{(t)}$ of the opponent, and the actions $a_{-i}^{(t)}$ the opponent has made. This means that the private history of player i after T stages has passed is a sequence $(s_i^{(t)} a_i^{(t)} o_1^{(t)} o_2^{(t)})_{t=1}^{T} s_i^{T+1}$. Player i can only use this information for decision making, and hence his behavioral strategy in PO-POSGs is defined as follows.

Definition 5.2 (Behavioral strategy)**.** Strategy $\sigma_i : (S_i A_i O_i O_{-i})^* S_i \to \Delta(A_i)$ of player i is a mapping from private histories of player i to randomized decisions. The set of all strategies of player i is denoted Σ_i.

Similarly to one-sided POSGs, we also reason about the strategies the players use to act in the current stage of the game only.

Definition 5.3 (Stage strategies)**.** A mapping $\pi_i : S_i \to \Delta(A_i)$ is called a *stage strategy* of player i. The set of all stage strategies of player i is denoted Π_i.

Importantly, the need for reasoning about the nested beliefs is avoided due to the fact that each player can infer the belief of his adversary. Let us assume that the belief of the adversary i is b_i (i.e., the probability that the current state of player $-i$ is $s_{-i} \in S_{-i}$ is $b_i(s_{-i})$). Now, given that a pair of public observations (o_1, o_2) is observed and player $-i$

plays stage strategy π_{-i}, the probability that the state of player $-i$ changes to s'_{-i} is

$$\mathbb{P}_{b,\pi_1,\pi_2}[s'_{-i} \mid o_i, o_{-i}] = \sum_{s_{-i} \in S_{-i}} \sum_{a_{-i} \in A_{-i}} \mathbb{P}_{b,\pi_1,\pi_2}[s_{-i}, a_{-i}, s'_{-i} \mid o_i, o_{-i}] \tag{5.1a}$$

$$= \frac{1}{\mathbb{P}_{b,\pi_1,\pi_2}[o_i \mid o_{-i}]} \sum_{s_{-i} \in S_{-i}} \sum_{a_{-i} \in A_{-i}} \mathbb{P}_{b,\pi_1,\pi_2}[s_{-i}, a_{-i}, o_i, s'_{-i} \mid o_{-i}] \tag{5.1b}$$

$$= \frac{1}{\mathbb{P}_{b,\pi_1,\pi_2}[o_i \mid o_{-i}]} \sum_{s_{-i} \in S_{-i}} \sum_{a_{-i} \in A_{-i}} \mathbb{P}_{b,\pi_1,\pi_2}[s'_{-i} \mid s_{-i}, a_{-i}, o_i, o_{-i}] \cdot \tag{5.1c}$$

$$\cdot \mathbb{P}_{b,\pi_1,\pi_2}[o_i \mid s_{-i}, a_{-i}, o_{-i}] \cdot \mathbb{P}_{b,\pi_1,\pi_2}[a_{-i} \mid s_{-i}, o_{-i}] \cdot \mathbb{P}_{b,\pi_1,\pi_2}[s_{-i} \mid o_{-i}] \ .$$

Due to the conditional independence, Equation (5.1c) can be simplified. Hence, we have

$$\mathbb{P}_{b,\pi_1,\pi_2}[s'_{-i} \mid o_i, o_{-i}] = \frac{1}{\mathbb{P}_{b,\pi_1,\pi_2}[o_i]} \sum_{s_{-i} \in S_{-i}} \sum_{a_{-i} \in A_{-i}} \mathbb{P}_{b,\pi_1,\pi_2}[s'_{-i} \mid s_{-i}, a_{-i}, o_i, o_{-i}] \cdot \tag{5.1d}$$

$$\cdot \mathbb{P}_{b,\pi_1,\pi_2}[o_i \mid s_{-i}, a_{-i}] \cdot \mathbb{P}_{b,\pi_1,\pi_2}[a_{-i} \mid s_{-i}] \cdot \mathbb{P}_{b,\pi_1,\pi_2}[s_{-i}]$$

$$= \frac{1}{\mathbb{P}_{b,\pi_1,\pi_2}[o_i]} \sum_{s_{-i} \in S_{-i}} \sum_{a_{-i} \in A_{-i}} T_{-i}(s'_{-i} \mid s_{-i}, a_{-i}, o_i, o_{-i}) \cdot \tag{5.1e}$$

$$\cdot Z_i(o_i \mid s_{-i}, a_{-i}) \cdot \pi_{-i}(a_{-i} \mid s_{-i}) \cdot b_i(s_{-i})$$

$$= \tau_i(b_i, \pi_{-i}, o_i, o_{-i})(s'_{-i}) \ . \tag{5.1f}$$

We use $\tau_i(b_i, \pi_{-i}, o_i, o_{-i})$ to denote the Bayesian update of the belief of player i. Note that this update depends solely on the current belief b_i of player i, the stage strategy π_{-i} of player $-i$ and the pair of public observations (o_i, o_{-i})—and no private information of player i can be used to refine the belief of player i. All of this information is also available to player $-i$, and hence both players can perform exactly the same computation.

5.2 Value of PO-POSGs

As in the case of one-sided POSGs, we first define the value of a strategy of player i. Then we define the value of PO-POSGs as the utility the players can achieve by choosing the best strategy to play.

Definition 5.4 (Value of strategy)**.** Let $\sigma_i \in \Sigma_1$ be a behavioral strategy of player i. A function $\mathrm{val}^{\sigma_i \mid b_{-i}} : \Delta(S_{-i}) \to \mathbb{R}$ is called *value of strategy σ_i in the belief b_{-i}* of the adversary, and it is defined as the utility σ_i achieves in the joint belief $b = (b_i, b_{-i})$ against the best response of the adversary, i.e.,

$$\mathrm{val}^{\sigma_1 \mid b_2}(b_1) = \inf_{\sigma_2 \in \Sigma_2} \mathbb{E}_{b_1,b_2,\sigma_1,\sigma_2}[\mathsf{Disc}^\gamma] \ , \text{ and} \tag{5.2}$$

$$\mathrm{val}^{\sigma_2 \mid b_1}(b_2) = \sup_{\sigma_1 \in \Sigma_1} \mathbb{E}_{b_1,b_2,\sigma_1,\sigma_2}[\mathsf{Disc}^\gamma] \ . \tag{5.3}$$

Observe that unlike in one-sided POSGs the value of a strategy σ_i depends on the belief b_{-i} of the adversary. We now represent the value of the game with an initial belief (b_1, b_2) as the utility the best possible strategy achieves in (b_1, b_2), i.e., $\sup_{\sigma_1 \in \Sigma_1} \mathrm{val}^{\sigma_1|b_2}(b_1)$ and $\inf_{\sigma_2 \in \Sigma_2} \mathrm{val}^{\sigma_2|b_1}(b_2)$ from the perspective of player 1 and player 2, respectively. Note that since the value of zero-sum POSGs with discounted-sum objective exists (see Theorem 2.3), these two values coincide.

Definition 5.5 (Optimal value function)**.** A function $V^* : \Delta(S_2) \times \Delta(S_1) \to \mathbb{R}$ where

$$V^*(b_1, b_2) = \sup_{\sigma_1 \in \Sigma_1} \mathrm{val}^{\sigma_1|b_2}(b_1) = \inf_{\sigma_2 \in \Sigma_2} \mathrm{val}^{\sigma_2|b_1}(b_2) \tag{5.4}$$

is called the *optimal value function* of a PO-POSG.

We now proceed by showing structural properties of values of strategies, and the optimal value function V^*. First, we show that when the belief b_{-i} of the adversary is fixed, the value $\mathrm{val}^{\sigma_i|b_{-i}}$ is a linear function in the belief $b_i \in \Delta(S_{-i})$ of player i. Similarly to one-sided POSGs (Proposition 3.1), we use $L = \min R(\cdot)/(1-\gamma)$ and $U = \max R(\cdot)/(1-\gamma)$ to denote minimum and maximum utilities, respectively, in the game.

Lemma 5.1. *Let σ_i be a strategy of player i, and let us assume that the belief $b_{-i} \in \Delta(S_i)$ of the adversary is fixed. Then the expected utility $\mathrm{val}^{\sigma_i|b_{-i}} : \Delta(S_{-i}) \to \mathbb{R}$ of playing σ_i against the best-responding opponent $-i$ is linear in the belief $b_i \in \Delta(S_{-i})$ of player i, and it is $(U-L)/2$-Lipschitz continuous.*

Proof. Player $-i$ knows σ_i as well as his true state s_{-i}, and his only uncertainty is about the state s_i (the probability of which is $b_{-i}(s_i)$). It is thus possible to focus on a best response for each state s_{-i} separately. Let us denote the expected utility of playing a best response against σ_i starting from s_{-i} (when $s_i \sim b_{-i}$) by $\xi(s_{-i})$. Since the strategy σ_i is fixed (and thus does not depend on b_i), the expected value of playing σ_i against a best response of the adversary is the expectation over values $\xi(s_{-i})$, $\mathrm{val}^{\sigma_i|b_{-i}}(b_i) = \sum_{s_{-i}} b_i(s_{-i}) \cdot \xi(s_{-i})$, and thus the value $\mathrm{val}^{\sigma_i|b_{-i}}$ is linear in b_i. Moreover, observe that similarly to one-sided POSGs (see Proposition 3.1), the values are bounded, and we have

$$L = \frac{\min R(\cdot)}{1-\gamma} \leq \mathrm{val}^{\sigma_i|b_{-i}}(b_i) \leq \frac{\max R(\cdot)}{1-\gamma} = U \tag{5.5}$$

Using Lemma 3.4 allows us to conclude that $\mathrm{val}^{\sigma_i|b_{-i}}$ is $(U-L)/2$-Lipschitz continuous. □

We now leverage that the optimal value function V^* is defined as a supremum and infimum over the values of strategies of player 1 and player 2, respectively, to establish that the value function V^* is convex-concave and $(U-L)/2$-Lipschitz continuous. As in the case of one-sided POSGs, we define $\delta = (U-L)/2$, and we endow the relevant spaces with the L_1-norm. We first show a technical proposition followed by the main result.

Proposition 5.2. *Let $V : \Delta(S_2) \times \Delta(S_1) \to \mathbb{R}$ be a value function which is δ-Lipschitz continuous in the belief $b_1 \in \Delta(S_2)$ of player 1 and δ-Lipschitz continuous in the belief $b_2 \in \Delta(S_1)$ of player 2. Then V is δ-Lipschitz continuous.*

Proof. Let $(b_1, b_2) \in \Delta(S_2)\times\Delta(S_1)$ and $(b'_1, b'_2) \in \Delta(S_2)\times\Delta(S_1)$ be arbitrary. Since V is δ-Lipschitz in the beliefs of player 1 and player 2, we have $|V(b_1, b_2)-V(b'_1, b_2)| \leq \delta\|b_1-b'_1\|_1$ and $|V(b'_1, b_2) - V(b'_1, b'_2)| \leq \delta\|b_2 - b'_2\|_1$. Now, we have

$$\begin{aligned}\delta\|(b_1, b_2) - (b'_1, b'_2)\|_1 &= \delta\|b_1 - b'_1\|_1 + \delta\|b_2 - b'_2\|_1 \\ &\geq |V(b_1, b_2) - V(b'_1, b_2)| + |V(b'_1, b_2) - V(b'_1, b'_2)| \\ &\geq |V(b_1, b_2) - V(b'_1, b_2) + V(b'_1, b_2) - V(b'_1, b'_2)| \\ &= |V(b_1, b_2) - V(b'_1, b'_2)|\end{aligned}$$

which proves δ-Lipschitz continuity of V. □

Theorem 5.3. *Value function V^* is convex and δ-Lipschitz continuous in the belief $b_1 \in \Delta(S_2)$ of the maximizing player 1 and concave and δ-Lipschitz continuous in the belief $b_2 \in \Delta(S_1)$ of the minimizing player 2. Moreover, V^* is δ-Lipschitz continuous.*

Proof. According to the Definition 5.5, for a fixed b_2, player 1 chooses a strategy that maximizes the utility, hence

$$V^*(b_1, b_2) = \sup_{\sigma_1 \in \Sigma_1} \mathrm{val}^{\sigma_1|b_2}(b_1) \tag{5.6}$$

As all $\mathrm{val}^{\sigma_1|b_2}$ are linear, V^* is convex in b_1. Vice versa, for given fixed b_1, player 2 chooses a minimizing strategy. Hence

$$V^*(b_1, b_2) = \inf_{\sigma_2 \in \Sigma_2} V^{\sigma_2|b_1}(b_2) \tag{5.7}$$

and V^* is concave in b_2. Since V^* is a pointwise supremum/infimum (Equations (5.6) and (5.7)) from δ-Lipschitz continuous functions $\mathrm{val}^{\sigma_i|b_{-i}}$, V^* is δ-Lipschitz continuous in the dimension of b_1 as well as b_2. Combining the Lipschitz constants in these two dimensions in the sense of Proposition 5.2 results in δ-Lipschitz continuity of V^*. □

5.3 Composing Strategies

Similarly to one-sided POSGs, every strategy of player i can be decomposed to the decision rule π_i used in the first stage of the game, and the behavioral strategies player i may use in the rest of the game. Here, we define the opposite principle of *strategy composition* where we combine a vector $(\Sigma_i)^{S_i\times A_i\times O_i\times O_{-i}}$ of strategies the player i chose

to follow after each possible outcome $(s_i, a_i, o_i, o_{-i}) \in S_i \times A_i \times O_i \times O_{-i}$ of the first stage where he follows a stage strategy π_i.

Definition 5.6 (Strategy composition)**.** Let $\overline{\zeta}^i = (\Sigma_i)^{S_i \times A_i \times O_i \times O_{-i}}$ be a vector of strategies and $\pi_i \in \Pi_i$ be a stage strategy for the first stage of the game. A behavioral strategy $\mathsf{comp}(\pi_i, \overline{\zeta}^i) \in \Sigma_1$ is called a *strategy composition of* $\overline{\zeta}^i$ *using* π_i, and is defined as

$$\mathsf{comp}(\pi_i, \overline{\zeta}^i)(s_i) = \pi_i(\cdot \mid s_i) \text{ , and}$$
$$\mathsf{comp}(\pi_i, \overline{\zeta}^i)(s_i a_i o_i o_{-i} \omega) = \zeta^i_{s_i, a_i, o_i, o_{-i}}(\omega) \text{ for every } s_i a_i o_i o_{-i} \omega \in (S_i A_i O_i O_{-i})^* S_i \text{ .}$$

Importantly, the value $\mathrm{val}^{\mathsf{comp}(\pi_i, \overline{\zeta}^i)|b_{-i}}$ can be computed similarly as in Lemma 3.11. The key difference here is that we have to reflect the belief $\tau_{-i}(b_{-i}, \pi_i, o_i, o_{-i})$ of the adversary when the strategy $\zeta^i_{s_i, a_i, o_i, o_{-i}}$ is about to be followed. We present the following result from the perspective of the maximizing player 1. The formulation for the minimizing player 2 can be obtained analogously by switching the roles of the players and replacing minimization over stage strategies $\pi_2 \in \Pi_2$ of player 2 characterizing her best response, by maximization over strategies $\pi_1 \in \Pi_1$ of player 1. Note that we prove a weaker result involving inequality, since player 2 knows only the public observations (o_1, o_2) and not state s_1 and action a_1 of player 1. The player 1, however, conditions the strategy $\zeta^1_{s_1, a_1, o_1, o_2}$ he follows in a subgame even on s_1 and a_1. Player 2 thus does not know exactly which strategy $\zeta^1_{s_1, a_1, o_1, o_2}$ is being followed—and hence need not be able to best-respond it to get value $\mathrm{val}^{\zeta^1_{s_1, a_1, o_1, o_2} | \tau_2(b_2, \pi_1, o_1, o_2)}(\tau_1(b_1, \pi_2, o_1, o_2))$ as considered in the proof.

Lemma 5.4. *Let* $\mathsf{comp}(\pi_1, \overline{\zeta}^1)$ *be a strategy composition of player 1, and* $(b_1, b_2) \in \Delta(S_2) \times \Delta(S_1)$ *be an arbitrary joint belief. Then the value* $\mathrm{val}^{\mathsf{comp}(\pi_1, \overline{\zeta}^1)|b_2}$ *of the strategy composition* $\mathsf{comp}(\pi_1, \overline{\zeta}^1)$ *satisfies*

$$\mathrm{val}^{\mathsf{comp}(\pi_1, \overline{\zeta}^1)|b_2}(b_1) \geq \min_{\pi_2 \in \Pi_2} \Big[\mathbb{E}_{b, \pi_1, \pi_2}[R(s_1, s_2, a_1, a_2)] + \tag{5.8}$$
$$+ \gamma \sum_{(s_1, a_1, o_1, o_2)} \mathbb{P}_{b, \pi_1, \pi_2}[s_1, a_1, o_1, o_2] \cdot \mathrm{val}^{\zeta^1_{s_1, a_1, o_1, o_2} | \tau_2(b_2, \pi_1, o_1, o_2)}(\tau_1(b_1, \pi_2, o_1, o_2)) \Big] \text{ .}$$

Proof. Let $b = (b_1, b_2) \in \Delta(S_2) \times \Delta(S_2)$ be an arbitrary joint belief. We will provide a lower bound $\underline{u}(\pi_2)$ on the utility the composite strategy $\mathsf{comp}(\pi_1, \overline{\zeta}^1)$ guarantees when player 2 plays an arbitrary stage strategy $\pi_2 \in \Pi_2$ in the first stage. We get the lower bound on the utility $\mathsf{comp}(\pi_1, \overline{\zeta}^1)$ guarantees against *any* strategy of player 2 (i.e., even against the best response of player 2 as in Definition 5.4) by allowing the player 2 to choose arbitrary stage strategy π_2, and results in Equation (5.8), which then completes the proof.

Let $\pi_2 \in \Pi_2$ be arbitrary. The expected reward in the first stage when playing π_2 against $\mathsf{comp}(\pi_1, \overline{\zeta}^1)$ is $\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1, s_2, a_1, a_2)]$. It remains to focus on the rewards player 2 is able to achieve in the rest of the game. By Definition 5.4, the minimizing player 2 cannot achieve lower utility than $\mathrm{val}^{\zeta^1_{s_1,a_1,o_1,o_2}|\tau_2(b_2,\pi_1,o_1,o_2)}(s'_2)$ when the player 1 is about to play strategy $\zeta^1_{s_1,a_1,o_1,o_2}$, the current distribution over states of player 1 (i.e., the belief of player 2) is $\tau_2(b_2, \pi_1, o_1, o_2)$ and the current state of player 2 is s'_2. Hence we have

$$\begin{aligned}
\underline{u}(\pi_2) &\geq \mathbb{E}_{b,\pi_1,\pi_2}[R(s_1, s_2, a_1, a_2)] + \gamma \sum_{(s_1,a_1,o_1,o_2)} \mathbb{P}_{b,\pi_1,\pi_2}[s_1, a_1, o_1, o_2] \sum_{s'_2} \mathbb{P}_{b,\pi_1,\pi_2}[s'_2 \mid o_1, o_2] \cdot \\
&\qquad \cdot \mathrm{val}^{\zeta^1_{s_1,a_1,o_1,o_2}|\tau_2(b_2,\pi_1,o_1,o_2)}(s'_2) \\
&= \mathbb{E}_{b,\pi_1,\pi_2}[R(s_1, s_2, a_1, a_2)] + \gamma \sum_{(s_1,a_1,o_1,o_2)} \mathbb{P}_{b,\pi_1,\pi_2}[s_1, a_1, o_1, o_2] \sum_{s'_2} \tau_1(b_1, \pi_2, o_1, o_2)(s'_2) \cdot \\
&\qquad \cdot \mathrm{val}^{\zeta^1_{s_1,a_1,o_1,o_2}|\tau_2(b_2,\pi_1,o_1,o_2)}(s'_2) \\
&= \mathbb{E}_{b,\pi_1,\pi_2}[R(s_1, s_2, a_1, a_2)] + \\
&\qquad + \gamma \sum_{(s_1,a_1,o_1,o_2)} \mathbb{P}_{b,\pi_1,\pi_2}[s_1, a_1, o_1, o_2] \cdot \mathrm{val}^{\zeta^1_{s_1,a_1,o_1,o_2}|\tau_2(b_2,\pi_1,o_1,o_2)}(\tau_1(b_1, \pi_2, o_1, o_2)) \ .
\end{aligned}$$

By allowing the player to choose π_2 that minimizes $\underline{u}(\pi_2)$, we get the claimed inequality. □

5.4 Bellman's Equation for PO-POSGs

In this section, we provide a Bellman's equation for PO-POSGs which allows us to approximate the optimal value function V^* iteratively. The structure of the Bellman's equation is similar to one-sided POSGs (see Theorem 3.15). The players choose their stage strategies $\pi_1 \in \Pi_1$ and $\pi_2 \in \Pi_2$ to optimize their expected utility. This utility consists of the expected reward in the first stage of the game starting from the joint belief (b_1, b_2) and the utility the players are able to achieve in the rest of the game. We prove that V^* is the solution of the Bellman's Equation (5.10), and that the Bellman's operator is a contraction mapping.

