Structure in the Value Function
of Two-Player Zero-Sum Games
of Incomplete Information

Auke Wiggers
6036163
July 2015
42 EC

Supervisors:
Dr. Frans A. Oliehoek
Diederik M. Roijers MSc

Assessors:
Dr. Joris M. Mooij

Intelligent Autonomous Systems group
Abstract

Decision-making in competitive games with incomplete information is a field with many promising applications for AI, both in games (e.g. poker) and in real-life settings (e.g. security). The most general game-theoretic framework that can be used model such games is the zero-sum Partially Observable Stochastic Game (zs-POSG). While use of this model enables agents to make rational decisions, reasoning about them is challenging: in order to act rationally in a zs-POSG, agents must consider stochastic policies (of which there are infinitely many), and they must take uncertainty about the environment as well as uncertainty about their opponents into account.

We aim to make reasoning about this class of models more tractable. We take inspiration from work from the collaborative multi-agent setting, where so-called plan-time sufficient statistics, representing probability distributions over joint sets of private information, have been shown to allow for a reduction from a decentralized model to a centralized one. This leads to increases in scalability, and allows the use of (adapted) solution methods for centralized models in the decentralized setting. We adapt these plan-time sufficient statistics for use in the competitive setting. Not only does this enable reduction from a (decentralized) zs-POSG to a (centralized) Stochastic Game, it turns out that the value function of the zs-POSG, when defined in terms of this new statistic, exhibits a particular concave/convex structure that is similar to the structure found in the collaborative setting. We propose an anytime algorithm that aims to exploit the found structure in order to find bounds on the value function, and evaluate performance of this methods in two domains of our design. As it does not outperform existing solution methods, we analyze its shortcomings, and give possible directions for future research.
Acknowledgements

I would like to thank my supervisors, Frans Oliehoek and Diederik Roijers, for their valuable insights and the encouragement they provided. Without our (often lengthy) discussions and meetings, this thesis would not be what it is now. Furthermore, their combined effort enabled me to publish and present my work — a milestone in my academic career.

I would also like to thank my parents, sister, and girlfriend for their patience, and, most of all, for their invaluable support.
# Contents

Abstract i

Acknowledgements ii

1 Introduction 1
   1.1 Decision-Making in Competitive Games .......................... 2
   1.2 Contributions ............................................. 2
   1.3 Outline .................................................. 4

2 Background 5
   2.1 Game-Theoretic Frameworks .................................... 6
      2.1.1 Normal Form Games ........................................ 7
      2.1.2 Bayesian Games .......................................... 7
      2.1.3 Extensive Form Games ................................. 9
      2.1.4 Partially Observable Stochastic Games ................. 10
   2.2 Solution Concepts ............................................ 11
      2.2.1 Strategies .............................................. 11
      2.2.2 Nash Equilibria ........................................ 13
      2.2.3 Value of