Theorem 5.5. *Let $b = (b_1, b_2) \in \Delta(S_2) \times \Delta(S_1)$ be an arbitrary joint belief. Optimal value function V^* satisfies*

$$\begin{aligned}
V^*(b_1, b_2) = [HV^*](b_1, b_2) = \max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2} \Big[&\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1, s_2, a_1, a_2)] + \\
&+ \gamma \sum_{(o_1,o_2) \in O_1 \times O_2} \mathbb{P}_{b,\pi_1,\pi_2}[o_1, o_2] \cdot V^*\left(\tau_1(b_1, \pi_2, o_1, o_2), \tau_2(b_2, \pi_1, o_2, o_1)\right) \Big] \ .
\end{aligned} \tag{5.9}$$

We provide the proof of Theorem 5.5 at the end of this section. The Equation (5.10) introduces the Bellman's operator H that we now define formally.

Definition 5.7 (Bellman's operator)**.** Denote $\mathcal{F}_{\Delta(S_2)\times\Delta(S_1)}$ the set of all continuous value functions $f : \Delta(S_2) \times \Delta(S_1) \to \mathbb{R}$. *Bellman's operator* is a mapping $H : \mathcal{F}_{\Delta(S_2)\times\Delta(S_1)} \to \mathcal{F}_{\Delta(S_2)\times\Delta(S_1)}$ where

$$[HV](b_1, b_2) = \max_{\pi_1\in\Pi_1} \min_{\pi_2\in\Pi_2} \Big[\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1, s_2, a_1, a_2)] + \\ + \gamma \sum_{(o_1,o_2)\in O_1\times O_2} \mathbb{P}_{b,\pi_1,\pi_2}[o_1, o_2] \cdot V\left(\tau_1(b_1, \pi_2, o_1, o_2), \tau_2(b_2, \pi_1, o_2, o_1)\right)\Big] . \tag{5.10}$$

Before we prove Theorem 5.5, we define a stage game based on the objective of the maximin optimization in Equation (5.10), and we prove an additional lemma that allows us to swap the order of maximization and minimization over π_1 and π_2, respectively, in the Equation (5.10).

Definition 5.8 (Stage game)**.** Let $V : \Delta(S_2) \times \Delta(S_1) \to \mathbb{R}$ be a continuous convex-concave value function mapping joint beliefs $(b_1, b_2) \in \Delta(S_2)\times\Delta(S_1)$ to real numbers. Let $b = (b_1, b_2) \in \Delta(S_2) \times \Delta(S_1)$ be an arbitrary joint belief. A two-player zero-sum game with strategy spaces Π_1 and Π_2 for the maximizing and minimizing players, respectively, and utility function $u^{V,b}$, where

$$u^{V,b}(\pi_1, \pi_2) = \mathbb{E}_{b,\pi_1,\pi_2}[R(s_1, s_2, a_1, a_2)] + \\ + \gamma \sum_{(o_1,o_2)\in O_1\times O_2} \mathbb{P}_{b,\pi_1,\pi_2}[o_1, o_2] \cdot V\left(\tau_1(b_1, \pi_2, o_1, o_2), \tau_2(b_2, \pi_1, o_2, o_1)\right)\Big] \tag{5.11}$$

is called a *stage game with respect to the value function V and joint belief b.*

Lemma 5.6. *Let $V : \Delta(S_2) \times \Delta(S_1)$ be a continuous convex-concave function (i.e., convex in $b_1 \in \Delta(S_2)$ and concave in $b_2 \in \Delta(S_1)$, and let $b = (b_1, b_2) \in \Delta(S_2)\times\Delta(S_1)$ be an arbitrary joint belief. Then*

$$\max_{\pi_1\in\Pi_1} \min_{\pi_2\in\Pi_2} u^{V,b}(\pi_1, \pi_2) = \min_{\pi_2\in\Pi_2} \max_{\pi_1\in\Pi_1} u^{V,b}(\pi_1, \pi_2) . \tag{5.12}$$

Proof. We use von Neumann's minimax theorem [von Neumann, 1928; Nikaido, 1953] to prove the equivalence. Clearly, sets Π_1 and Π_2 are convex compact sets, hence it suffices to check that the utility function $u^{V,b}$ is convex in the strategy $\pi_2 \in \Pi_2$ of the minimizing player, and concave in the strategy $\pi_1 \in \Pi_1$ of the maximizing player.

The expectation $\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1, s_2, a_1, a_2)]$ over the finite number of possible outcomes of the first stage is linear in both π_1 and π_2, hence it suffices to focus on the second term in $u^{V,b}$. We prove that $\gamma \sum_{o_1,o_2} \mathbb{P}_{b,\pi_1,\pi_2}[o_1, o_2] \cdot V(\tau_1(b_1, \pi_2, o_1, o_2), \tau_2(b_2, \pi_1, o_1, o_2))$ is convex in π_2. The fact that this expression is concave in π_1 can be shown analogously.

Let us fix the stage strategy $\pi_1 \in \Pi_1$ of player 1, and define $V|_{b_2} : \Delta(S_2) \to \mathbb{R}$ as a function $V|_{b_2}(b_1) = V(b_1, b_2)$. Since V is convex in b_1, the function $V|_{b_2}$ is convex as well—and hence by Proposition 3.9 we can represent $V|_{b_2}$ as a supremum of linear functions Γ^{b_2}. Now, we have

$$\gamma \sum_{(o_1,o_2)\in O_1\times O_2} \mathbb{P}_{b,\pi_1,\pi_2}[o_1,o_2] \cdot V(\tau_1(b_1,\pi_2,o_1,o_2),\tau_2(b_2,\pi_1,o_1,o_2)) = \tag{5.13a}$$

$$= \gamma \sum_{(o_1,o_2)\in O_1\times O_2} \mathbb{P}_{b,\pi_1,\pi_2}[o_1] \cdot \mathbb{P}_{b,\pi_1,\pi_2}[o_2] \cdot V|_{\tau_2(b_2,\pi_1,o_1,o_2)}(\tau_1(b_1,\pi_2,o_1,o_2)) \tag{5.13b}$$

$$= \gamma \sum_{(o_1,o_2)\in O_1\times O_2} \mathbb{P}_{b,\pi_1,\pi_2}[o_1] \cdot \mathbb{P}_{b,\pi_1,\pi_2}[o_2] \cdot \sup_{\alpha\in\Gamma^{\tau_2(b_2,\pi_1,o_1,o_2)}} \alpha(\tau_1(b_1,\pi_2,o_1,o_2)) \tag{5.13c}$$

$$= \gamma \sum_{(o_1,o_2)\in O_1\times O_2} \mathbb{P}_{b,\pi_1,\pi_2}[o_1] \cdot \mathbb{P}_{b,\pi_1,\pi_2}[o_2] \cdot \sup_{\alpha\in\Gamma^{\tau_2(b_2,\pi_1,o_1,o_2)}} \sum_{s_2'\in S_2} \tau_1(b_1,\pi_2,o_1,o_2)(s_2') \cdot \alpha(s_2)\ . \tag{5.13d}$$

Equation (5.13b) leverages that the observations are generated independently and rewrites $V(b_1, b_2)$ as $V|_{b_2}(b_1)$. Equations (5.13c) and (5.13d) rewrite the convex functions $V|_{b_2}$ as supremum over linear functions. After expanding τ_1 using Equation (5.1e) and canceling out the terms $\mathbb{P}_{b,\pi_1,\pi_2}[o_1]$, we further have

$$\gamma \sum_{(o_1,o_2)\in O_1\times O_2} \mathbb{P}_{b,\pi_1,\pi_2}[o_1,o_2] \cdot V(\tau_1(b_1,\pi_2,o_1,o_2),\tau_2(b_2,\pi_1,o_1,o_2)) =$$

$$= \gamma \sum_{(o_1,o_2)\in O_1\times O_2} \mathbb{P}_{b,\pi_1,\pi_2}[o_2] \cdot \sup_{\alpha\in\Gamma^{\tau_2(b_2,\pi_1,o_1,o_2)}} \sum_{(s_2,a_2,s_2')\in S_2\times A_2\times S_2} \alpha(s_2')T_2(s_2' \mid s_2,a_2,o_1,o_2)\cdot \tag{5.13e}$$

$$\cdot Z_1(o_1 \mid s_2,a_2) \cdot \pi_2(a_2 \mid s_2) \cdot b_1(s_2)\ .$$

The probability $\mathbb{P}_{b,\pi_1,\pi_2}[o_2]$ is independent of the stage strategy π_2 of player 2 (observation o_2 depends only on states and actions of player 1), hence the expression in Equation (5.13e) is a supremum over functions that are linear in π_2. Expression in Equation (5.13e) is therefore convex in π_2, as is $u^{V,b}$. □

We are now ready to prove that V^* solves the Bellman's equation introduced in Theorem 5.5.

Proof of Theorem 5.5. Let $b = (b_1, b_2) \in \Delta(S_2) \times \Delta(S_1)$ be an arbitrary joint belief. We prove that $[HV^*](b_1, b_2) = \max_{\pi_1} \min_{\pi_2} u^{V^*,b}(\pi_1, \pi_2) \leq V^*(b_1, b_2)$. We have

$$\begin{aligned}[HV^*](b_1,b_2) &= \max_{\pi_1\in\Pi_1} \min_{\pi_2\in\Pi_2} \Big[\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1,s_2,a_1,a_2)] + \\ &\qquad + \gamma \sum_{(o_1,o_2)} \mathbb{P}_{b,\pi_1,\pi_2}[o_1,o_2] \cdot V^*(\tau_1(b_1,\pi_2,o_1,o_2),\tau_2(b_2,\pi_1,o_1,o_2))\Big] \\ &= \max_{\pi_1\in\Pi_1} \min_{\pi_2\in\Pi_2} \Big[\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1,s_2,a_1,a_2)] + \\ &\qquad + \gamma \sum_{(s_1,a_1,o_1,o_2)} \mathbb{P}_{b,\pi_1,\pi_2}[s_1,a_1,o_1,o_2] \cdot V^*(\tau_1(b_1,\pi_2,o_1,o_2),\tau_2(b_2,\pi_1,o_1,o_2))\Big]\ .\end{aligned}$$

Denote $\Gamma_{\pi_1}^{o_1,o_2} = \{\mathrm{val}^{\sigma_1|\tau(b_2,\pi_1,o_1,o_2)} \mid \sigma_1 \in \Sigma_1\}$ the set of all values $\mathrm{val}^{\sigma_1|\tau(b_2,\pi_1,o_1,o_2)}$ of strategies $\sigma_1 \in \Sigma_1$ of player 1 when the belief of the adversary is $\tau(b_2, \pi_1, o_1, o_2)$. By Definition 5.5 and Proposition 3.8, we have

$$\begin{aligned}
[HV^*](b_1,b_2) &= \max_{\pi_1\in\Pi_1}\min_{\pi_2\in\Pi_2}\Big[\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1,s_2,a_1,a_2)] + \\
&\qquad + \gamma \sum_{(s_1,a_1,o_1,o_2)} \mathbb{P}_{b,\pi_1,\pi_2}[s_1,a_1,o_1,o_2] \cdot \sup_{\alpha\in\Gamma_{\pi_1}^{o_1,o_2}} \alpha(\tau_1(b_1,\pi_2,o_1,o_2))\Big] \\
&= \max_{\pi_1\in\Pi_1}\min_{\pi_2\in\Pi_2}\Big[\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1,s_2,a_1,a_2)] + \\
&\qquad + \gamma \sum_{(s_1,a_1,o_1,o_2)} \mathbb{P}_{b,\pi_1,\pi_2}[s_1,a_1,o_1,o_2] \cdot \sup_{\alpha\in\mathsf{Conv}(\Gamma_{\pi_1}^{o_1,o_2})} \alpha(\tau_1(b_1,\pi_2,o_1,o_2))\Big] \\
&= \max_{\pi_1\in\Pi_1}\min_{\pi_2\in\Pi_2}\sup_{\overline{\alpha}|\alpha_{s_1,a_1,o_1,o_2}\in\mathsf{Conv}(\Gamma_{\pi_1}^{o_1,o_2})}\Big[\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1,s_2,a_1,a_2)] + \qquad (5.14)\\
&\qquad + \gamma \sum_{(s_1,a_1,o_1,o_2)} \mathbb{P}_{b,\pi_1,\pi_2}[s_1,a_1,o_2] \cdot \mathbb{P}_{b,\pi_1,\pi_2}[o_1] \cdot \alpha_{s_1,a_1,o_1,o_2}(\tau_1(b_1,\pi_2,o_1,o_2))\Big] .
\end{aligned}$$

The last equality uses the fact that the observation o_1 is generated independently of s_1, a_1 and o_2 (it only conditionally depends on the state s_2 and action a_2 of player 2). After expanding $\tau_1(b_1, \pi_2, o_1, o_2)$ and using that each α_{s_1,a_1,o_1,o_2} is a linear function, we get

$$\begin{aligned}
[HV^*](b_1,b_2) = \max_{\pi_1\in\Pi_1}\min_{\pi_2\in\Pi_2}\sup_{\overline{\alpha}|\alpha_{s_1,a_1,o_1,o_2}\in\mathsf{Conv}(\Gamma_{\pi_1}^{o_1,o_2})}&\Big[\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1,s_2,a_1,a_2)] + \\
&+ \gamma \sum_{(s_1,a_1,o_1,o_2)} \mathbb{P}_{b,\pi_1,\pi_2}[s_1,a_1,o_2] \sum_{(s_2,a_2,s_2')} b_1(s_2)\cdot\pi_2(a_2 \mid s_2)\cdot \\
&\cdot Z_1(o_1 \mid s_2,a_2)\cdot T(s_2' \mid s_2,a_2,o_1,o_2)\cdot\alpha_{s_1,a_1,o_1,o_2}(s_2')\Big]
\end{aligned}$$

The objective of the min-sup optimization problem is continuous and linear in both π_2 and $\overline{\alpha}$, and since the set of all $\overline{\alpha}$ is convex, and Π_2 is a convex compact set, we can apply the Sion's minimax theorem [Sion, 1958] to Equation (5.14) to get

$$\begin{aligned}
[HV^*](b_1,b_2) = \max_{\pi_1\in\Pi_1}\sup_{\overline{\alpha}|\alpha_{s_1,a_1,o_1,o_2}\in\mathsf{Conv}(\Gamma_{\pi_1}^{o_1,o_2})}\min_{\pi_2\in\Pi_2}&\Big[\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1,s_2,a_1,a_2)] + \qquad (5.15)\\
&+ \gamma \sum_{(s_1,a_1,o_1,o_2)} \mathbb{P}_{b,\pi_1,\pi_2}[s_1,a_1,o_2,o_1] \cdot \alpha_{s_1,a_1,o_1,o_2}(\tau_1(b_1,\pi_2,o_1,o_2))\Big] .
\end{aligned}$$

Every $\alpha_{s_1,a_1,o_1,o_2} \in \mathsf{Conv}(\Gamma_{\pi_1}^{o_1,o_2})$ is a convex combination of values $\sum_{\sigma_1}\lambda^{\sigma_1}\mathrm{val}^{\sigma_1|\tau(b_2,\pi_1,o_1,o_2)}$ of strategies of player 1. We can interpret such combination as a value of a strategy where player 1 mixes between strategies $\sigma_1 \in \Sigma_1$, and plays σ_1 with probability λ^{σ_1}. This is a valid strategy of player 1, and hence its value $\mathrm{val}^{\Sigma\lambda^{\sigma_1}\sigma_1|\tau(b_2,\pi_1,o_1,o_2)} \geq \alpha_{s_1,a_1,o_1,o_2}$ is contained in the set $\Gamma_{\pi_1}^{o_1,o_2}$. We can therefore avoid convexifying the sets $\Gamma_{\pi_1}^{o_1,o_2}$ and get

$$
\begin{aligned}
&[HV^*](b_1, b_2) = \\
&= \max_{\pi_1 \in \Pi_1} \sup_{\overline{\alpha} | \alpha_{s_1,a_1,o_1,o_2} \in \Gamma^{o_1,o_2}_{\pi_1}} \min_{\pi_2 \in \Pi_2} \Big[\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1, s_2, a_1, a_2)] + \qquad (5.16a) \\
&\quad + \gamma \sum_{(s_1,a_1,o_1,o_2)} \mathbb{P}_{b,\pi_1,\pi_2}[s_1, a_1, o_2, o_1] \cdot \alpha_{s_1,a_1,o_1,o_2}(\tau_1(b_1, \pi_2, o_1, o_2)) \Big] \\
&= \max_{\pi_1 \in \Pi_1} \sup_{\overline{\zeta}^1 \in (\Sigma_1)^{S_1 \times A_1 \times O_1 \times O_2}} \min_{\pi_2 \in \Pi_2} \Big[\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1, s_2, a_1, a_2)] + \qquad (5.16b) \\
&\quad + \gamma \sum_{(s_1,a_1,o_1,o_2)} \mathbb{P}_{b,\pi_1,\pi_2}[s_1, a_1, o_2, o_1] \cdot \mathrm{val}^{\zeta^1_{s_1,a_1,o_1,o_2} | \tau_2(b_2,\pi_1,o_1,o_2)}(\tau_1(b_1, \pi_2, o_1, o_2)) \Big] \ .
\end{aligned}
$$

The criterion of the supremum in Equation (5.16b) coincides with the lower bound on the value of a composite strategy $\mathsf{comp}(\pi_1, \overline{\zeta}^1)$. We can therefore apply Lemma 5.4 to get

$$
[HV^*](b_1, b_2) \leq \sup \left\{ \mathrm{val}^{\mathsf{comp}(\pi_1,\overline{\zeta}^1) | b_2} \mid \pi_1 \in \Pi_1, \overline{\zeta}^1 \in (\Sigma_1)^{S_1 \times A_1 \times O_1 \times O_2} \right\} .
$$

Since every composite strategy is a valid behavioral strategy of player 1, we further have

$$
\begin{aligned}
[HV^*](b_1, b_2) &\leq \sup \left\{ \mathrm{val}^{\mathsf{comp}(\pi_1,\overline{\zeta}^1) | b_2}(b_1) \mid \pi_1 \in \Pi_1, \overline{\zeta}^1 \in (\Sigma_1)^{S_1 \times A_1 \times O_1 \times O_2} \right\} \\
&= \sup_{\sigma_1 \in \Sigma_1} \mathrm{val}^{\sigma_1 | b_2}(b_1) = V^*(b_1, b_2) \ .
\end{aligned}
$$

By Lemma 5.6, we have that $\max_{\pi_1} \min_{\pi_2} u^{V^*,b}(\pi_1, \pi_2) = \min_{\pi_2} \max_{\pi_1} u^{V^*,b}(\pi_1, \pi_2)$. We can thus reverse the roles of the players and use an analogous proof to show that $[HV^*](b_1, b_2) = \min_{\pi_2} \max_{\pi_1} u^{V^*,b}(\pi_1, \pi_2) \geq V^*(b_1, b_2)$. Hence $[HV^*](b_1, b_2) = V^*(b_1, b_2)$. □

5.4.1 Contractivity Properties of Bellman's Operator

In Theorem 5.5, we have introduced the Bellman's operator H for PO-POSGs, and we have shown that the optimal value function V^* is the solution of the Bellman's equation $V^* = HV^*$. In this section, we show that the operator H is a contraction—hence V^* is the only solution to the Bellman's equation. Furthermore, we can generate a sequence of value functions $\{V_i\}_{i=1}^{\infty}$ where $V_{i+1} = HV_i$ to iteratively approximate the fixpoint V^*.

Lemma 5.7. *Let $b = (b_1, b_2) \in \Delta(S_2) \times \Delta(S_1)$ be a joint belief and $V, W : \Delta(S_2) \times \Delta(S_1) \to \mathbb{R}$ be two continuous convex-concave value functions such that $[HV](b) \leq [HW](b)$. Let (π_1^V, π_2^V) and (π_1^W, π_2^W) be Nash equilibrium strategy profiles in stage games $[HV](b)$ and $[HW](b)$, respectively, and $C \geq 0$. Assume that*

$$
W(\tau_1(b_1, \pi_2^V, o_1, o_2), \tau_2(b_2, \pi_1^W, o_1, o_2)) - V(\tau_1(b_1, \pi_2^V, o_1, o_2), \tau_2(b_2, \pi_1^W, o_1, o_2)) \leq C
$$

for every (o_1, o_2) where $\mathbb{P}_{b,\pi_1^W,\pi_2^V}[o_1, o_2] > 0$. Then $[HW](b) - [HV](b) \leq \gamma C$.

Proof. The proof is analogous to the proof of Lemma 3.17. Deviating from the Nash equilibrium strategy can only worsen the utility of the player, hence we have

$$u^{V,b}(\pi_1^W, \pi_2^V) \leq u^{V,b}(\pi_1^V, \pi_2^V) = [HV](b) \leq \\ \leq [HW](b) = u^{W,b}(\pi_1^W, \pi_2^W) \leq u^{W,b}(\pi_1^W, \pi_2^V) \ . \tag{5.17}$$

By subtracting $u^{W,b}(\pi_1^W, \pi_2^V) - u^{V,b}(\pi_1^W, \pi_2^V)$, we get

$$u^{W,b}(\pi_1^W, \pi_2^V) - u^{V,b}(\pi_1^W, \pi_2^V) = \gamma \sum_{(o_1,o_2)} \mathbb{P}_{b,\pi_1^W,\pi_2^V}[o_1, o_2] \cdot \\ \cdot \left[W(\tau_1(b_1, \pi_2^V, o_1, o_2), \tau_2(b_2, \pi_1^W, o_1, o_2)) - V(\tau_1(b_1, \pi_2^V, o_1, o_2), \tau_2(b_2, \pi_1^W, o_1, o_2))\right] \ . \tag{5.18}$$

According to the assumption, the difference $W(\cdot) - V(\cdot)$ in Equation (5.18) is less than or equal to C for every (o_1, o_2) with non-zero probability $\mathbb{P}_{b,\pi_1^W,\pi_2^V}[o_1, o_2]$. Hence, Equation (5.18) is an expectation over values that are all at most C. This expectation is then multiplied by $\gamma < 1$, and we have

$$[HW](b) - [HV](b) \leq u^{W,b}(\pi_1^W, \pi_2^V) - u^{V,b}(\pi_1^W, \pi_2^V) \leq \gamma C \tag{5.19}$$

which concludes the proof. □

Theorem 5.8. *Bellman's operator H is a contraction mapping with contractivity factor H.*

Proof. The proof is similar to the proof of Lemma 4.1. Let us consider continuous value functions $V, W : \Delta(S_2) \times \Delta(S_1)$ such that $\|V - W\|_\infty = \max_{(b_1,b_2)\in\Delta(S_2)\times\Delta(S_1)} |V(b_1, b_2) - W(b_1, b_2)| \leq C$ for some $C \geq 0$. We show that $|[HV](b_1, b_2) - [HW](b_1, b_2)| \leq \gamma C$ for arbitrary joint belief $(b_1, b_2) \in \Delta(S_2) \times \Delta(S_1)$.

Let $b = (b_1, b_2) \in \Delta(S_2) \times \Delta(S_1)$. Without loss of generality, assume that $[HV](b) \leq [HW](b)$. Let π_1^V and π_1^W be the maximizers from the problem $[HV](b)$ and $[HW](b)$, respectively, as introduced in Equation (5.10). We will show that by using π_1^W instead of π_1^V in $[HV](b)$, the utility is at least $[HW](b) - \gamma C$ which shows that $[HW](b) - [HV](b) \leq \gamma C$.

$$[HV](b) = \max_{\pi_1\in\Pi_1} \min_{\pi_2\in\Pi_2} u^{V,b}(\pi_1, \pi_2) \geq \min_{\pi_2\in\Pi_2} u^{V,b}(\pi_1^W, \pi_2) \\ = \min_{\pi_2\in\Pi_2} \Big[\mathbb{E}_{b,\pi_1^W,\pi_2}[R(s_1, s_2, a_1, a_2)] + \\ + \gamma \sum_{(o_1,o_2)\in O_1\times O_2} \Pr_{b,\pi_1^W,\pi_2}[o_1, o_2] \cdot V(\tau_1(b_1, \pi_2, o_1, o_2), \tau_2(b_2, \pi_1^W, o_2, o_1))\Big] \tag{5.20}$$

$$\geq \min_{\pi_2\in\Pi_2} \Big[\mathbb{E}_{b,\pi_1^W,\pi_2}[R(s_1, s_2, a_1, a_2)] + \\ + \gamma \sum_{(o_1,o_2)\in O_1\times O_2} \Pr_{b,\pi_1^W,\pi_2}[o_1, o_2] \cdot \Big[W(\tau_1(b_1, \pi_2, o_1, o_2), \tau_2(b_2, \pi_1^W, o_2, o_1)) - C\Big]\Big] \tag{5.21}$$

$$= -\gamma C + \min_{\pi_2 \in \Pi_2} \Big[\mathbb{E}_{b,\pi_1^W,\pi_2}[R(s_1, s_2, a_1, a_2)] + \tag{5.22}$$
$$+ \gamma \sum_{(o_1,o_2) \in O_1 \times O_2} \Pr_{b,\pi_1^W,\pi_2}[o_1, o_2] \cdot \Big[W(\tau_1(b_1, \pi_2, o_1, o_2), \tau_2(b_2, \pi_1^W, o_2, o_1)) \Big] \Big]$$
$$= -\gamma C + \max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2} \Big[\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1, s_2, a_1, a_2)] + \tag{5.23}$$
$$+ \gamma \sum_{(o_1,o_2) \in O_1 \times O_2} \Pr_{b,\pi_1,\pi_2}[o_1, o_2] \cdot [W(\tau_1(b_1, \pi_2, o_1, o_2), \tau_2(b_2, \pi_1, o_2, o_1))] \Big]$$
$$= [HW](b) - \gamma C \ .$$

Equation (5.23) holds due to the fact that π_1^W is assumed to be the maximizer in $[HW](b)$. Now, we have $[HW](b) - [HV](b) \leq \gamma C$. We can use analogous reasoning to show that $[HV](b) - [HW](b) \leq \gamma C$ which shows that $|[HV](b) - [HW](b)| \leq \gamma C$ and concludes the proof. □

5.5 Heuristic Search Value Iteration for PO-POSGs

Evaluating the dynamic programming operator H directly using Definition 5.7 is impossible since the set of all joint beliefs is infinite. To design a practical algorithm, we first establish an approximation scheme for V^* that can be represented using a finite number of elements (Section 5.5.1). Then we provide mathematical programs for computing HV when this approximation scheme is used (Section 5.5.2). Finally, we state our scalable algorithm to obtain ε-approximation of $V^*(b^{\text{init}})$ in PO-POSGs in Section 5.5.3.

5.5.1 Approximating V^*

In POMDPs (or one-sided POSGs), the value function V^* is commonly represented either as a point-wise maximum over a set linear functions (termed α-vectors), or by considering a lower convex hull of a set of points (see, e.g., Section 3.6.1 for more details). Both of these approaches leverage the fact that the value function V^* is convex which is not the case for PO-POSGs where the value function is convex in the belief $b_1 \in \Delta(S_2)$ of player 1, but concave in the belief $b_2 \in \Delta(S_1)$ of player 2. In this section, we present a way to form a lower bound approximation of a convex-concave function V^* inspired by both of the above-mentioned approaches (the construction of the upper bound is analogous).

To represent the value of $V_{\text{LB}}^{\Gamma_1}$ we use a generalized notion of the value $\text{val}^{\sigma_1|b_2}$ of strategies $\sigma_1 \in \Sigma_1$ of player 1. We consider arbitrary linear functions, termed $\alpha\beta$-vectors, that form a lower bound on the value function V^*. Since the value $\text{val}^{\sigma_1|b_2}$ depends on the belief $b_2 \in \Delta(S_1)$ of the adversary, we also let $\alpha\beta$-vectors to depend on the belief β of the adversary.

Definition 5.9. An $\alpha\beta$-vector of player i is a tuple consisting of a δ-Lipschitz continuous linear function $\alpha : \Delta(S_{-i}) \to \mathbb{R}$ and the belief of the adversary $\beta \in \Delta(S_i)$ satisfying

$$\alpha(b_1) \leq V^*(b_1, \beta) \quad \text{, or} \quad \alpha(b_2) \geq V^*(\beta, b_2) \tag{5.24}$$

for player 1 or player 2, respectively.

Similarly to the way the lower bound is represented in one-sided POSGs (see Section 3.6.1 for more details), we use a finite set of $\alpha\beta$-vectors to represent the lower bound $V_{\text{LB}}^{\Gamma_1}$ on V^*. Denote $\Gamma_1 = \{\alpha_i\beta_i \mid 1 \leq i \leq k\}$ an arbitrary finite set of $\alpha\beta$-vectors of player 1. We will now discuss the construction of a lower bound $V_{\text{LB}}^{\Gamma_1}$ on V^* when considering the set of $\alpha\beta$-vectors Γ_1. First of all, observe that we can use convex combinations of $\alpha\beta$-vectors of player 1 to form lower bound V^* in the sense of the following lemma. For simplicity of the notation, we denote the set of all valid coefficients of a convex combination of k elements $\mathcal{C}^k$, and we have

$$\mathcal{C}^k = \left\{\overline{\lambda} \in \mathbb{R}^k_{\geq 0} \mid \textstyle\sum_{i=1}^k \lambda_i = 1\right\} . \tag{5.25}$$

Lemma 5.9. *Let $\overline{\lambda} \in \mathcal{C}^k$ be coefficients of convex combination, and let $(b_1, b_2) \in \Delta(S_2) \times \Delta(S_1)$ be an arbitrary joint belief. Furthermore, let $\Gamma_1 = \{\alpha_i\beta_i \mid 1 \leq i \leq k\}$ be a finite set of $\alpha\beta$-vectors of player 1. Then the following holds:*

$$V^*(b_1, b_2) \geq V^*(b_1, \textstyle\sum_{i=1}^k \lambda_i\beta_i) - \delta\|b_2 - \sum_{i=1}^k \lambda_i\beta_i\|_1 \tag{5.26a}$$

$$\geq \sum_{i=1}^k \lambda_i V^*(b_1, \beta_i) - \delta\|b_2 - \textstyle\sum_{i=1}^k \lambda_i\beta_i\|_1 \tag{5.26b}$$

$$\geq \sum_{i=1}^k \lambda_i\alpha_i(b_1) - \delta\|b_2 - \textstyle\sum_{i=1}^k \lambda_i\beta_i\|_1 . \tag{5.26c}$$

Proof. The first two inequalities follow from the fact that the optimal value function V^* is concave in the belief of the player 2 and δ-Lipschitz (see Theorem 5.3). We then use the Definition 5.9 of $\alpha\beta$-vectors of player 1 to obtain the final inequality. □

We can directly use Lemma 5.9 to define the lower bound $V_{\text{LB}}^{\Gamma_1}$ on V^* (see Figure 5.1 for illustration).

Definition 5.10. Let $\Gamma_1 = \{\alpha_i\beta_i \mid 1 \leq i \leq k\}$ be a finite set of $\alpha\beta$-vectors of player 1. The *lower bound* $V_{\text{LB}}^{\Gamma_1} : \Delta(S_2) \times \Delta(S_1)$ on V^* is a function defined by

$$V_{\text{LB}}^{\Gamma_1}(b_1, b_2) = \max_{\overline{\lambda} \in \mathcal{C}^k} \left[\sum_{i=1}^k \lambda_i\alpha_i(b_1) - \delta\|b_2 - \textstyle\sum_{i=1}^k \lambda_i\beta_i\|_1\right] . \tag{5.27}$$

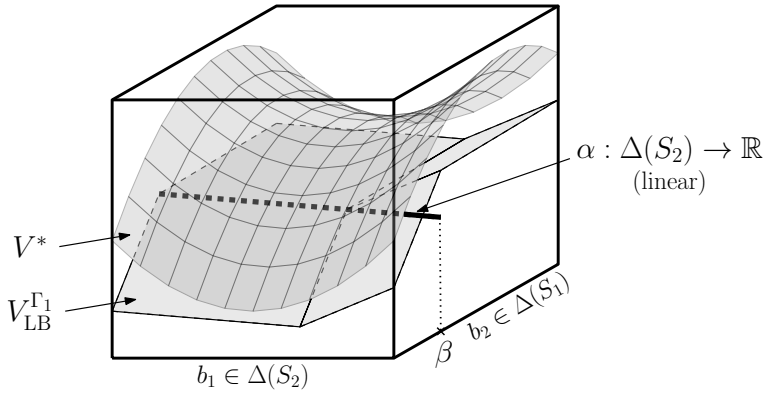


Figure 5.1: Lower bound $V_{\mathrm{LB}}^{\Gamma_1}$ on V^*. Facets of $V_{\mathrm{LB}}^{\Gamma_1}$ are formed by convex combinations of $\alpha\beta$ vectors of player 1 in Γ_1.

From the perspective of the proof of the correctness of the heuristic search value iteration algorithm for solving PO-POSGs, it is important that the lower bound $V_{\mathrm{LB}}^{\Gamma_1}$ is convex-concave and δ-Lipschitz continuous, which we prove in the next lemma.

Lemma 5.10. *Let $\Gamma_1 = \{\alpha_i\beta_i \mid 1 \leq i \leq k\}$ be a finite set of $\alpha\beta$-vectors of player 1. $V_{\mathrm{LB}}^{\Gamma_1}$ is convex and δ-Lipschitz continuous in the beliefs $\Delta(S_2)$ of player 1, and concave and δ-Lipschitz continuous in the beliefs $\Delta(S_1)$ of player 2. Furthermore, $V_{\mathrm{LB}}^{\Gamma_1}$ is δ-Lipschitz continuous.*

Proof. First, let us show that $V_{\mathrm{LB}}^{\Gamma_1}$ is concave and δ-Lipschitz continuous in the belief of player 2. Let $b_2, b_2' \in \Delta(S_1)$ be arbitrary beliefs of player 2, and $b_1 \in \Delta(S_2)$ be a fixed belief of player 1. Denote $\overline{\lambda}$ and $\overline{\lambda}'$ the coefficients of convex combination from Equation (5.27) that maximize $V_{\mathrm{LB}}^{\Gamma_1}(b_1, b_2)$ and $V_{\mathrm{LB}}^{\Gamma_1}(b_1, b_2')$, respectively, and let $\rho \in [0, 1]$ be arbitrary. Now, we show that $\rho V_{\mathrm{LB}}^{\Gamma_1}(b_1, b_2) + (1-\rho) V_{\mathrm{LB}}^{\Gamma_1}(b_1, b_2') \leq V_{\mathrm{LB}}^{\Gamma_1}(b_1, \rho b_2 + (1-\rho) b_2')$ to prove that $V_{\mathrm{LB}}^{\Gamma_1}$ is concave in the belief of player 2.

$$
\begin{aligned}
&\rho V_{\mathrm{LB}}^{\Gamma_1}(b_1, b_2) + (1-\rho) V_{\mathrm{LB}}^{\Gamma_1}(b_2, b_2') \\
&\quad = \rho \left[\sum_{i=1}^{k} \lambda_i \alpha_i(b_1) - \delta \| b_2 - \textstyle\sum_{i=1}^{k} \lambda_i \beta_i \|_1 \right] + (1-\rho) \left[\sum_{i=1}^{k} \lambda_i' \alpha_i(b_1) - \delta \| b_2' - \textstyle\sum_{i=1}^{k} \lambda_i' \beta_i \|_1 \right] \\
&\quad = \sum_{i=1}^{k} [\rho\lambda_i + (1-\rho)\lambda_i'] \alpha_i(b_1) - \delta \| \rho b_2 - \textstyle\sum_{i=1}^{k} \rho\lambda_i \beta_i \|_1 - \delta \| (1-\rho) b_2' - \textstyle\sum_{i=1}^{k} (1-\rho)\lambda_i' \beta_i \|_1 \\
&\quad \leq \sum_{i=1}^{k} [\rho\lambda_i + (1-\rho)\lambda_i'] \alpha_i(b_1) - \delta \| \rho b_2 + (1-\rho) b_2' - \textstyle\sum_{i=1}^{k} [\rho\lambda_i + (1-\rho)\lambda_i'] \beta_i \|_1 \\
&\quad \leq \max_{\overline{\lambda} \in \mathcal{C}^k} \left[\sum_{i=1}^{k} \lambda_i \alpha_i(b_1) - \delta \| \rho b_2 + (1-\rho) b_2' - \textstyle\sum_{i=1}^{k} \lambda_i \beta_i \|_1 \right] \\
&\quad = V_{\mathrm{LB}}^{\Gamma_1}(b_1, \rho b_2 + (1-\rho) b_2')
\end{aligned}
$$

We continue by proving that $V_{\mathrm{LB}}^{\Gamma_1}$ is also δ-Lipschitz continuous in the belief of player 2. Without loss of generality, let us assume that $V_{\mathrm{LB}}^{\Gamma_1}(b_1, b_2) \geq V_{\mathrm{LB}}^{\Gamma_1}(b_1, b_2')$. Assuming that $\overline{\lambda}$ is the maximizer for $V_{\mathrm{LB}}^{\Gamma_1}(b_1, b_2)$ from Equation (5.27), we have

$$\begin{aligned} V_{\mathrm{LB}}^{\Gamma_1}(b_1, b_2') &\geq \sum_{i=1}^{k} \lambda_i \alpha_i(b_1) - \delta \|b_2' - \textstyle\sum_{i=1}^{k} \lambda_i \beta_i\|_1 \\ &\geq \Big[\sum_{i=1}^{k} \lambda_i \alpha_i(b_1) - \delta \|b_2 - \textstyle\sum_{i=1}^{k} \lambda_i \beta_i\|_1 \Big] - \delta \|b_2' - b_2\|_1 \\ &= V_{\mathrm{LB}}^{\Gamma_1}(b_1, b_2) - \delta \|b_2' - b_2\|_1 \ . \end{aligned}$$

and hence $V_{\mathrm{LB}}^{\Gamma_1}$ is δ-Lipschitz continuous in the belief of player 2.

To prove that $V_{\mathrm{LB}}^{\Gamma_1}$ is convex and δ-Lipschitz continuous in the belief of player 1, let us fix an arbitrary belief $b_2 \in \Delta(S_1)$ of player 2. Observe that every coefficients $\overline{\lambda} \in \mathcal{C}^k$ of convex combination can be associated with a linear function $\alpha^{\overline{\lambda}} : \Delta(S_2) \to \mathbb{R}$ where $\alpha^{\overline{\lambda}}(b_1) = \sum_{i=1}^{k} \lambda_i \alpha_i(b_1) - \delta \|b_2 - \sum_{i=1}^{k} \lambda_i \beta_i\|_1$, i.e., the objective of Equation (5.27). We can now rewrite Equation (5.27) for the fixed belief b_2 as

$$V_{\mathrm{LB}}^{\Gamma_1}(b_1, b_2) = \max_{\overline{\lambda} \in \mathcal{C}^k} \alpha^{\overline{\lambda}}(b_1) \ .$$

According to Definition 5.9, every α_i is δ-Lipschitz continuous linear function in beliefs $b_1 \in \Delta(S_2)$ of player 1. Since $\psi = \delta \|b_2 - \sum_{i=1}^{k} \lambda_i \beta_i\|_1$ is a constant independent of b_1, $\alpha^{\overline{\lambda}}$ is δ-Lipschitz continuous linear function as well. When the belief b_2 of player 2 is fixed, $V_{\mathrm{LB}}^{\Gamma_1}$ is a point-wise maximum over δ-Lipschitz continuous linear functions $\alpha^{\overline{\lambda}}$—hence by Proposition 3.7 it is convex and δ-Lipschitz continuous in the belief of player 1. The δ-Lipschitz continuity of $V_{\mathrm{LB}}^{\Gamma_1}$ then follows from Proposition 5.2. □

By switching the roles of the players and by considering a finite set $\Gamma_2 = \{\alpha_i \beta_i \mid 1 \leq i \leq k\}$ of $\alpha\beta$-vectors of player 2, we can use analogous reasoning to derive an upper bound $V_{\mathrm{UB}}^{\Gamma_2}$ on V^*. Also, Lemma 5.10 can be adapted to show that $V_{\mathrm{UB}}^{\Gamma_2}$ is δ-Lipschitz continuous convex-concave function.

Definition 5.11. Let $\Gamma_2 = \{\alpha_i \beta_i \mid 1 \leq i \leq k\}$ be a finite set of $\alpha\beta$-vectors of player 2. The *upper bound* $V_{\mathrm{UB}}^{\Gamma_2} : \Delta(S_2) \times \Delta(S_1)$ on V^* is a function defined by

$$V_{\mathrm{UB}}^{\Gamma_2}(b_1, b_2) = \min_{\overline{\lambda} \in \mathcal{C}^k} \left[\sum_{i=1}^{k} \lambda_i \alpha_i(b_2) + \delta \|b_1 - \textstyle\sum_{i=1}^{k} \lambda_i \beta_i\|_1 \right] \ . \tag{5.28}$$

5.5.2 Evaluating Bellman's Operator

In this section, we present a linear programming formulation to solve the optimization problem $[HV](b_1, b_2)$ introduced in Definition 5.7 when the value function is represented using the technique introduced in Section 5.5.1. In this section, we focus on the evaluation

of the Bellman's operator H with respect to the lower bound $V_{\text{LB}}^{\Gamma_1}$, i.e., computation of $[HV_{\text{LB}}^{\Gamma_1}](b_1, b_2)$. Analogous reasoning is used to obtain the linear program to compute $[HV_{\text{UB}}^{\Gamma_2}](b_1, b_2)$.

Assume that $\Gamma_1 = \{\alpha_i\beta_i \mid 1 \leq i \leq k\}$ is a finite set of $\alpha\beta$-vectors of player 1 used to represent $V_{\text{LB}}^{\Gamma_1}$. Using Definition 5.7 and Definition 5.10 and since the observations are generated independently, we have

$$
\begin{aligned}
&[HV_{\text{LB}}^{\Gamma_1}](b_1, b_2) = \\
&= \max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2} \Big[\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1, s_2, a_1, a_2)] + \gamma \sum_{(o_1,o_2)} \mathbb{P}_{b,\pi_1,\pi_2}[o_1] \cdot \mathbb{P}_{b,\pi_1,\pi_2}[o_2] \cdot \qquad (5.29) \\
&\quad \cdot \max_{\lambda^{o_1,o_2} \in \mathcal{C}^k} \big[\textstyle\sum_{i=1}^k \lambda_i^{o_1,o_2} \alpha_i(\tau_1(b_1, \pi_2, o_1, o_2)) - \delta \| \tau_2(b_2, \pi_1, o_1, o_2) - \sum_{i=1}^k \lambda_i^{o_1,o_2} \beta_i \|_1 \big] \Big] \\
&= \max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2} \max_{\lambda^{o_1,o_2} \in \mathcal{C}^k \forall (o_1,o_2)} \Big[\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1, s_2, a_1, a_2)] + \gamma \sum_{(o_1,o_2)} \mathbb{P}_{b,\pi_1,\pi_2}[o_1] \cdot \mathbb{P}_{b,\pi_1,\pi_2}[o_2] \cdot \\
&\quad \cdot \big[\textstyle\sum_{i=1}^k \lambda_i^{o_1,o_2} \alpha_i(\tau_1(b_1, \pi_2, o_1, o_2)) - \delta \| \tau_2(b_2, \pi_1, o_1, o_2) - \sum_{i=1}^k \lambda_i^{o_1,o_2} \beta_i \|_1 \big] \Big] . \qquad (5.30)
\end{aligned}
$$

After expanding τ_1 according to Equation (5.1e), using that α_i are linear functions, canceling terms $\mathbb{P}_{b,\pi_1,\pi_2}[o_1]$, and rewriting $\| \cdot \|_1$ as sum of absolute values, we get

$$
\begin{aligned}
&= \max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2} \max_{\lambda^{o_1,o_2} \in \mathcal{C}^k \forall (o_1,o_2)} \Big[\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1, s_2, a_1, a_2)] + \gamma \sum_{(o_1,o_2)} \mathbb{P}_{b,\pi_1,\pi_2}[o_2] \cdot \qquad (5.31) \\
&\quad \cdot \big[\textstyle\sum_{i=1}^k \lambda_i^{o_1,o_2} \sum_{s_2,a_2,s_2'} b_1(s_2) \pi_2(a_2|s_2) Z_1(o_1|s_2, a_2) T(s_2'|s_2, a_2, o_1, o_2) \alpha_i(s_2') - \\
&\qquad - \delta \mathbb{P}_{b,\pi_1,\pi_2}[o_1] \sum_{s_1'} |\tau_2(b_2, \pi_1, o_1, o_2)(s_1') - \sum_{i=1}^k \lambda_i^{o_1,o_2} \beta_i(s_1')| \ \big] \Big] .
\end{aligned}
$$

Now, let us introduce substitution $\hat{\lambda}_i^{o_1,o_2} = \mathbb{P}_{b,\pi_1,\pi_2}[o_2] \cdot \lambda_1^{o_1,o_2}$, i.e., since $\lambda^{o_1,o_2} \in \mathcal{C}^k$ were coefficients of convex combination, $\sum_{i=1}^k \hat{\lambda}_i^{o_1,o_2} = \mathbb{P}_{b,\pi_1,\pi_2}[o_2]$ has to hold. This yields

$$
\begin{aligned}
&= \max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2} \max_{\hat{\lambda}} \Big[\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1, s_2, a_1, a_2)] + \qquad (5.32) \\
&\quad + \gamma \sum_{(o_1,o_2)} \Big[\sum_{i=1}^k \hat{\lambda}_i^{o_1,o_2} \sum_{s_2,a_2,s_2'} b_1(s_2) \pi_2(a_2|s_2) Z_1(o_1|s_2, a_2) T(s_2'|s_2, a_2, o_1, o_2) \alpha_i(s_2') - \\
&\qquad - \delta \mathbb{P}_{b,\pi_1,\pi_2}[o_1] \sum_{s_1'} |\mathbb{P}_{b,\pi_1,\pi_2}[o_2] \cdot \tau_2(b_2, \pi_1, o_1, o_2)(s_1') - \sum_{i=1}^k \hat{\lambda}_i^{o_1,o_2} \beta_i(s_1')| \ \Big] \Big] .
\end{aligned}
$$

Now, we expand $\tau_2(\cdot)$ according to Equation (5.1e) and we cancel $\mathbb{P}_{b,\pi_1,\pi_2}[o_2]$. Furthermore, we expand $\mathbb{E}_{b,\pi_1,\pi_2}[R(s_1, s_2, a_1, a_2)]$.

$$
\begin{aligned}
&= \max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2} \max_{\hat{\lambda}} \Big[\sum_{s_1,s_2,a_1,a_2} b_2(s_1) b_1(s_2) \pi_1(a_1|s_1) \pi_2(a_2|s_2) R(s_1, s_2, a_1, a_2) + \qquad (5.33)
\end{aligned}
$$

$$+\gamma \sum_{(o_1,o_2)} \Big[\sum_{i=1}^{k} \hat{\lambda}_i^{o_1,o_2} \sum_{s_2,a_2,s_2'} b_1(s_2)\pi_2(a_2|s_2)Z_1(o_1|s_2,a_2)T(s_2'|s_2,a_2,o_1,o_2)\alpha_i(s_2') -$$
$$- \delta \mathbb{P}_{b,\pi_1,\pi_2}[o_1] \sum_{s_1'} | \sum_{s_1,a_1} b_2(s_1)\pi_1(a_1|s_1)Z_2(o_2|s_1,a_1)T_1(s_1'|s_1,a_1,o_1,o_2) -$$
$$- \sum_{i=1}^{k} \hat{\lambda}_i^{o_1,o_2}\beta_i(s_1')| \Big] \Big]$$
$$\text{s.t.} \sum_{i=1}^{k} \hat{\lambda}_i^{o_1,o_2} = \mathbb{P}_{b,\pi_1,\pi_2}[o_2] \qquad \forall (o_1,o_2)$$

Since $\mathbb{P}_{b,\pi_1,\pi_2}[o_1] = \sum_{s_2,a_2} b_1(s_2)\pi_2(a_2 \mid s_2)Z_1(o_1 \mid s_2,a_2)$, the objective of the mathematical program is linear in π_2 and concave in $\hat{\lambda}$. Hence von Neumann's theorem [von Neumann, 1928; Nikaido, 1953] applies and we can reverse the order of minimization/maximization over π_2 and $\hat{\lambda}$. Furthermore, we simplify the formulation by using the fact that since the objective is linear in π_2, the solution will be found within pure stage strategies of player 2 that assign action a_2 to play in deterministic way. Next, we promote the summation over s_2 and minimize over each action a_2 of the player 2 independently. Note that $Z_1(o_1 \mid s_2,a_2)$ corresponds to the probability $\mathbb{P}_{b,\pi_1,\pi_2}[o_1 \mid s_2]$ when $\pi_2(s_2)$ deterministically assigns a_2 to play in state s_2.

$$= \max_{\pi_1 \in \Pi_1} \max_{\hat{\lambda}} \sum_{s_2} b_1(s_2) \cdot \min_{a_2} \Big[\sum_{s_1,a_1} b_2(s_1)\pi_1(a_1|s_1)R(s_1,s_2,a_1,a_2) + \tag{5.34}$$
$$+\gamma \sum_{(o_1,o_2)} \Big[\sum_{i=1}^{k} \hat{\lambda}_i^{o_1,o_2} \sum_{s_2'} Z_1(o_1|s_2,a_2)T(s_2'|s_2,a_2,o_1,o_2)\alpha_i(s_2') - \delta Z_1(o_1|s_2,a_2) \cdot$$
$$\cdot \sum_{s_1'} | \sum_{s_1,a_1} b_2(s_1)\pi_1(a_1|s_1)Z_2(o_2|s_1,a_1)T_1(s_1'|s_1,a_1,o_1,o_2) - \sum_{i=1}^{k} \hat{\lambda}_i^{o_1,o_2}\beta_i(s_1')| \Big] \Big]$$
$$\text{s.t.} \sum_{i=1}^{k} \hat{\lambda}_i^{o_1,o_2} = \mathbb{P}_{b,\pi_1,\pi_2}[o_2] \qquad \forall (o_1,o_2)$$

This formulation can be directly rewritten as a linear program by rewriting minimization over a_2 as constraints, and rewriting the absolute value $A = |x|$ as constraints $A \geq x$ and $A \geq -x$. Furthermore, we expand $\mathbb{P}_{b,\pi_1,\pi_2}[o_2]$ to $\sum_{s_1,a_1} b_2(s_1)\pi_1(a_1 \mid s_1)Z_2(o_2 \mid s_1,a_1)$. The resulting linear programming formulation LP1($HV_{\text{LB}}^{\Gamma_1}(b_1,b_2)$) to solve $[HV_{\text{LB}}^{\Gamma_1}](b_1,b_2)$ is shown in Figure 5.2. The linear program LP2($HV_{\text{UB}}^{\Gamma_2}(b_1,b_2)$) for solving the optimization problem $[HV_{\text{UB}}^{\Gamma_2}](b_1,b_2)$ can be formed analogously.

Extracting $\alpha\beta$-vectors We will now show that the variables V_1 in LP1($HV_{\text{LB}}^{\Gamma_1}(b_1,b_2)$), see Figure 5.2, can be used to form a new $\alpha\beta$-vector $\alpha_1 b_2$ of player 1. Let us fix variables V_1 to values corresponding to the solution of LP1($HV_{\text{LB}}^{\Gamma_1}(b_1,b_2)$) used to solve $[HV_{\text{LB}}^{\Gamma_1}](b_1,b_2)$, and define $\widehat{\alpha}_1(b_1) = \sum_{s_2} b_1(s_2)V_1(s_2)$ to be the value of the objective parameterized by the belief of player 1. Since the belief b_1 of player 1 occurs only in the objective of LP1($HV_{\text{LB}}^{\Gamma_1}(b_1,b_2)$), values of V_1 remain feasible even after changing the

$$\max_{\pi_1,\hat{\lambda},V_1,C,A} \sum_{s_2} b_1(s_2)\cdot V_1(s_2) \tag{5.35a}$$

$$\text{s.t. } V_1(s_2) \le \sum_{s_1,a_1} b_2(s_1)\pi_1(a_1|s_2)R(s_1,s_2,a_1,a_2) + \gamma \sum_{o_1,o_2} C^{o_1,o_2} \qquad \forall s_2,a_2 \tag{5.35b}$$

$$C^{o_1,o_2} = \sum_{i=1}^{k} \hat{\lambda}_i^{o_1,o_2} \sum_{s_2'} Z_1(o_1|s_2,a_2)T(s_2'|s_2,a_2,o_1,o_2)\alpha_i(s_2') - \qquad \forall o_1,o_2 \tag{5.35c}$$
$$- \delta Z_1(o_1|s_2,a_2)\sum_{s_1'} A^{o_1,o_2,s_1'}$$

$$A^{o_1,o_2,s_1'} \ge \sum_{s_1,a_1} b_2(s_1)\pi_1(a_1|s_1)Z_2(o_2|s_1,a_1)T_1(s_1'|s_1,a_1,o_1,o_2) - \sum_{i=1}^{k} \hat{\lambda}_i^{o_1,o_2}\beta_i(s_1')$$

$$A^{o_1,o_2,s_1'} \ge \sum_{i=1}^{k} \hat{\lambda}_i^{o_1,o_2}\beta_i(s_1') - \sum_{s_1,a_1} b_2(s_1)\pi_1(a_1|s_1)Z_2(o_2|s_1,a_1)T_1(s_1'|s_1,a_1,o_1,o_2)$$

$$\forall o_1,o_2,s_1' \tag{5.35d}$$

$$\sum_{a_1} \pi_1(a_1|s_1) = 1 \qquad \forall s_1 \tag{5.35e}$$

$$\sum_{i=1}^{k} \hat{\lambda}_i^{o_1,o_2} = \sum_{s_1,a_1} b_2(s_1)\pi_1(a_1|s_1)Z_2(o_2|s_1,a_1) \qquad \forall o_1,o_2 \tag{5.35f}$$

$$\pi_1(a_1|s_1) \ge 0 \qquad \forall s_1,a_1 \tag{5.35g}$$

$$\hat{\lambda}_i^{o_1,o_2} \ge 0 \qquad \forall o_1,o_2,1\le i\le k \tag{5.35h}$$

Figure 5.2: Linear program LP1($HV_{\text{LB}}^{\Gamma_1}(b_1,b_2)$) for solving optimization problem $[HV_{\text{LB}}^{\Gamma_1}](b_1,b_2)$

belief of player 1. We thus have that $\widehat{\alpha}_1(b'_1) \leq [HV^{\Gamma_1}_{\mathrm{LB}}](b'_1, b_2)$ for every belief $b'_1 \in \Delta(S_2)$ of player 1. Furthermore, $V^{\Gamma_1}_{\mathrm{LB}}$ is a lower bound on V^* and we have $V^{\Gamma_1}_{\mathrm{LB}} \leq V^*$ and $HV^{\Gamma_1}_{\mathrm{LB}} \leq HV^* = V^*$. Therefore also $\widehat{\alpha}_1(b'_1) \leq V^*(b'_1, b_2)$ for every b'_1, and $\widehat{\alpha}$ satisfies the condition (5.24) from Definition 5.9.

However, Definition 5.9 further requires that α is δ-Lipschitz continuous to ensure that $V^{\Gamma_1}_{\mathrm{LB}}$ is δ-Lipschitz continuous (Lemma 5.10). Recall that the value $\mathrm{val}^{\sigma_1|b_2}$ of every strategy $\sigma_1 \in \Sigma_1$ of player 1 is bounded by L and U. We can thus truncate values $V_1(s_2)$ that are smaller than L, and define the desired linear function α_1 using the values in the vertices of the $\Delta(S_2)$ simplex as $\alpha_1(s_2) = \max\{L, V_1(s_2)\}$. We now have that $L \leq \alpha_1(s_2) \leq U$ and α_1 is δ-Lipschitz continuous by Lemma 3.4. Hence $\alpha_1 b_2$ is the desired $\alpha\beta$-vector of player 1. Note that we have that $\alpha_1(b_1) \geq \widehat{\alpha}_1(b_1) = [HV^{\Gamma_1}_{\mathrm{LB}}](b_1, b_2)$. Hence after we perform a point-based update and set $\Gamma'_1 = \Gamma_1 \cup \{\alpha_1 b_2\}$, we have that $V^{\Gamma'_1}_{\mathrm{LB}}(b_1, b_2) \geq [HV^{\Gamma_1}_{\mathrm{LB}}](b_1, b_2)$.

Analogous reasoning can be used to obtain $\alpha\beta$-vector of player 2 from the linear program $\mathrm{LP2}(HV^{\Gamma_2}_{\mathrm{UB}}(b_1, b_2))$ that is formed analogously to LP1. Here, the objective is $\sum_{s_1} b_2(s_1)V_2(s_1)$, and $\alpha_2 b_1$ for $\alpha_2(s_1) = \min\{U, V_2(s_1)\}$ is the desired $\alpha\beta$-vector of player 2.

Computing strategy of adversary In order to compute strategy of player 2 from $\mathrm{LP1}(HV^{\Gamma_1}_{\mathrm{LB}}(b_1, b_2))$, it suffices to consider dual variables to constraints (5.35b). Here, the value z^{s_2,a_2} of the dual variable associated to the constraint (s_2, a_2) corresponds to the joint probability that action a_2 is to be played in state s_2, i.e., $\pi_2(a_2 \mid s_2) = z^{s_2,a_2}/b_1(s_2)$ if $b_1(s_2) > 0$. Similarly, we can compute strategy of player 1 from $\mathrm{LP2}(HV^{\Gamma_2}_{\mathrm{UB}}(b_1, b_2))$.

5.5.3 The Algorithm

We are now ready to state our algorithm to compute an ε-approximation of V^* in the joint belief $(b_1^{\mathrm{init}}, b_2^{\mathrm{init}})$ and to prove its correctness. The algorithm (Algorithm 4.1) follows the ideas of the HSVI algorithm for POMDPs [Smith and Simmons, 2004, 2005] and one-sided POSGs (Section 3.6) while replacing the point-based update step with the computation of optimal $\alpha\beta$-vectors to add using the linear program from Figure 5.2.

Algorithm 5.1: HSVI algorithm for discounted PO-POSGs.

Initialize $V^{\Gamma_1}_{\mathrm{LB}}$ and $V^{\Gamma_2}_{\mathrm{UB}}$
while $\mathrm{excess}_0(b_1^{\mathrm{init}}, b_2^{\mathrm{init}}) > 0$ **do** `explore`$((b_1^{\mathrm{init}}, b_2^{\mathrm{init}}), 0)$
procedure `explore`$(b^t = (b_1^t, b_2^t), t)$
  Extract π_1^{UB} from $\mathrm{LP2}(HV^{\Gamma_2}_{\mathrm{UB}}(b^t))$ and π_2^{LB} from $\mathrm{LP1}(HV^{\Gamma_1}_{\mathrm{LB}}(b^t))$
  $\text{w-excess}(o_1, o_2) = \mathbb{P}_{b^t, \pi_1^{\mathrm{UB}}, \pi_2^{\mathrm{LB}}}[o_1, o_2] \cdot \mathrm{excess}_{t+1}(\tau_1(b_1^t, \pi_2^{\mathrm{LB}}, o_1, o_2), \tau_2(b_2^t, \pi_1^{\mathrm{UB}}, o_2, o_1))$
  $(o_1^*, o_2^*) \leftarrow \arg\max_{(o_1,o_2)} \text{w-excess}(o_1, o_2)$
  if $\text{w-excess}(o_1^*, o_2^*) > 0$ **then**
    `explore`$((\tau_1(b_1^t, \pi_2^{\mathrm{LB}}, o_1^*, o_2^*), \tau_2(b_2^t, \pi_1^{\mathrm{UB}}, o_2^*, o_1^*)), t+1)$
  Extract $\alpha_1 b_2^t$ from $\mathrm{LP1}(HV^{\Gamma_1}_{\mathrm{LB}}(b^t))$ (see Section 5.5.2 for more details)
  Extract $\alpha_2 b_1^t$ from $\mathrm{LP2}(HV^{\Gamma_2}_{\mathrm{UB}}(b^t))$ (see Section 5.5.2 for more details)
  $\Gamma_1 \leftarrow \Gamma_1 \cup \{\alpha_1 b_2^t\}$; $\Gamma_2 \leftarrow \Gamma_2 \cup \{\alpha_2 b_1^t\}$

Since we want to focus on the key characteristics of the algorithm, we initialize $V_{\mathrm{LB}}^{\Gamma_1}$ and $V_{\mathrm{UB}}^{\Gamma_2}$ on line 1 of Algorithm 4.1 using the minimum and maximum possible utilities of player 1,

$$L = \min_{s_1,s_2,a_1,a_2} R(s_1,s_2,a_1,a_2)/(1-\gamma) \tag{5.36}$$

$$U = \max_{s_1,s_2,a_1,a_2} R(s_1,s_2,a_1,a_2)/(1-\gamma)\ . \tag{5.37}$$

To initialize the lower bound, we form the initial set Γ_1 that is used to represent $V_{\mathrm{LB}}^{\Gamma_1}$ by considering one $\alpha\beta$-vector of player 1 for every state $s_1 \in S_1$ of player 1. Namely,

$$\Gamma_1 = \{\alpha b_2^{s_1} \mid s_1 \in S_1\} \qquad \alpha(b_1) = L \qquad b_2^{s_1}(s_1') = \begin{cases} 1 & s_1 = s_1' \\ 0 & \text{otherwise} \end{cases}\ . \tag{5.38}$$

We obtain Γ_2 to form initial $V_{\mathrm{UB}}^{\Gamma_2}$ similarly by forming $\alpha\beta$-vectors of player 2 for every $s_2 \in S_2$ and using U as the utility. In practice, we can obtain tighter bounds (and consequently faster convergence) by either leveraging domain knowledge, or solving a simplified version of the game (similarly to the initialization of bounds in the HSVI algorithm for one-sided POSGs, see Section 3.6.1).

The remaining structure of the algorithm follows the structure of the HSVI algorithm for one-sided POSGs (Section 3.6). Similarly as in one-sided POSGs, we run the algorithm as long as the excess gap $\mathrm{excess}_0(b_1^{\mathrm{init}}, b_2^{\mathrm{init}})$ in the initial belief is positive. The excess gap is defined identically to Equation (3.52), except for using value functions $V_{\mathrm{LB}}^{\Gamma_1}$ and $V_{\mathrm{UB}}^{\Gamma_2}$,

$$\begin{aligned} &\mathrm{excess}_t(b_1,b_2) = V_{\mathrm{UB}}^{\Gamma_2}(b_1,b_2) - V_{\mathrm{LB}}^{\Gamma_1}(b_1,b_2) - \rho(t)\ \text{, where} \\ &\rho(0) = \varepsilon \qquad\qquad \rho(t+1) = [\rho(t) - 2\delta D]/\gamma\ . \end{aligned} \tag{5.39}$$

In every call to the `explore` procedure, we first compute the optimistic strategies of the players (the maximizing player 1 obtains the strategy π_1^{UB} from the overestimating value function $V_{\mathrm{UB}}^{\Gamma_2}$, while the minimizing player 2 obtains the strategy π_2^{LB} from the underestimating value function $V_{\mathrm{LB}}^{\Gamma_1}$). Then the joint belief $(\tau_1(b_1^t, \pi_2^{\mathrm{LB}}, o_1, o_2), \tau_2(b_2^t, \pi_1^{\mathrm{UB}}, o_1, o_2))$ with the highest weighted excess gap, denoted w-excess, is targeted. Finally, a point based update in the joint belief (b_1^t, b_2^t) is performed by extracting $\alpha\beta$-vectors of both players from the solutions of $\mathrm{LP1}([HV_{\mathrm{LB}}^{\Gamma_1}](b^t))$ and $\mathrm{LP2}([HV_{\mathrm{UB}}^{\Gamma_2}](b^t))$. These $\alpha\beta$-vectors are then used to update sets Γ_1 and Γ_2, and thus to refine bounds $V_{\mathrm{LB}}^{\Gamma_1}$ and $V_{\mathrm{UB}}^{\Gamma_2}$.

We will now prove that the algorithm is correct, i.e., it terminates with valid bounds $V_{\mathrm{LB}}^{\Gamma_1}$ and $V_{\mathrm{UB}}^{\Gamma_2}$ on V^* (this is a consequence of Lemma 5.9), and it holds that $V_{\mathrm{UB}}^{\Gamma_2}(b_1^{\mathrm{init}}, b_2^{\mathrm{init}}) - V_{\mathrm{LB}}^{\Gamma_1}(b_1^{\mathrm{init}}, b_2^{\mathrm{init}}) \leq \varepsilon$. The proof is analogous to the proof of Theorem 3.25 showing the correctness of the HSVI algorithm for solving one-sided POSGs.

Theorem 5.11. *Algorithm 5.1 terminates with an ε-approximation of $V^*(b_1^{\mathrm{init}}, b_2^{\mathrm{init}})$.*

Proof (sketch). The proof is closely similar to the proof of Theorem 3.25 showing the correctness of the HSVI algorithm for one-sided POSGs and relies on the fact that $V_{\mathrm{LB}}^{\Gamma_1}$ and $V_{\mathrm{UB}}^{\Gamma_2}$ are δ-Lipschitz continuous bounds on V^* (Lemma 5.10). Assume for the sake of contradiction that the algorithm does not terminate and generates an infinite number of `explore` trials. Since the length of a trial is bounded by a finite number $T_{\max}$[1], the number of trials of length T (for some $0 \leq T \leq T_{\max}$) must be infinite. The set $\Delta(S_2) \times \Delta(S_1)$ is compact and hence also totally bounded. It is therefore impossible to fit an infinite number of belief points (b_1, b_2) satisfying $\|(b_1, b_2) - (b'_1, b'_2)\|_2 > D$ within $\Delta(S_2) \times \Delta(S_1)$. Hence there must be two trials of length T, $\{(b_{11}^{(t)}, b_{21}^{(t)})\}_{t=0}^{T}$ and $\{(b_{12}^{(t)}, b_{22}^{(t)})\}_{t=0}^{T}$, such that $\|(b_{11}^{(T)}, b_{21}^{(T)}) - (b_{12}^{(T)}, b_{22}^{(T)})\|_2 \leq D$. Without loss of generality, assume that $(b_{11}^{(T)}, b_{21}^{(T)})$ was visited the first. Applying Lemma 5.7, we can use a similar reasoning to the proof of Lemma 3.24 to obtain that $\mathrm{excess}_T(b_{11}^{(T)}, b_{21}^{(T)}) \leq -2\delta D$ after the point-based update in $(b_{11}^{(T)}, b_{21}^{(T)})$ is performed. Since $V_{\mathrm{LB}}^{\Gamma_1}$ and $V_{\mathrm{UB}}^{\Gamma_2}$ are δ-Lipschitz continuous (Lemma 5.10) and $V_{\mathrm{UB}}^{\Gamma_2} - V_{\mathrm{LB}}^{\Gamma_1}$ is thus 2δ-Lipschitz continuous, we also have $\mathrm{excess}_T(b_{12}^{(T)}, b_{22}^{(T)}) \leq 0$. Based on the condition on line 7 of Algorithm 5.1, this contradicts that $(b_{12}^{(T)}, b_{22}^{(T)})$ was selected by the algorithm. □

5.5.4 Implementation Details

In this section, we provide some of the details on our practical implementation of the HSVI algorithm for PO-POSGs.

Pruning The number of $\alpha\beta$-vectors grows in the course of the algorithm, however, not all of the vectors are needed to represent $V_{\mathrm{LB}}^{\Gamma_1}$ (or $V_{\mathrm{UB}}^{\Gamma_2}$) accurately. To counteract this growth, we run a pruning procedure every time the size of Γ_i gets $1.5\times$ larger than after the pruning was last performed. An $\alpha\beta$-vector is pruned if there exists a convex combination of vectors in Γ_i that dominates it.

Lipschitz continuity The theoretical proof of the correctness of the algorithm relies on the fact that the approximating functions are δ-Lipschitz continuous. In the implementation, we use a simpler representation of $V_{\mathrm{LB}}^{\Gamma_1}$,

$$V_{\mathrm{LB}}^{\Gamma_1}(b_1, b_2) = \max_{\overline{\lambda} \in \mathcal{C}^k} \left\{ \sum_{i=1}^{k} \lambda_i \alpha_i(b_1) \mid \sum_{i=1}^{k} \lambda_i \beta_i = b_2 \right\} . \tag{5.40}$$

This formulation does not necessarily form a δ-Lipschitz continuous function in the beliefs $b_2 \in \Delta(S_1)$ of player 2. However, despite breaking the assumption of δ-Lipschitz continuity of $V_{\mathrm{LB}}^{\Gamma_1}$ and $V_{\mathrm{UB}}^{\Gamma_2}$, we did not experience any convergence issues in the experiments.

Other Similarly to our previous work on one-sided POSGs (Chapter 3), we use the idea of modifying ε between iterations. We get $\varepsilon_{\mathrm{imm}}$ for the current iteration as $\varepsilon_{\mathrm{imm}} = \varepsilon + 0.5(V_{\mathrm{UB}}^{\Gamma_2}(b^{\mathrm{init}}) - V_{\mathrm{LB}}^{\Gamma_1}(b^{\mathrm{init}}) - \varepsilon)$. This allows the algorithm to perform shorter trials in

[1] This is again caused by the fact that $V_{\mathrm{LB}}^{\Gamma_1} \geq L$ and $V_{\mathrm{UB}}^{\Gamma_2} \leq U$

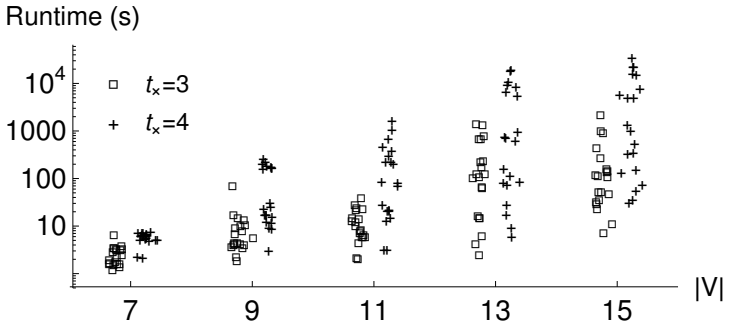


Figure 5.3: Experimental results on the Patrolling domain for different sizes of graph $|V|$. Time to reach $V^{\Gamma_2}_{\mathrm{UB}}(b^{\mathrm{init}}) - V^{\Gamma_2}_{\mathrm{LB}}(b^{\mathrm{init}}) \leq 1$.

the initial phases of the search (when the bounds do not provide accurate information about what parts of the belief space to target).

We construct a compact version of linear programs discussed in Section 5.5.2. Namely, we consider only states, actions and observation pairs that can be played/observed in the current joint belief (b_1, b_2). Furthermore, we adopt a column generation approach to incrementally add variables $\hat{\lambda}^{o_i,o_j}(\cdot)$. Initially, we start with one $\alpha\beta$-vector (and its $\hat{\lambda}^{o_i,o_j}(\alpha\beta)$) for each pure belief of the opponent and we add additional $\alpha\beta$-vectors once they are necessary to accurately represent $V(\tau_1(b_1, \pi_2, o_1, o_2), \tau_2(b_2, \pi_1, o_2, o_1))$.

5.6 Experiments

We demonstrate the scalability of our algorithm on two fundamentally different domains—partially observable patrolling inspired by [Basilico et al., 2009b] and a lasertag game inspired by *Tag* from [Pineau et al., 2003]. All experiments use discount factor $\gamma = 0.95$ and were run on Intel i7-8700K (solving 6 instances in parallel).

Patrolling The game is played by two players—the *patroller* and the *intruder*. The patroller moves between vertices V of a graph $G = (V, E)$ and attempts to locate an intruder before the intruder succeeds in causing damage. The intruder starts initially outside of the graph and observes the position of the patroller whenever he steps on one of the observable vertices $O \subseteq V$ (otherwise the position of the patroller remains hidden). The intruder may decide to attack any target vertex $v \in T$, $T \subseteq O$. Once the intruder decides to attack, he has to stay undetected in the chosen vertex v for $t_\times$ time steps to complete his attack and get a reward $c(v)$.

In our experimental evaluation, we consider $t_\times = 3$ and $t_\times = 4$ and generate random graphs from the Dorogovtsev-Mendes model such that the shortest cycle covering all targets is longer than $t_\times$ (i.e., the patroller cannot cover the targets perfectly). There are $|T| = \lceil V/4 \rceil$ targets and $|O| = \lceil 2V/3 \rceil$ observable nodes. The costs $c(v)$ of targets are generated uniformly from the $[70, 100]$ interval. Figure 5.3 summarizes the runtime of our algorithm on 200 randomly generated instances of Patrolling (time to reach precision 1, i.e., 1% of the maximum cost, is reported). All instances have been solved within 10

hours, while 97 instances with $t_\times = 3$ out of 100 and 82 instances with $t_\times = 4$ out of 100 have been solved in less than 20 minutes.

Lasertag The game is played by two players—the *tagger* and the *evader*—on a grid. In each time step, the players can decide to move to an adjacent square (free of an obstacle), or, the tagger can additionally shoot a laser beam either horizontally or vertically (which is effective until hitting the first obstacle). If the beam tags the evader, the tagger receives a reward $+10$ and the game ends, otherwise his reward is -10 and the game continues. Unless the tagger decides to use the laser beam, his reward is -1 in each step. Hence, the tagger attempts to terminate the game by tagging the adversary as quickly as possible. Neither player knows the position of each other until the tagger decides to shoot. In such a case the evader can observe the light ray (and thus deduce possible positions of the tagger).

We consider lasertag games played on a 4×4 grid with 3 obstacles where the tagger starts in the top-left corner, while the evader starts at position $(3, 4)$ next to the opposite corner. The obstacles are placed randomly while guaranteeing the existence of a path between the players (we discard symmetrical instances). We ran the algorithm with $\varepsilon = 0.05$ for 5 hours. While the algorithm did not terminate within this limit on 16 out of 20 instances, the average excess gap in the initial belief relative to the value of the lower bound was $10\% \pm 2.6\%$ (where the confidence interval marks standard error). For grid size 3×3, all non-symmetric instances with players starting in opposite corners have been solved in less than 8 seconds.

Analysis We provide a detailed analysis of the performance of the algorithm for two instances of patrolling, an 11-vertex instance with $t_\times = 4$ solved in 307s and a 13-vertex instance with $t_\times = 4$ solved in 11004s. On both of the instances, 85% of the runtime corresponds to the operations with the approximating functions (especially computing values of $V_{\mathrm{LB}}^{\Gamma_1}$ and $V_{\mathrm{UB}}^{\Gamma_2}$ in a given joint belief), while the construction and solving LP$i(HV(b_1, b_2))$ took only 10% of the runtime. The remaining 5% of the runtime corresponds to the pruning step, initiated 95 times on the larger instance within the 1556 iterations. The pruning eliminated 22126 $\alpha\beta$-vectors out of 50404 generated on the larger instance. Unlike in the patrolling domain, on a lasertag instance solved in 9337s the pruning was much more frequent (approximately one execution of the pruning procedure per 6 iterations of the algorithm) and considerably more demanding (took 22% of runtime).

CHAPTER 6

Beyond Discounted Sum

Although the discounted-sum objective is one of the most commonly studied objectives in the AI literature, it makes modeling of some real-world problems problematic or even impossible. Since the sequence of weights $\{\gamma^{t-1}\}_{t=1}^{\infty}$ is decreasing, the rewards obtained early in the game are assumed to be of greater significance, compared to the rewards obtained at later stages.

Consider a robot planning example, where the goal is to minimize the amount of consumed energy until the mission is completed. Here, we can assign an execution cost to each of the actions the robot can perform (corresponding to the amount of energy consumed while performing the given action), and the goal is to optimize the *undiscounted* sum of costs. Observe that approximating the problem using the discounted-sum objective (for arbitrary discount factor $\gamma < 1$) can have undesirable results. For example, from the perspective of minimizing the discounted-sum of costs, it can be beneficial to wait for a long time before accomplishing the mission. Even though the undiscounted sum of the costs of the actions that lead to the mission completion is the same (or may even increase), by waiting for t steps, the discounted sum of the costs is decreased by a factor γ^t.

The theoretical guarantees of the algorithms for solving one-sided POSGs and POSGs with public observations presented in Chapters 3 and 5, however, apply to the discounted-sum objective only. The introduction of the discount factor allowed us to establish, e.g., the following two fundamental properties: Lipschitz continuity of the optimal value functions V^* (Theorem 3.6), and contractivity of the Bellman's operator used to solve these games (Theorem 3.18).

In this chapter, we study the model of Goal-POMDPs and analyze the applicability of the heuristic search value iteration algorithm (HSVI) [Smith and Simmons, 2004, 2005]

This chapter is based on following publications:

[Horák et al., 2018] Horák, K., Bošanský, B., and Chatterjee, K. (2018). Goal-HSVI: Heuristic Search Value Iteration for Goal POMDPs. In *27th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4764–4770 (45%)

to these models. The objective in Goal-POMDPs is to minimize the infinite sum of *undiscounted* costs. We provide an extended analysis of the convergence of the HSVI algorithm when applied to Goal-POMDPs, and we illustrate the key challenges when applying HSVI to undiscounted problems. Based on these insights, we propose a novel algorithm called *Goal-HSVI* that provably converges to ε-optimal solution in the Goal-POMDP setting. Since our methods for solving OS-POSGs and PO-POSGs are based on the heuristic search value iteration algorithm, we believe that this analysis is the first step towards applying these methods to solve game-theoretic problems where no discounting is considered.

6.1 Goal-POMDPs

Apart of the standard discounted-sum objective, *indefinite-horizon* objective is another classical and widely studied objective for MDPs and POMDPs [Bertsekas and Tsitsiklis, 1996; Patek, 2001; Bonet and Geffner, 2009; Kolobov et al., 2011; Chatterjee et al., 2016], often under the name of Goal-POMDPs. In this case, there is a set of target states, all positive costs, and the goal is to minimize the expected total cost till the target set is reached. Note that this objective is not discounted-sum but a total sum without discounts. The objective is also not finite-horizon, as there is no a priori bound on the time when the target set is reached.

In contrast to POMDPs with discounted-sum objectives, to the best of our knowledge, there does not exist a scalable method for solving Goal-POMDPs that provably converges to a near-optimal policy. The algorithms used previously in practice are RTDP-Bel [Bonet, 1998; Bonet and Geffner, 2009], and also heuristic search value iteration (HSVI) algorithm [Smith and Simmons, 2004, 2005] that was used for solving Goal-POMDPs in [Warnquist et al., 2013]. However, none of these algorithms guarantee convergence, and HSVI need not even work for Goal-POMDPs in general [Smith, 2007, Theorem 6.9]. For the overview of both of these algorithms, see Section 2.2.

Our contributions We extend the discussion on the convergence of HSVI algorithm when applied to Goal-POMDPs, and we illuminate the key issues of HSVI on counter-examples. We address these issues and, based on our insights, we present a novel *Goal-HSVI* algorithm for solving Goal-POMDPs. Goal-HSVI is an advancement over previous approaches from the theoretical as well as practical perspective: (1) From the theoretical perspective, Goal-HSVI provides upper and lower bounds on the optimal value (and the quality of the currently considered policy) at all points of time and these bounds converge. Thus we provide the first algorithm with a theoretical guarantee of convergence for Goal-POMDPs, and our algorithm provides an *anytime* approximation. (2) From the practical perspective, we present an implementation of our algorithm and experimental results on several classical POMDP examples from the literature. While Goal-HSVI is comparable to RTDP-Bel on the RockSample domain, it dramatically outperforms RTDP-Bel on several other domains.

Goal-POMDPs We use the notation from [Bonet and Geffner, 2009] and define a Goal-POMDP as a tuple $\langle S, G, A, O, P, Q, c, b^{\text{init}} \rangle$, where S is a finite non-empty set of states, G is a non-empty set of target (or goal) states ($G \subseteq S$), A is a finite non-empty set of actions, O is a finite non-empty set of observations ($o_g \in O$ notifies the agent about reaching the goal), $P_a(s, s')$ is the probability to transition from s to s' by using action a, $Q_a(o|s')$ is the probability to observe o when entering state s' by using action a, $c(s, a) > 0$ is the cost for taking action a in a non-target state s and $b^{\text{init}} \in \Delta(S)$ is the initial belief. Without loss of generality, we assume $G = \{g\}$.

We assume that the agent does not incur any cost after reaching the target state g, i.e. $c(g, a) = 0$ for every $a \in A$. Moreover, state g is absorbing, i.e., $P_a(g, g) = 1$ for every $a \in A$, and the agent is always certain about reaching g (i.e., $Q_a(o_g|g) = 1$ for every $a \in A$ and $Q_a(o_g|s) = 0$ for every $s \neq g$). We also assume that the goal state is reachable from every non-target state, i.e., the agent can never enter a dead-end. This requirement can, however, be overcome by precomputing a set of allowed actions for each belief support [Chatterjee et al., 2016]. The algorithms for Goal-POMDPs can, therefore, be easily extended to the problems with dead-ends by considering only actions that are allowed in the current belief (and thus avoid dead-ends). Note that the assumption of positive costs is, however, essential for the approximability of the problem as allowing negative costs renders any approximation undecidable [Chatterjee et al., 2016, Theorem 2].

Note that this formulation of Goal-POMDPs is analogous to the formulation of POMDPs from Section 2.2. Namely, we can obtain POMDP from Definition 2.7 as a tuple (S, A, O, T, R), where $T(o, s' \mid s, a) = P_a(s, s') \cdot Q_a(o \mid s')$ and $R(s, a) = -c(s, a)$. The discussion from Section 2.2 on solution techniques therefore applies.

6.2 Vanilla-HSVI and Goal-POMDPs

While HSVI2 [Smith and Simmons, 2005] (termed *vanilla-HSVI* for our purposes) was applied in Goal-POMDPs [Warnquist et al., 2013], it loses its desirable theoretical guarantees as shown already by Smith [2007]. We discuss three key issues related to the algorithm in the Goal-POMDP setting and illustrate them on examples.

Since we use a different notation to describe POMDPs compared to Section 2.2, we first restate the HSVI2 algorithm using this notation (Algorithm 6.2). Notice that the key difference is that in Goal-POMDPs, we are minimizing costs instead of maximizing the rewards. Hence, the value functions are concave (instead of convex when maximization of rewards is considered). Furthermore, also the role of the bounds in the vanilla-HSVI algorithm is reversed. Here, the lower bound on cost V^{Υ}_{LB} is concave and uses the sawtooth representation (see Figure 2.1c for the convex counterpart). On the other hand, the upper bound on cost V^{Γ}_{UB} is represented as a point-wise *minimum* over α-vectors. We also omitted discount factor γ from the pseudocode since Goal-POMDPs are undiscounted (i.e., $\gamma = 1$).

Algorithm 6.1: Point-based `update`(b) procedure of Vanilla-HSVI.

1 $\alpha_a^o \leftarrow \arg\min_{\alpha\in\Gamma} \sum_{s'} \tau(b,a,o)(s') \cdot \alpha(s')\rangle$ for all $a \in A, o \in O$
2 $\alpha_a(s) \leftarrow c(s,a) + \sum_{o,s'} \mathbb{P}_a(o,s'|s) \cdot \alpha_a^o(s')$ $\quad \forall s, a$
3 $\Gamma \leftarrow \Gamma \cup \{\arg\min_{\alpha_a} \sum_s b(s) \cdot \alpha_a(s)\rangle\}$
4 $\Upsilon \leftarrow \Upsilon \cup \{(b, \min_a \left[\sum_s b(s)c(s,a) + \gamma \sum_o \mathbb{P}_b[o|a] \cdot V_{\mathrm{LB}}^{\Upsilon}(\tau(b,a,o))\right])\}$

Algorithm 6.2: Vanilla-HSVI (HSVI2 applied to Goal-POMDPs). The pseudo-code follows the ZMDP implementation and includes `update` on line 6.

1 Initialize $V_{\mathrm{LB}}^{\Upsilon}$ and V_{UB}^{Γ}
2 **while** $V_{\mathrm{UB}}^{\Gamma}(b^{\mathrm{init}}) - V_{\mathrm{LB}}^{\Upsilon}(b^{\mathrm{init}}) > \varepsilon$ **do** `explore`$(b^{\mathrm{init}}, \varepsilon, 0)$
3 **procedure** `explore`(b, ε, t)
4 $\quad$ **if** $V_{\mathrm{UB}}^{\Gamma}(b) - V_{\mathrm{LB}}^{\Upsilon}(b) \leq \varepsilon$ **then return**
5 $\quad a^* \leftarrow \arg\min_a \left[\sum_s b(s)c(s,a) + \gamma \sum_o \mathbb{P}_b[o|a] V_{\mathrm{LB}}^{\Upsilon}(\tau(b,a,o))\right]$
6 $\quad$ `update`(b)
7 $\quad o^* \leftarrow \arg\max_o \mathbb{P}_b[o|a^*] \cdot [V_{\mathrm{UB}}^{\Gamma}(\tau(b,a^*,o)) - V_{\mathrm{LB}}^{\Upsilon}(\tau(b,a^*,o)) - \varepsilon]$
8 $\quad$ `explore`$(\tau(b,a^*,o^*), \varepsilon, t+1)$
9 $\quad$ `update`(b)

Initial values can be infinite. Vanilla-HSVI initializes value function V_{UB}^{Γ} by considering the values of a blind policy (i.e., a policy prescribing the agent to use a fixed action a forever). Such policies, however, need not reach the goal with probability 1. Observe that in the example from Figure 6.1 the policies 'play a forever' and 'play b forever' *never* reach the goal and their cost is thus infinite. Moreover, since the play stays in s_1 with positive probability, $V_{\mathrm{UB}}^{\Gamma}(s_1)$ remains infinite forever.

Solution: Instead of blind policies, we initialize V_{UB}^{Γ} using the uniform policy. Since the goal state g is reachable from *every* state, the uniform policy reaches g with probability 1 and thus it has finite values [Chatterjee et al., 2016, Lemma 5 and 6].

Exploration need not terminate. In discounted-sum problems, the sequence $\varepsilon\gamma^{-t}$ is strictly increasing and unbounded (since $0 < \gamma < 1$). Its value therefore eventually exceeds the gap $V_{\mathrm{UB}}^{\Gamma}(b) - V_{\mathrm{LB}}^{\Upsilon}(b)$ (which is guaranteed by the initialization to be bounded) and the recursive `explore` procedure of vanilla-HSVI terminates. This is clearly not the case in Goal-POMDPs where $\gamma = 1$. In [Smith, 2007, Theorem 6.9], it has been shown that the observation-selection heuristic of vanilla-HSVI (HSVI2) may cause the algorithm to enter an infinite loop. We show that even the action selection (line 5 of Algorithm 6.2)

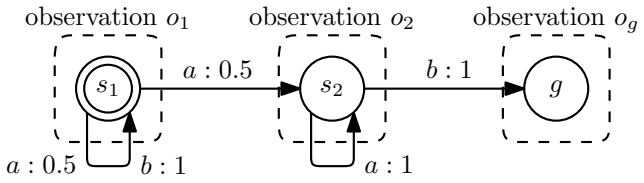
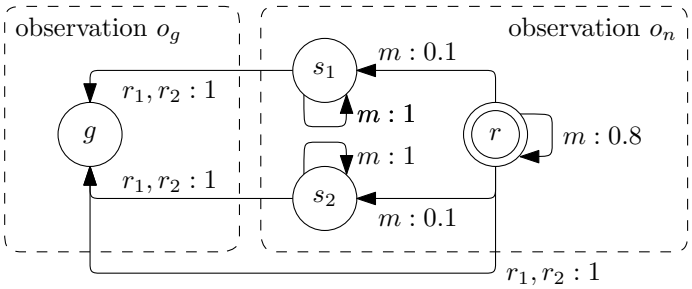


Figure 6.1: Blind policies have infinite values, $b^{\mathrm{init}}(s_1) = 1$.



$c(r, r_i) = 50$, $c(s_1, r_2) = c(s_2, r_1) = 100$, other costs in non-goals 1

Figure 6.2: Goal-POMDP where the `explore` procedure of vanilla-HSVI does not terminate, r_i must be played to reach g, $b^{\text{init}}(r) = 1$. Edges are labeled with actions and respective transition probabilities when using the given action.

is susceptible to this behavior (and thus modifying the observation-selection heuristic does not fix the algorithm).

Consider the Goal-POMDP shown in Figure 6.2 where r is the initial state. The target state g can only be reached by playing action r_1 or r_2 at some point. We show, however, that the way the value function V_{LB}^{Υ} is updated and beliefs are changed during the `explore` recursion prevents these actions to be ever considered. First, observe that the only reachable non-goal beliefs in this POMDP are b^T, where $T \in \mathbb{Z}_0^+$ and

$$b^T(r) = 0.8^T \qquad b^T(s_1) = b^T(s_2) = (1 - 0.8^T)/2 \ . \tag{6.1}$$

Furthermore, since the goal state g is reached with probability 1 when using action r_i, the lower bound on the cost of playing r_i in b^T (i.e., the objective of argmin on line 5 of Algorithm 6.2), denoted $\underline{v}^{r_i}(b^T)$, is independent of T and constant.

$$\begin{aligned} \underline{v}^{r_i}(b^T) &= V_{\text{LB}}^{\Upsilon}(\tau(b^t, r_i, o_g)) + b^T(r)c(r, r_i) + \\ &\quad + \ b^T(s_i)c(s_i, r_i) + b^T(s_{-i}, r_i)c(s_{-i}, r_i) \\ &= 0 + 0.8^T \cdot 50 + (1 - 0.8^T)/2 \cdot 1 + (1 - 0.8^T)/2 \cdot 99 = 50 \ . \end{aligned} \tag{6.2}$$

This means that the `explore` procedure selects r_i (line 5 of Algorithm 6.2) in b^T only if the lower bound on playing m in b^T, $\underline{v}^m(b^T) = V_{\text{LB}}^{\Upsilon}(b^{T+1}) + c(*, m)$ is greater than the lower bound on the cost of playing r_i, $\underline{v}^{r_i}(b^T) = 50$. We show that this never happens during the trial.

Denote b^∞ the limit belief of the sequence $\{b^T\}_{T=0}^{\infty}$, i.e., $b^\infty(s_1) = b^\infty(s_2) = 0.5$. The recursion starts with a linear V_{LB}^{Υ} where $V_{\text{LB}}^{\Upsilon}(b^\infty) = 1$ and $V_{\text{LB}}^{\Upsilon}(b^0) = 6$ (as initialized by the fast informed bound). In this situation, value $V_{\text{LB}}^{\Upsilon}(b^1)$ is a convex combination $0.8V_{\text{LB}}^{\Upsilon}(b^0) + 0.2V_{\text{LB}}^{\Upsilon}(b^\infty)$ and action m is optimal in b^0, as its value in b^0 is

$$\underline{v}^m(b^0) = V_{\text{LB}}^{\Upsilon}(b^1) + c(*, m) = 5 + 1 = 6 = y^0 \tag{6.3}$$

and the addition of the point (b^0, y^0) to the set Υ does not change V_{LB}^{Υ}.

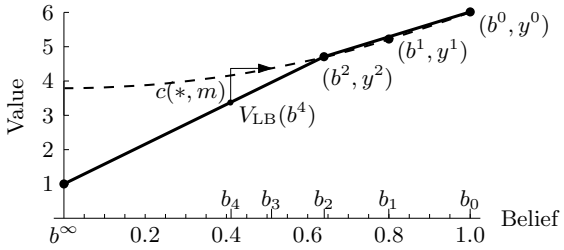


Figure 6.3: Value function $V^{\Upsilon}_{\mathrm{LB}}$ for the example from Figure 6.2 in the third level of `explore` recursion.

At the T-th level of `explore` recursion, set Υ contains $(b^0, y^0), \ldots, (b^{T-1}, y^{T-1})$ and $(b^\infty, 1)$. A point-based update is performed in b^T (generating point (b^T, y^T)). It holds

$$\underline{v}^m(b^T) = c(*, m) + V^{\Upsilon}_{\mathrm{LB}}(b^{T+1}) = 1 + V^{\Upsilon}_{\mathrm{LB}}(b^{T+1}) \ . \tag{6.4}$$

At this point, $V^{\Upsilon}_{\mathrm{LB}}(b^{T+1})$ is a convex combination of values $y^{T-1} = V^{\Upsilon}_{\mathrm{LB}}(b^{T-1})$ and $V^{\Upsilon}_{\mathrm{LB}}(b^\infty)$, $0.8^2 y^{T-1} + 0.36 \cdot 1$ (see Figure 6.3 for illustration). If m was always optimal, the sequence of $y^T = \underline{v}^m(b^T)$ values can be characterized by a difference equation $y^T = 1 + 0.8^2 y^{T-1} + 0.36 \cdot 1$. For $y^0 = 6$, this sequence (generating points on the dashed line in Figure 6.3) is decreasing and it never exceeds value 50. Hence m is indeed always optimal and actions r_1 and r_2 are never used during the trial (since $\underline{v}^m(b^T) \leq \underline{v}^{r_i} = 50$. Note that $V^{\Gamma}_{\mathrm{UB}}(b^T) \geq V^*(b^T) = 50$ for every $T \in \mathbb{Z}^+_0$ (we cannot avoid playing r_i which yields expected cost 50). Therefore, for sufficiently small ε the excess gap $V^{\Gamma}_{\mathrm{UB}}(b^T) - V^{\Upsilon}_{\mathrm{LB}}(b^T) - \varepsilon$ is always positive, and the trial never terminates.

Solution: In our Goal-HSVI algorithm, we cut off excessively long samples while guaranteeing that an ε-optimal solution is found within this limit. Such an approach has been used previously in practice without studying its impact on the solution quality (e.g., in POMDPSolver[1] for discounted-sum POMDPs that was used in [Warnquist et al., 2013], the depth limit is set to 200). However, using any fixed depth limit for all indefinite-horizon POMDPs is not sound (consider a POMDP where more than 200 precisely timed steps have to be taken to reach the goal). We address this and consider a depth limit that provides theoretical guarantees on the solution quality.

Observation-selection heuristic may suppress exploration. Finally, terminating excessively long trials alone is not sufficient. Consider the example from [Smith, 2007, Theorem 6.9]. Value functions $V^{\Upsilon}_{\mathrm{LB}}$ and V^{Γ}_{UB} are never changed during the trial. Hence, if we terminate a trial prematurely, a new trial operates on the same $V^{\Upsilon}_{\mathrm{LB}}$ and V^{Γ}_{UB}, and thus it visits the same beliefs again where no improvement can be made. Thus it remains in states s_0 and s_1 – and the state s_2 that is necessary for convergence is never considered.

Solution: There are multiple possible solutions to this problem (e.g., changing the observation-selection heuristic). To remain consistent with vanilla-HSVI (and use the same heuristic), our Goal-HSVI algorithm keeps a data structure (a *closed-list*, denoted

[1]Retrieved on December 20, 2019 from `http://www.bgu.ac.il/~shanigu/`

CL) tracking the action-observation sequences that have been already considered. When selecting an observation according to the observation-selection heuristic, the algorithm avoids choosing one that would lead to an action-observation sequence fully explored earlier.

6.3 Goal-HSVI

In this section, we present our novel Goal-HSVI approach to solve Goal-POMDPs. In Section 6.3.1, we present the basic algorithm and its theoretical guarantees, and in Section 6.3.2, we extend this algorithm to obtain a practical approach.

6.3.1 Basic Algorithm

The changes we presented in the previous section constitute the basis for our Goal-HSVI algorithm (see Algorithm 6.3). Our algorithm extends the vanilla-HSVI (Algorithm 6.2) with the three following key modifications:

(1) *The uniform policy is used for initialization of* V^{Γ}_{UB} *(line 1 of Algorithm 6.3).*

(2) *The search depth is bounded (line 5).* We terminate trials longer than $\frac{\overline{C}}{c_{\min}} \cdot \frac{\overline{C}-\eta\varepsilon}{(1-\eta)\varepsilon}$ steps where $\overline{C}$ is the upper bound on cost of playing the uniform policy and $c_{\min}$ is the minimum per-step cost. We prove that this choice together with a stricter termination condition $V^{\Gamma}_{\mathrm{UB}}(b) - V^{\Upsilon}_{\mathrm{LB}}(b) \leq \eta\varepsilon$ (for $\eta \in [0,1)$) guarantees that an ε-optimal solution is found by the Goal-HSVI algorithm (see Theorem 6.1).

(3) *Exploring the same history more than once is avoided.* We keep track of the action-observation history $(\overline{\mathbf{a}}, \overline{\mathbf{o}})$ during the exploration. A history is marked as *closed* by adding it to the closed list CL in case the history is terminal (line 6) or all histories reachable when using action a^* are already closed (lines 9-10). The set of observations O' denotes all observations that lead to an action-observation history that has not yet been closed. Together with the choice of $o^* \in O'$ (line 11 of Algorithm 6.3), this guarantees that no action-observation history is considered twice.

Note that the closed list CL can be efficiently represented using a prefix tree. Each action-observation history is represented by a path in the tree (where each node corresponds to playing the given action or seeing the given observation). Furthermore, each node is attributed a binary flag indicating whether the given action-observation history is closed. The memory efficiency of this representation comes from the fact that multiple histories share the same prefix (and thus the same part of the path). This claim is supported by the experimental evaluation (see Table 6.1) as the number of nodes in the prefix tree is typically comparable with the number of elements representing value functions $V^{\Upsilon}_{\mathrm{LB}}$ and V^{Γ}_{UB}.

Algorithm 6.3: Goal-HSVI, $\eta \in [0,1)$

1 Initialize $V^{\Upsilon}_{\mathrm{LB}}$ and V^{Γ}_{UB} (V^{Γ}_{UB} initialized by the uniform policy)
2 $CL \leftarrow \varnothing$
3 **while** $V^{\Gamma}_{\mathrm{UB}}(b^{\mathrm{init}}) - V^{\Upsilon}_{\mathrm{LB}}(b^{\mathrm{init}}) > \varepsilon$ **do** `explore`$(b^{\mathrm{init}}, \varepsilon, 0, \varnothing, \varnothing)$
4 **procedure** `explore`$(b, \varepsilon, t, \overline{\mathbf{a}}, \overline{\mathbf{o}})$
5 **if** $V^{\Gamma}_{\mathrm{UB}}(b) - V^{\Upsilon}_{\mathrm{LB}}(b) \leq \eta\varepsilon$ **or** $t \geq \frac{\overline{C}}{c_{\min}} \cdot \frac{\overline{C}-\eta\varepsilon}{(1-\eta)\varepsilon}$ **then**
6 $CL \leftarrow CL \cup \{(\overline{\mathbf{a}}, \overline{\mathbf{o}})\}$ and **return**
7 $a^* \leftarrow \arg\min_a \left[\sum_s b(s)c(s,a) + \sum_o \mathbb{P}_b[o|a] V^{\Upsilon}_{\mathrm{LB}}(\tau(b,a,o))\right]$
8 `update`(b)
9 $O' \leftarrow \{o \in O \mid (\overline{\mathbf{a}}a^*, \overline{\mathbf{o}}o) \notin CL\}$
10 **if** $O' = \varnothing$ **then** $CL \leftarrow CL \cup \{(\overline{\mathbf{a}}, \overline{\mathbf{o}})\}$ and **return**
11 $o^* \leftarrow \arg\max_{o \in O'} \mathbb{P}_b[o|a^*] \cdot [V^{\Gamma}_{\mathrm{UB}}(\tau(b,a^*,o)) - V^{\Upsilon}_{\mathrm{LB}}(\tau(b,a^*,o)) - \eta\varepsilon]$
12 `explore`$(\tau(b,a^*,o^*), \varepsilon, t+1, \overline{\mathbf{a}}a^*, \overline{\mathbf{o}}o^*)$
13 `update`(b)

Theorem 6.1. *Assume $\eta \in [0,1)$ and let $\overline{C}$ be the maximum cost of playing the uniform policy ($\overline{C} = \max_s V^{\Gamma}_{\mathrm{UB}}(b^s)$ for the initial V^{Γ}_{UB}). Let $c_{\min} = \min_{s,a} c(s,a)$. Then the Goal-HSVI algorithm (Algorithm 6.3) with the cutoff at depth $\overline{T} = \frac{\overline{C}}{c_{\min}} \cdot \frac{\overline{C}-\eta\varepsilon}{(1-\eta)\varepsilon}$ terminates and yields an ε-approximation of $V^*(b^{\mathrm{init}})$.*

Proof. Let $(\varnothing, \varnothing) \in CL$. Let ω be an action-observation history and consider a policy $\pi(\omega)$ where the agent plays action a^* chosen in `explore` when ω (represented by $(\overline{\mathbf{a}}, \overline{\mathbf{o}})$ in the algorithm) was closed on line 10. Since a history is closed when the horizon $\overline{T}$ or precision $\eta\varepsilon$ is reached, or when all action-observation histories reached by playing a^* are closed, it is clear that all plays according to π eventually reach a terminal action-observation history (closed on line 6). Let b_ω be the belief after experiencing action-observation history ω and let us assign values $v_{\mathrm{LB}}(\omega)$ and $v_{\mathrm{UB}}(\omega)$ to each terminal action-observation history ω corresponding to values $V^{\Upsilon}_{\mathrm{LB}}(b_\omega)$ and $V^{\Gamma}_{\mathrm{UB}}(b_\omega)$ at the time ω has been closed. Let us propagate values v_{LB} and v_{UB} using Bellman equation in a bottom-up manner, i.e. for $i \in \{\mathrm{LB}, \mathrm{UB}\}$,

$$v_i(\omega) = c(\omega, \pi(\omega)) + \textstyle\sum_o \mathbb{P}_{b^\omega}[o \mid \pi(\omega)] \cdot v_i(\omega\pi(\omega)o) \ . \tag{6.5}$$

Since the bounds might have improved since the histories were closed, it holds $v_{\mathrm{LB}}(\varnothing) \leq V^{\Upsilon}_{\mathrm{LB}}(b^{\mathrm{init}}) \leq V^{\Gamma}_{\mathrm{UB}}(b^{\mathrm{init}}) \leq v_{\mathrm{UB}}(\varnothing)$. Observe that the probability that any history consistent with π reaches the depth limit $\overline{T}$ (i.e., the sum of probabilities of all plays reaching the depth $\overline{T}$ when following π) is at most $p_{\mathrm{cutoff}} \leq (1-\eta)\varepsilon/(\overline{C} - \eta\varepsilon)$. Otherwise, the contribution of those plays to the expected cost $v_{\mathrm{LB}}(\varnothing)$ would have been greater than $\overline{C}$ (as at least $c_{\min}$ is paid per step) which would have contradicted that $\overline{C}$ is the upper bound. Now, since $v_{\mathrm{UB}}(\omega) - v_{\mathrm{LB}}(\omega)$ in terminal histories that were cut off is less than $\overline{C}$

(reached with probability p_{cutoff}), and $v_{\text{UB}}(\omega) - v_{\text{LB}}(\omega) \leq \eta\varepsilon$ otherwise, the difference $v_{\text{UB}}(\varnothing) - v_{\text{LB}}(\varnothing)$ in the root is a weighted average of these values,

$$v_{\text{UB}}(\varnothing) - v_{\text{LB}}(\varnothing) \leq p_{\text{cutoff}}\overline{C} + (1 - p_{\text{cutoff}})\eta\varepsilon \leq \varepsilon \ . \tag{6.6}$$

As $v_{\text{UB}}(\varnothing) - v_{\text{LB}}(\varnothing) \geq V^{\Gamma}_{\text{UB}}(b^{\text{init}}) - V^{\Upsilon}_{\text{LB}}(b^{\text{init}})$, the result follows. □

6.3.2 Practical Extension: Iterative Deepening

The bound on the search depth induced by Theorem 6.1 can be unnecessarily large in practice. To avoid generating excessively long trials, we propose a variant of our Goal-HSVI algorithm enriched by the ideas of iterative deepening. Instead of terminating trials when they exceed $\overline{T} = \frac{\overline{C}}{c_{\min}} \cdot \frac{\overline{C}-\eta\varepsilon}{(1-\eta)\varepsilon}$, a cut-off depth $\widehat{T}$ is introduced, starting with $\widehat{T} \coloneqq 1$ and increasing it when one of the following situations occur:

(1) *All action-observation histories within the current depth limit $\widehat{T}$ are explored.* If no ε-optimal solution was found with the current depth limit, the limit must be increased. This situation is indicated by $(\varnothing, \varnothing) \in CL$.

(2) *The improvement during the current trial was insufficient.* We say that the improvement is sufficient if the difference $V^{\Gamma}_{\text{UB}}(b) - V^{\Upsilon}_{\text{LB}}(b)$ *before* and *after* a call to `explore` (denoted δ and δ') weighted by the probability of seeing the observation sequence $\overline{\mathbf{o}}$ when playing actions $\overline{\mathbf{a}}$ (denoted $O_{\overline{\mathbf{a}}}(\overline{\mathbf{o}})$) is greater than $\rho(\widehat{T})$, i.e. $O_{\overline{\mathbf{a}}}(\overline{\mathbf{o}}) \cdot (\delta - \delta') \geq \rho(\widehat{T})$. Our implementation uses $\rho(\widehat{T}) = p^{\widehat{T}}$ with $p = 0.95$.

Whenever $\widehat{T}$ is increased, CL is set to $\varnothing$ to allow the algorithm to re-explore action-observation histories considered previously and search them to a greater depth. In our implementation, we always increase the search depth $\widehat{T}$ by one.

6.4 Empirical Evaluation

We present an experimental evaluation of Goal-HSVI in comparison with RTDP-Bel [Bonet and Geffner, 2009]. We do not include vanilla-HSVI (HSVI2 [Smith and Simmons, 2005]) in the experiments as the algorithm without modifications is theoretically incorrect in the Goal-POMDP setting, and it indeed crashed due to the recursion limit on some instances (Hallway 2, Seq[5,5,3], Seq[5,5,4]). We start with the description of the setting of the algorithms considered.

Goal-HSVI. Our implementation is based on the ZMDP[2] implementation of HSVI2. We updated the solver according to Section 6.3. Few other changes were made: (1) We do not use adaptive precision from ZMDP that changes ε between iterations (fixed values $\varepsilon = 2$ and $\eta = 0.8$ are used). (2) We do not use α-vector masking as the implementation of this technique in ZMDP is incompatible with Goal-POMDPs. We terminate the algorithm after 900s if an ε-optimal solution is not found.

[2] https://github.com/trey0/zmdp

				Goal-HSVI (15min time limit)							RTDP-Bel (150k trials)		
	$\|S\|$	$\|A\|$	$\|O\|$	Cost	%Goal	Bounds	Time	t_{ref}	$\|CL\|$	$\|\Gamma \cup \Upsilon\|$	Cost	%Goal	Time
Hallway	60	5	21	14.4±0.04	100.0%	[12.8 .. 15.0]	904s	1s	14277	16788	64.7±1.2	97.5%	397s
Hallway 2	92	5	17	29.2±0.07	100.0%	[13.4 .. 66.4]	909s	2s	14616	12648	356.1±2.8	83.5%	5071s
RS[4,4]	257	9	2	231.0±0.01	100.0%	[229.0 .. 231.0]	23s	1s	16348	6651	230.9±0.3	100.0%	12s
RS[5,5]	801	10	2	306.9±0.04	100.0%	[299.2 .. 306.9]	900s	9s	62926	31661	309.9±0.3	100.0%	23s
RS[5,7]	3201	12	2	336.5±0.01	100.0%	[310.8 .. 336.5]	901s	120s	17137	8078	336.3±0.4	100.0%	62s
Seq[5,5,2]	121	5	2	15.5±0.02	100.0%	[14.0 .. 16.0]	196s	1s	19675	24107	138.9±1.9	93.8%	38s
Seq[5,5,3]	281	5	2	35.4±0.04	100.0%	[27.4 .. 36.7]	901s	1s	46753	37915	1496.7±3.3	25.9%	645s
Seq[5,5,4]	601	5	2	43.4±0.07	100.0%	[32.5 .. 51.0]	901s	1s	31355	30070	1426.5±3.5	29.6%	841s

Table 6.1: Experimental results (on Intel Core i7-8700K). Cost denotes average cost of the computed policy for the first 2,000 steps taken over 250,000 simulated plays. %Goal is the percentage of the simulated plays that reached a target in less than 2,000 steps. t_{ref} denotes the time when the Goal-HSVI upper bound $V^{\Gamma}_{\text{UB}}(b^{\text{init}})$ reached the confidence interval of the cost of RTDP-Bel. $|CL|$ denotes the size of the closed list as the number of nodes of the representing prefix tree. 95% confidence intervals are reported.

RTDP-Bel. GPT solver[3] is used as a reference implementation of RTDP-Bel. Since there are no guidelines for choosing the parameters of RTDP-Bel, we use the default values used in GPT (most importantly, $K = 15$ as in [Bonet and Geffner, 2009]) except for increasing the cutoff parameter from 250 to 2000. In our experiments we let RTDP-Bel perform 150,000 trials before terminating. As RTDP-Bel is a randomized algorithm, we perform 12 independent runs and report the result with the lowest average cost. We consider the cost of RTDP-Bel policies as a reference, and we report the time when Goal-HSVI finds a policy of the same quality as t_{ref}.

Policy evaluation. We evaluate the quality of the policies computed by the algorithms using simulation. We perform 250,000 simulated plays (we cut each of them after 2,000 steps if the goal is not reached by that time) and we report the average total cost. We also report the percentage of simulated plays that did not terminate within the limit.

We evaluate the performance of our Goal-HSVI algorithm on three different domains. The domains are: *Hallway* [Littman et al., 1995] and *RockSample* [Smith and Simmons, 2004] in their Goal-POMDP variants, and a new domain of *Sequencing* (inspired by [Kress-Gazit et al., 2009]).

Hallway [Littman et al., 1995]**.** An agent is navigating in a maze trying to reach the goal location while using unreliable actuators and sensors. In the original version, the agent receives a reward only when the goal is reached, and the discounted-sum objective is considered. For the Goal-POMDP version of the problem, we assume that the goal state is absorbing and each step of the agent costs one unit. Bonet and Geffner [2009] observed that RTDP-Bel had been outperformed by HSVI2 in the discounted-sum setting. Our Goal-HSVI algorithm similarly outperforms RTDP-Bel in the Goal-POMDP variant of Hallway (see Table 6.1). Moreover, the upper bound on cost produced by Goal-HSVI

[3] https://github.com/bonetblai/gpt-rewards

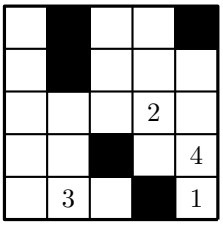


Figure 6.4: Seq[5,5,4]

after 2s is lower than the cost of RTDP-Bel, and unlike RTDP-Bel, the policy produced by our algorithm always reached the goal in less than 2000 steps. (RTDP-Bel failed to reach goal in 2.5% and 16.5% of plays on Hallway and Hallway 2, respectively.)

RockSample[n,k] [Smith and Simmons, 2004]**.** A robot is operating in an $n \times n$ grid with k rocks. Each of the rocks can be either 'good' or 'bad' (unknown to the agent). The goal is to sample all the good rocks (approach them and perform expensive sampling) and then leave the map. In the Goal-POMDP version [Chatterjee et al., 2016], a cost is associated with each movement. Moreover, the agent pays a penalty for all the 'good' rocks he failed to sample upon leaving the map.

RTDP-Bel works well on discounted RockSample [Bonet and Geffner, 2009] due to the problem structure (e.g., observability of the current position), and the same is to be expected in the Goal-POMDP setting. Although Goal-HSVI does not leverage the problem structure, it is competitive on all RockSample instances we consider, see Table 6.1. Moreover, it *provably* found solutions of a comparable (or even better) quality as RTDP-Bel by decreasing the upper bound on cost (see t_{ref} for the time required). Recall that RTDP-Bel cannot provide any such guarantees on the quality of the computed policy.

Sequencing[n,k,t]**.** An agent inspects t targets in an $n \times n$ grid world with k obstacles (see Figure 6.4). He is uncertain about his position, and he has 5 actions available – 4 movement actions N, S, W, E and the inspection action. The movement actions are not reliable and may result in a step in any unintended direction with probability 0.15. The inspection action is deterministic and inspects the target (if there is one at the current position). The agent may receive two observations – either the last action succeeded (he stepped on an empty square / inspected a target) or it failed (he hit an obstacle / there is no target to inspect). An additional challenge is that he has to inspect the targets in the prescribed order – otherwise he pays a penalty $100t$ where t is the number of targets he should have inspected earlier. For example, if he inspects targets in Figure 6.4 in the order $(4, 1, 3, 2)$, he accumulates a penalty 400.

We observe that RTDP-Bel does not work well for Sequencing and fails to find a policy reaching the goal state reliably, especially on the larger two instances. In contrary, our Goal-HSVI algorithm produces superior policies that always reached the goal (see Table 6.1). Notice that on Sequencing[5,5,3] and Sequencing[5,5,4], the time to complete 150,000 trials of RTDP-Bel is comparable to the time given to Goal-HSVI, yet still, the policy of RTDP-Bel is inferior.

CHAPTER 7

Conclusion

Partially observable stochastic games represent a very general class of models for reasoning about a wide range of real-world adversarial problems. Their generality, however, comes at a computational cost as solving partially observable stochastic games is highly intractable [Goldsmith and Mundhenk, 2008]. In this thesis, we have provided scalable algorithms for solving relevant subclasses of partially observable stochastic games, and we have also explored the application potential of the proposed methods. In this chapter, we provide an overview of the main contributions of the thesis, and we will also outline directions of future work.

7.1 Thesis Contributions

Solving one-sided POSGs (Chapter 3) One-sided partially observable stochastic games (OS-POSGs) represent a subclass of general partially observable stochastic games. Here, the game is played by two competing players, where only one player is imperfectly informed about the course of the game, while the opponent is able to observe the entire history of the game. While this assumption can be seen as overly restrictive, a wide range of problems, especially those arising in security applications, fit well within the boundaries of the one-sided POSG model. In security applications, we typically do not know the information available to the adversary (i.e., the attacker). By assuming that the attacker is perfectly informed in the game, we can derive robust strategies for the defender that provide guarantees even when the attacker is less informed. The examples of security problems that naturally fit within the boundaries of the one-sided POSG model include, e.g., patrolling [Basilico et al., 2009a; Vorobeychik et al., 2014].

Despite the importance of this class of games, to the best of our knowledge, no previous scalable algorithm to compute solutions of OS-POSGs existed. We provide a

detailed theoretical analysis of the class of one-sided POSGs.[1] Based on these insights and inspired by the results on efficient point-based methods for solving single-agent POMDPs [Pineau et al., 2003; Smith and Simmons, 2004, 2005; Spaan and Vlassis, 2005] which rely on similar structural results from the realm of POMDPs, we design a scalable algorithm to solve OS-POSGs. Our proposed algorithm [Horák et al., 2017a] is inspired by the well-known *heuristic search value iteration* algorithm for solving POMDPs [Smith and Simmons, 2004, 2005] and provably converges to near-optimal solutions of the one-sided POSG model.

The importance and novelty of our results is highlighted by a recent publication of Zhang et al. [2019] which appeared during the time of writing this thesis. Here, the authors study the model of so-called *non-cooperative inverse reinforcement learning* (N-CIRL). This model represents an interaction between the defender and the attacker, where only the *intent* of the attacker (that influences the rewards) is unknown to the defender. The N-CIRL model naturally belongs to the class of one-sided POSGs, and hence our methods directly apply to this model as well.

Compact representation of information to improve scalability (Chapter 4) In one-sided POSGs, a belief is a probability distribution over the states of the game. Such representation becomes unmanageable in case the number of states of the game is high. Instead of dealing with high-dimensional beliefs, we propose to represent beliefs compactly using low-dimensional characteristic vectors. We further show that the algorithmic results from one-sided POSGs can be extended to use the compact representation.

We apply the ideas on the problem of *lateral movement* that is well-established in the cyber-security literature [Kamdem et al., 2017]. Here, we assume that the attacker has managed to establish his foothold within a computer network (e.g., he has access to at least one node within the network), and he now plans his steps towards increasing his presence by compromising additional assets. On the other hand, the defender is trying to harden the progress of the attacker by placing honeypots within the network. Since the attacker can control almost arbitrary subsets of nodes at a time, the state space of the game is exponential in the size of the network. We demonstrate that the approximate algorithm based on the use of low-dimensional characterization of beliefs significantly improves the scalability on this game when compared to the exact approach from Chapter 3.

POSGs with two-sided information (Chapter 5) We generalize the one-sided POSG model towards a setting where *both* players are imperfectly informed about the course of the game. We introduce a model of partially observable stochastic games with public observations (PO-POSGs), where the state space of the game is factored, and the dynamics of the state spaces of the players is coupled only via a pair of public observations.

[1] A limited subset of the presented theoretical results has appeared in [Sorin, 2003]. Our original publication [Horák et al., 2017a] has been prepared independently of these results. We expand this discussion significantly, while our main contribution is the algorithmic solution to solve one-sided POSGs.

We generalize the structural results from one-sided POSG case to PO-POSGs, and we show that we can also extend the HSVI algorithm towards PO-POSGs.

While the model cannot capture all kinds of imperfect information in POSGs (we require that each player has to be able to infer the belief of the adversary), we believe that the class of games that can be modeled as PO-POSGs is rather wide. As an example, consider the one-sided N-CIRL model from [Zhang et al., 2019]. The N-CIRL model assumes that the only imperfect information in the game is the lack of the defender's knowledge of the intent of the attacker. Using the PO-POSG model, we can easily add uncertainty for the attacker, e.g., about the location of valuable assets or the state of defensive measures.

HSVI for Goal-POMDPs (Chapter 6) The algorithms for solving one-sided POSGs (Chapter 3) and POSGs with public observations (Chapter 5) apply to the setting where the discounted sum $\sum_{t=1}^{\infty} \gamma^{t-1} r_t$ of rewards is the optimization objective. In some cases, however, discounting the rewards is not desirable. The examples include, e.g., mission planning in robotics scenarios. Here, the goal is to optimize the *total*, i.e., undiscounted, energy consumption of the robot during the time he executes the mission. Such objective is known as *indefinite-horizon* objective [Bertsekas and Tsitsiklis, 1996; Patek, 2001; Bonet and Geffner, 2009; Kolobov et al., 2011; Chatterjee et al., 2016] and cannot be modeled as a discounted problem. For example, if we use discounting in the robotics scenario, the robot would have been able to delay the execution of costly parts of the mission to decrease the discounted sum of costs, while the actual undiscounted cost of executing the mission remains the same, or may even increase.

As a first step towards designing algorithms for solving POSGs with indefinite-horizon objective, we study the problem of Goal-POMDPs. Here, the goal of the agent is to reach the goal state while minimizing the total expected cost to reach the goal. We study the problems associated with applying HSVI algorithm [Smith and Simmons, 2005] to Goal-POMDPs. After identifying these problems, we propose a variant of the HSVI algorithm that is tailored to solving Goal-POMDPs and provably converges to the ε-optimal solution, unlike the prior heuristic approach RTDP-Bel [Bonet, 1998; Bonet and Geffner, 2009]. Furthermore, experimental results suggest that the algorithm is able to outperform RTDP-Bel in the quality of the solution.

7.2 Future Work

We believe that the algorithmic ideas presented in this thesis open up a wide range of possible directions for future work. In this section, we discuss some of these directions.

Domain-Independent Compact Representation In Chapter 4, we have presented the idea of using compact representation of the beliefs to improve the scalability of the algorithm for solving one-sided POSGs proposed in Chapter 3. While the theoretical discussion of the approach has been framed in a general setting, the algorithmic results

have been obtained for a class of games originating in cybersecurity. It is natural to aim at extending these algorithmic results to a broader class of games. As an example, we can consider that the domain is characterized by propositional variables, and states in the game are characterized by true/false assignments to these variables[2]. We can then define the characteristic vector χ using a set of formulas $\{\varphi_1, \ldots, \varphi_k\}$, such that χ_i corresponds to the probability that the formula φ_i is true in the current belief.

Another line of research applies to the refinements of the proposed abstraction scheme. Instead of using a fixed specification of characteristic vectors as in Chapter 4, we can incrementally generate new coordinates of characteristic vectors while solving the game. As an example, we can add new coordinates to characteristic vectors when we realize that the information provided by the currently considered characteristic vectors is insufficient. We hope that such ideas can result in an algorithm that can provide approximation guarantees without reasoning about high-dimensional beliefs of the unabstracted game.

We also believe that the compact representation of beliefs may allow us to tackle games beyond the one-sided POSG setting. As an example, consider games where player 2 can no longer observe the true state of the game as in one-sided POSGs, but he still observes actions and observations of player 1. In this setting, player 2 is more informed than player 1, but since he does not know the state, he will have to form a belief about the current state of the game. This belief depends not only on information available to player 1 (i.e., actions and observations of player 1), but also on private information player 2 acquired (e.g., private actions of player 2 also influence the transitions). The less informed player 1 has to reason about the belief of the adversary, and hence he will have to reason about possible private histories of player 2. These histories are, however, sequences of privately acquired pieces of information and hence, similarly to [Wiggers et al., 2016], their number grows exponentially in time. The compact representation may allow us to handle this exponential growth, and even allow us to approximately solve infinite-horizon games with the above-mentioned information structure where the number of possible private histories of player 2 is unbounded. To achieve this goal, we can extract significant features of possible private histories of player 2 (e.g., number of timesteps elapsed since the attacker last saw the defender) and use these features to form characteristic vectors of bounded dimension.

Solving One-Sided Reachability Games In Chapters 3 and 5, we have presented algorithms for solving games with the discounted sum objective. Another important objective for security problems is the reachability objective [De Alfaro et al., 2007; Hansen et al., 2009, 2011] which is common in verification community. Here, the defender does not optimize his discounted payoff. Instead, we optimize the probability that the defender can successfully accomplish his goal (such as capturing the evader or bringing the system

[2]Similar representation is common in planning and is used in the STRIPS formalism [Fikes and Nilsson, 1971].

to a safe state). To the best of our knowledge, no algorithms for quantitatively[3] solving stochastic games with the reachability objective and partial observability exist. Although the negative results for concurrent reachability games with perfect information [Hansen et al., 2009, 2011] suggest that we cannot expect to design a practical algorithm for the worst-case instances, we believe that we can obtain a scalable algorithm to solve practical instances of reachability games where the defender is imperfectly informed by extending the results from Chapters 3 and 6.

[3]By solving the game quantitatively, we compute the probability that the defender can successfully accomplish his goal.

APPENDIX A

Publications

The list below is a summary of the publications of the author. Publications in individual categories are sorted by citation counts (excluding self-citations).

A.1 Publications related to the thesis

Journal publications (with IF)

[Horák et al., 2019b] Horák, K., Bošanský, B., Tomášek, P., Kiekintveld, C., and Kamhoua, C. (2019b). Optimizing honeypot strategies against dynamic lateral movement using partially observable stochastic games. *Computers & Security*, 87:101579 (40%)

CORE A* conference publications

[Horák et al., 2017a] Horák, K., Bošanský, B., and Pěchouček, M. (2017a). Heuristic Search Value Iteration for One-Sided Partially Observable Stochastic Games. In *31st AAAI Conference on Artificial Intelligence*, pages 558–564 (50%, **9 citations**)

[Horák and Bošanský, 2019] Horák, K. and Bošanský, B. (2019). Solving partially observable stochastic games with public observations. In *33rd AAAI Conference on Artificial Intelligence*, pages 2029–2036 (70%, **1 citation**)

[Horák et al., 2018] Horák, K., Bošanský, B., and Chatterjee, K. (2018). Goal-HSVI: Heuristic Search Value Iteration for Goal POMDPs. In *27th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4764–4770 (45%)

[Horák et al., 2019a] Horák, K., Bošanský, B., Kiekintveld, C., and Kamhoua, C. (2019a). Compact Representation of Value Function in Partially Observable Stochastic Games. In *28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 350–356 (65%)

Other publications

[Horák et al., 2017b] Horák, K., Zhu, Q., and Bošanský, B. (2017b). Manipulating adversary's belief: A dynamic game approach to deception by design for proactive network security. In *International Conference on Decision and Game Theory for Security (GameSec)*, pages 273–294 (33%, **11 citations**)

Other WoS publications

[Horák and Bošanský, 2016] Horák, K. and Bošanský, B. (2016). A Point-Based Approximate Algorithm for One-Sided Partially Observable Pursuit-Evasion Games. In *International Conference on Decision and Game Theory for Security (GameSec)*, pages 435–454 (50%, **4 citations**)

[Horák and Bošanský, 2017] Horák, K. and Bošanský, B. (2017). Dynamic Programming for One-sided Partially Observable Pursuit-evasion Games. In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART)*, pages 503–510 (50%)

A.2 Other publications

Journal publications (with IF)

[Čermák et al., 2018] Čermák, J., Bošanský, B., Horák, K., Lisý, V., and Pěchouček, M. (2018). Approximating maxmin strategies in imperfect recall games using A-loss recall property. *International Journal of Approximate Reasoning*, 93:290–326 (10%, **4 citations**)

Other publications

[Durkota et al., 2017] Durkota, K., Lisý, V., Kiekintveld, C., Horák, K., Bošanský, B., and Pevný, T. (2017). Optimal strategies for detecting data exfiltration by internal and external attackers. In *International Conference on Decision and Game Theory for Security (GameSec)*, pages 171–192 (16%, **2 citations**)

Other WoS publications

[Bošanský et al., 2017] Bošanský, B., Čermák, J., Horák, K., and Pěchouček, M. (2017). Computing Maxmin Strategies in Extensive-form Zero-sum Games with Imperfect Recall. In *9th International Conference on Agents and Artificial Intelligence (ICAART)*, pages 63–74 (20%)

Bibliography

[Abuzainab and Saad, 2018] Abuzainab, N. and Saad, W. (2018). Dynamic connectivity game for adversarial internet of battlefield things systems. *IEEE Internet of Things Journal*, 5(1):378–390.

[Amato et al., 2010] Amato, C., Bernstein, D. S., and Zilberstein, S. (2010). Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *9th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 21(3):293–320.

[Amigoni and Basilico, 2012] Amigoni, F. and Basilico, N. (2012). A game theoretical approach to finding optimal strategies for pursuit evasion in grid environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2155–2162.

[Astrom, 1965] Astrom, K. J. (1965). Optimal control of Markov processes with incomplete state information. *Journal of mathematical analysis and applications*, 10(1):174–205.

[Barto et al., 1995] Barto, A. G., Bradtke, S. J., and Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial intelligence*, 72(1-2):81–138.

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

[Basilico et al., 2009b] Basilico, N., Gatti, N., Rossi, T., Ceppi, S., and Amigoni, F. (2009b). Extending Algorithms for Mobile Robot Patrolling in the Presence of Adversaries to More Realistic Settings. In *IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 557–564.

[Basilico et al., 2016] Basilico, N., Nittis, G. D., and Gatti, N. (2016). A Security Game Combining Patrolling and Alarm–Triggered Responses Under Spatial and Detection Uncertainties. In *30th AAAI Conference on Artificial Intelligence*, pages 397–403.

[Bernstein et al., 2009] 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*, 34:89–132.

[Bernstein et al., 2005] Bernstein, D. S., Hansen, E. A., and Zilberstein, S. (2005). Bounded policy iteration for decentralized POMDPs. In *19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 52–57.

[Bertsekas and Tsitsiklis, 1996] Bertsekas, D. P. and Tsitsiklis, J. N. (1996). Neuro-dynamic programming. *Athena Scientific*.

[Bonet, 1998] Bonet, B. (1998). Solving large POMDPs using real time dynamic programming. In *AAAI Fall Symposium on POMDPs*.

[Bonet and Geffner, 2009] Bonet, B. and Geffner, H. (2009). Solving POMDPs: RTDP-Bel vs. Point-based Algorithms. In *19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1641–1646.

[Bošanský et al., 2014] Bošanský, B., Kiekintveld, C., Lisý, V., and Pěchouček, M. (2014). An Exact Double-Oracle Algorithm for Zero-Sum Extensive-Form Games with Imperfect Information. *Journal of Artificial Intelligence Research*, 51:829–866.

[Bošanský et al., 2017] Bošanský, B., Čermák, J., Horák, K., and Pěchouček, M. (2017). Computing Maxmin Strategies in Extensive-form Zero-sum Games with Imperfect Recall. In *9th International Conference on Agents and Artificial Intelligence (ICAART)*, pages 63–74.

[Bowen et al., 2009] Bowen, B. M., Hershkop, S., Keromytis, A. D., and Stolfo, S. J. (2009). Baiting inside attackers using decoy documents. In *International Conference on Security and Privacy in Communication Systems*, pages 51–70.

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

[Bowling and Veloso, 2000] Bowling, M. and Veloso, M. (2000). An analysis of stochastic game theory for multiagent reinforcement learning. Technical report, Carnegie Mellon University.

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[Brown and Sandholm, 2018] Brown, N. and Sandholm, T. (2018). Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424.

[Burch, 2018] Burch, N. (2018). *Time and space: Why imperfect information games are hard*. PhD thesis, University of Alberta.

[Burch et al., 2014] Burch, N., Johanson, M., and Bowling, M. (2014). Solving imperfect information games using decomposition. In *28th AAAI Conference on Artificial Intelligence*, pages 602–608.

[Cai et al., 2016] Cai, G.-l., Wang, B.-s., Hu, W., and Wang, T.-z. (2016). Moving target defense: state of the art and characteristics. *Frontiers of Information Technology & Electronic Engineering*, 17(11):1122–1153.

[Čermák et al., 2018] Čermák, J., Bošanský, B., Horák, K., Lisý, V., and Pěchouček, M. (2018). Approximating maxmin strategies in imperfect recall games using A-loss recall property. *International Journal of Approximate Reasoning*, 93:290–326.

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

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

[Cheng, 1988] Cheng, H.-T. (1988). *Algorithms for partially observable Markov decision processes*. PhD thesis, University of British Columbia.

[Chung et al., 2011] Chung, T. H., Hollinger, G. A., and Isler, V. (2011). Search and pursuit-evasion in mobile robotics. *Autonomous robots*, 31(4):299–316.

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

[Cole and Kocherlakota, 2001] Cole, H. L. and Kocherlakota, N. (2001). Dynamic games with hidden actions and hidden states. *Journal of Economic Theory*, 98(1):114–126.

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

[Dibangoye et al., 2016] Dibangoye, J. S., Amato, C., Buffet, O., and Charpillet, F. (2016). Optimally solving Dec-POMDPs as continuous-state MDPs. *Journal of Artificial Intelligence Research*, 55:443–497.

[Du et al., 2017] Du, M., Li, Y., Lu, Q., and Wang, K. (2017). Bayesian Game Based Pseudo Honeypot Model in Social Networks. In *International Conference on Cloud Computing and Security*, pages 62–71.

[Durkota et al., 2015] Durkota, K., Lisý, V., Bošanský, B., and Kiekintveld, C. (2015). Approximate solutions for attack graph games with imperfect information. In *International Conference on Decision and Game Theory for Security (GameSec)*, pages 228–249.

[Durkota et al., 2017] Durkota, K., Lisý, V., Kiekintveld, C., Horák, K., Bošanský, B., and Pevný, T. (2017). Optimal strategies for detecting data exfiltration by internal and external attackers. In *International Conference on Decision and Game Theory for Security (GameSec)*, pages 171–192.

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

[Fang et al., 2016] Fang, F., Nguyen, T. H., Pickles, R., Lam, W. Y., Clements, G. R., An, B., Singh, A., Tambe, M., and Lemieux, A. (2016). Deploying PAWS: Field optimization of the protection assistant for wildlife security. In *30th AAAI Conference on Artificial Intelligence*, pages 3966–3973.

[Fang et al., 2015] Fang, F., Stone, P., and Tambe, M. (2015). When Security Games Go Green: Designing Defender Strategies to Prevent Poaching and Illegal Fishing. In *24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2589–2595.

[Fikes and Nilsson, 1971] Fikes, R. E. and Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208.

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

[Garg and Grosu, 2007] Garg, N. and Grosu, D. (2007). Deception in honeynets: A game-theoretic analysis. In *2007 IEEE SMC Information Assurance and Security Workshop*, pages 107–113.

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

[Gostev and Soumenkov, 2011] 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*.

[Hansen et al., 2004] Hansen, E. A., Bernstein, D. S., and Zilberstein, S. (2004). Dynamic Programming for Partially Observable Stochastic Games. In *National Conference on Artificial Intelligence (AAAI)*, pages 709–715.

[Hansen et al., 2011] Hansen, K. A., Ibsen-Jensen, R., and Miltersen, P. B. (2011). The complexity of solving reachability games using value and strategy iteration. In *International Computer Science Symposium in Russia*, pages 77–90.

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

[Hauskrecht, 2000] Hauskrecht, M. (2000). Value-function approximations for partially observable Markov decision processes. *Journal of artificial intelligence research*, 13:33–94.

[Horák and Bošanský, 2016] Horák, K. and Bošanský, B. (2016). A Point-Based Approximate Algorithm for One-Sided Partially Observable Pursuit-Evasion Games. In *International Conference on Decision and Game Theory for Security (GameSec)*, pages 435–454.

[Horák and Bošanský, 2019] Horák, K. and Bošanský, B. (2019). Solving partially observable stochastic games with public observations. In *33rd AAAI Conference on Artificial Intelligence*, pages 2029–2036.

[Horák and Bošanský, 2017] Horák, K. and Bošanský, B. (2017). Dynamic Programming for One-sided Partially Observable Pursuit-evasion Games. In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART)*, pages 503–510.

[Horák et al., 2018] Horák, K., Bošanský, B., and Chatterjee, K. (2018). Goal-HSVI: Heuristic Search Value Iteration for Goal POMDPs. In *27th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4764–4770.

[Horák et al., 2019a] Horák, K., Bošanský, B., Kiekintveld, C., and Kamhoua, C. (2019a). Compact Representation of Value Function in Partially Observable Stochastic Games. In *28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 350–356.

[Horák et al., 2017a] Horák, K., Bošanský, B., and Pěchouček, M. (2017a). Heuristic Search Value Iteration for One-Sided Partially Observable Stochastic Games. In *31st AAAI Conference on Artificial Intelligence*, pages 558–564.

[Horák et al., 2019b] Horák, K., Bošanský, B., Tomášek, P., Kiekintveld, C., and Kamhoua, C. (2019b). Optimizing honeypot strategies against dynamic lateral movement using partially observable stochastic games. *Computers & Security*, 87:101579.

[Horák et al., 2017b] Horák, K., Zhu, Q., and Bošanský, B. (2017b). Manipulating adversary's belief: A dynamic game approach to deception by design for proactive network security. In *International Conference on Decision and Game Theory for Security (GameSec)*, pages 273–294.

[Isler et al., 2005] Isler, V., Kannan, S., and Khanna, S. (2005). Randomized pursuit-evasion in a polygonal environment. *IEEE Transactions on Robotics*, 21(5):875–884.

[Isler and Karnad, 2008] Isler, V. and Karnad, N. (2008). The role of information in the cop-robber game. *Theoretical Computer Science*, 399(3):179–190.

[Jain et al., 2013] Jain, M., Conitzer, V., and Tambe, M. (2013). Security Scheduling for Real-world Networks. In *12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 215–222.

[Jain et al., 2011] Jain, M., Korzhyk, D., Vaněk, O., Conitzer, V., Tambe, M., and Pěchouček, M. (2011). Double Oracle Algorithm for Zero-Sum Security Games on

Graph. In *10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 327–334.

[Jajodia et al., 2012] Jajodia, S., Ghosh, A. K., Subrahmanian, V., Swarup, V., Wang, C., and Wang, X. S. (2012). *Moving Target Defense II: Application of Game Theory and Adversarial Modeling*, volume 100. Springer.

[Kamdem et al., 2017] Kamdem, G., Kamhoua, C., Lu, Y., Shetty, S., and Njilla, L. (2017). A Markov game theoritic approach for power grid security. In *37th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 139–144.

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

[Kiekintveld et al., 2015] Kiekintveld, C., Lisý, V., and Píbil, R. (2015). Game-theoretic foundations for the strategic use of honeypots in network security. In *Cyber Warfare*, pages 81–101. Springer.

[Koller et al., 1996] 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 et al., 2011] Kolobov, A., Mausam, Weld, D. S., and Geffner, H. (2011). Heuristic Search for Generalized Stochastic Shortest Path MDPs. In *21st International Conference on Automated Planning and Scheduling (ICAPS)*.

[Komiya, 1988] Komiya, H. (1988). Elementary proof for Sion's minimax theorem. *Kodai mathematical journal*, 11(1):5–7.

[Krausz and Rieder, 1997] Krausz, A. and Rieder, U. (1997). Markov games with incomplete information. *Mathematical Methods of Operations Research*, 46(2):263–279.

[Kreibich and Crowcroft, 2004] Kreibich, C. and Crowcroft, J. (2004). Honeycomb: creating intrusion detection signatures using honeypots. *ACM SIGCOMM computer communication review*, 34(1):51–56.

[Kress-Gazit et al., 2009] Kress-Gazit, H., Fainekos, G. E., and Pappas, G. J. (2009). Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics*, 25(6):1370–1381.

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

[Kumar and Zilberstein, 2009] Kumar, A. and Zilberstein, S. (2009). Dynamic programming approximations for partially observable stochastic games. In *22nd International FLAIRS Conference*, pages 547–552.

[La et al., 2016] La, Q. D., Quek, T. Q., Lee, J., Jin, S., and Zhu, H. (2016). Deceptive attack and defense game in honeypot-enabled networks for the internet of things. *IEEE Internet of Things Journal*, 3(6):1025–1035.

[Lanctot et al., 2017] Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Pérolat, J., Silver, D., and Graepel, T. (2017). A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4190–4203.

[Li et al., 2010] Li, X., Cheung, W. K., and Liu, J. (2010). Improving POMDP tractability via belief compression and clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40(1):125–136.

[Littman, 1994] Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier.

[Littman, 1996] Littman, M. L. (1996). *Algorithms for sequential decision making*. PhD thesis, Brown University.

[Littman et al., 1995] Littman, M. L., Cassandra, A. R., and Kaelbling, L. P. (1995). Learning policies for partially observable environments: Scaling up. In *International Conference on Machine Learning*, pages 362–370.

[MacDermed, 2013] MacDermed, C. L. (2013). *Value methods for efficiently solving stochastic games of complete and incomplete information*. PhD thesis, Georgia Institute of Technology.

[Madani et al., 1999] Madani, O., Hanks, S., and Condon, A. (1999). On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *16th National Conference on Artificial Intelligence (AAAI)*, pages 541–548.

[Mairh et al., 2011] Mairh, A., Barik, D., Verma, K., and Jena, D. (2011). Honeypot in network security: a survey. In *International Conference on Communication, Computing & Security*, pages 600–605.

[Mandiant Intelligence Center, 2013] Mandiant Intelligence Center (2013). APT1: Exposing One of China's Cyber Espionage Units. *Mandiant, Tech. Rep.* `https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf`.

[McMahan et al., 2003] McMahan, H. B., Gordon, G. J., and Blum, A. (2003). Planning in the Presence of Cost Functions Controlled by an Adversary. In *20th International Conference on Machine Learning (ICML)*, pages 536–543.

[Mohammadi et al., 2016] Mohammadi, A., Manshaei, M. H., Moghaddam, M. M., and Zhu, Q. (2016). A game-theoretic analysis of deception over social networks using

fake avatars. In *International Conference on Decision and Game Theory for Security (GameSec)*, pages 382–394.

[Moravčík et al., 2017] 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*, pages 508–513.

[Nash et al., 1950] Nash, J. F. et al. (1950). Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49.

[Nawrocki et al., 2016] Nawrocki, M., Wählisch, M., Schmidt, T. C., Keil, C., and Schönfelder, J. (2016). A Survey on Honeypot Software and Data Analysis. *arXiv preprint arXiv:1608.06249*.

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

[Nguyen et al., 2017] Nguyen, T. H., Wellman, M. P., and Singh, S. (2017). A Stackelberg Game Model for Botnet Data Exfiltration. In *International Conference on Decision and Game Theory for Security (GameSec)*, pages 151–170.

[Nikaido, 1953] Nikaido, H. (1953). On a minimax theorem and its applications to functional analysis. *Journal of the Mathematical Society of Japan*, 5(1):86–94.

[Noureddine et al., 2016] Noureddine, M. A., Fawaz, A., Sanders, W. H., and Başar, T. (2016). A game-theoretic approach to respond to attacker lateral movement. In *International Conference on Decision and Game Theory for Security (GameSec)*, pages 294–313.

[Oliehoek, 2013] Oliehoek, F. A. (2013). Sufficient plan-time statistics for decentralized POMDPs. In *23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 302–308.

[Oliehoek et al., 2016] Oliehoek, F. A., Amato, C., et al. (2016). *A concise introduction to decentralized POMDPs*, volume 1. Springer.

[Oliehoek et al., 2017] Oliehoek, F. A., Savani, R., Gallego-Posada, J., Van der Pol, E., De Jong, E. D., and Groß, R. (2017). GANGs: Generative Adversarial Network Games. *arXiv preprint arXiv:1712.00679*.

[Oliehoek et al., 2008] Oliehoek, F. A., Spaan, M. T., and Vlassis, N. (2008). Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32:289–353.

[Papadimitriou and Tsitsiklis, 1987] Papadimitriou, C. H. and Tsitsiklis, J. N. (1987). The Complexity of Markov Decision Processes. *Mathematics of Operations Research*, 12(3):441–450.

[Patek, 2001] Patek, S. D. (2001). On partially observed stochastic shortest path problems. In *IEEE Conference on Decision and Control*, pages 5050–5055.

[Pawlick and Zhu, 2015] Pawlick, J. and Zhu, Q. (2015). Deception by design: evidence-based signaling games for network defense. *arXiv preprint arXiv:1503.05458*.

[Píbil et al., 2012] Píbil, R., Lisý, V., Kiekintveld, C., Bošanský, B., and Pěchouček, M. (2012). Game theoretic model of strategic honeypot selection in computer networks. In *International Conference on Decision and Game Theory for Security (GameSec)*, pages 201–220.

[Pineau et al., 2003] Pineau, J., Gordon, G., Thrun, S., et al. (2003). Point-based value iteration: An anytime algorithm for POMDPs. In *3rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1025–1032.

[Pita et al., 2008] 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 *7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 125–132.

[Rao et al., 1973] Rao, S. S., Chandrasekaran, R., and Nair, K. (1973). Algorithms for discounted stochastic games. *Journal of Optimization Theory and Applications*, 11(6):627–637.

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

[Roy et al., 2005] Roy, N., Gordon, G., and Thrun, S. (2005). Finding approximate POMDP solutions through belief compression. *Journal of artificial intelligence research*, 23:1–40.

[Sandholm, 2015] Sandholm, T. (2015). Steering evolution strategically: Computational game theory and opponent exploitation for treatment planning, drug design, and synthetic biology. In *29th AAAI Conference on Artificial Intelligence*, pages 4057–4061.

[Seitz et al., 2019] Seitz, D., Kovarík, V., Lisý, V., Rudolf, J., Sun, S., and Ha, K. (2019). Value functions for depth-limited solving in imperfect-information games beyond poker. *arXiv preprint arXiv:1906.06412*.

[Shapley, 1953] Shapley, L. S. (1953). Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100.

[Shieh et al., 2012] 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 and Leyton-Brown, 2008] Shoham, Y. and Leyton-Brown, K. (2008). *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press.

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

[Sion, 1958] Sion, M. (1958). On general minimax theorems. *Pacific Journal of mathematics*, 8(1):171–176.

[Smith, 2007] Smith, T. (2007). *Probabilistic planning for robotic exploration*. PhD thesis, Carnegie Mellon University.

[Smith and Simmons, 2004] Smith, T. and Simmons, R. (2004). Heuristic search value iteration for POMDPs. In *20th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 520–527.

[Smith and Simmons, 2005] Smith, T. and Simmons, R. (2005). Point-based POMDP algorithms: improved analysis and implementation. In *21st Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 542–549.

[Somani et al., 2013] Somani, A., Ye, N., Hsu, D., and Lee, W. S. (2013). DESPOT: Online POMDP planning with regularization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1772–1780.

[Sondik, 1971] Sondik, E. J. (1971). The optimal control of partially observable Markov processes. Technical report, Stanford University.

[Sondik, 1978] Sondik, E. J. (1978). The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations research*, 26(2):282–304.

[Sorin, 2003] Sorin, S. (2003). Stochastic games with incomplete information. In *Stochastic Games and applications*, pages 375–395. Springer.

[Spaan and Vlassis, 2005] Spaan, M. T. and Vlassis, N. (2005). Perseus: Randomized point-based value iteration for POMDPs. *Journal of artificial intelligence research*, 24:195–220.

[Spitzner, 2003] Spitzner, L. (2003). *Honeypots: tracking hackers*, volume 1. Addison-Wesley Reading.

[Stoll, 1989] Stoll, C. (1989). *The cuckoo's egg: tracking a spy through the maze of computer espionage*. Doubleday.

[Szer et al., 2005] Szer, D., Charpillet, F., and Zilberstein, S. (2005). MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *21st Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 576–583.

[Underbrink, 2016] Underbrink, A. (2016). Effective cyber deception. In *Cyber Deception*, pages 115–147. Springer.

[Vanderbei, 2015] Vanderbei, R. J. (2015). *Linear programming*. Springer.

[Vaněk et al., 2012] Vaněk, O., Yin, Z., Jain, M., Bošanský, B., Tambe, M., and Pěchouček, M. (2012). Game-theoretic Resource Allocation for Malicious Packet Detection in Computer Networks. In *11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 902–915.

[Virvilis and Gritzalis, 2013] Virvilis, N. and Gritzalis, D. (2013). The big four-what we did wrong in advanced persistent threat detection? In *International Conference on Availability, Reliability and Security*, pages 248–254.

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

[Vorobeychik et al., 2014] 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)*, pages 314–322.

[Wang et al., 2017] Wang, K., Du, M., Maharjan, S., and Sun, Y. (2017). Strategic honeypot game model for distributed denial of service attacks in the smart grid. *IEEE Transactions on Smart Grid*, 8(5):2474–2482.

[Wang et al., 2019] Wang, Y., Shi, Z. R., Yu, L., Wu, Y., Singh, R., Joppa, L., and Fang, F. (2019). Deep reinforcement learning for green security games with real-time information. In *33rd AAAI Conference on Artificial Intelligence*, pages 1401–1408.

[Warnquist et al., 2013] Warnquist, H., Kvarnström, J., and Doherty, P. (2013). Exploiting Fully Observable and Deterministic Structures in Goal POMDPs. In *23rd International Conference on Automated Planning and Scheduling (ICAPS)*, pages 242–250.

[Wiggers et al., 2016] Wiggers, A. J., Oliehoek, F. A., and Roijers, D. M. (2016). Structure in the value function of two-player zero-sum games of incomplete information. In *22nd European Conference on Artificial Intelligence (ECAI)*, pages 1628–1629.

[Wikipedia, 2019] Wikipedia (2019, accessed December 15, 2019). Scotland Yard (board game). `https://en.wikipedia.org/wiki/Scotland_Yard_(board_game)`.

[Wood, 2015] Wood, J. (2015). Doug Polk and Team Beat Claudico to Win $100,000 from Microsoft & the Rivers Casino. `https://pokerfuse.com/news/media-and-software/26854-doug-polk-and-team-beat-claudico-win-100000-microsoft/`.

[Yang et al., 2014] Yang, R., Ford, B., Tambe, M., and Lemieux, A. (2014). Adaptive resource allocation for wildlife protection against illegal poachers. In *13th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 453–460.

[Zhang and Zhang, 2001] Zhang, N. L. and Zhang, W. (2001). Speeding up the convergence of value iteration in partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 14:29–51.

[Zhang et al., 2019] Zhang, X., Zhang, K., Miehling, E., and Basar, T. (2019). Non-Cooperative Inverse Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9482–9493.

[Zhou et al., 2010] Zhou, E., Fu, M. C., and Marcus, S. I. (2010). Solving continuous-state POMDPs via density projection. *IEEE Transactions on Automatic Control*, 55(5):1101–1116.

[Zhu et al., 2012] Zhu, Q., Clark, A., Poovendran, R., and Başar, T. (2012). Deceptive routing games. In *51st IEEE Annual Conference on Decision and Control*, pages 2704–2711.

[Zhu et al., 2013] Zhu, Q., Clark, A., Poovendran, R., and Basar, T. (2013). Deployment and exploitation of deceptive honeybots in social networks. In *IEEE 52nd Annual Conference on Decision and Control*, pages 212–219.

[Zinkevich et al., 2008] Zinkevich, M., Johanson, M., Bowling, M., and Piccione, C. (2008). Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1729–1736.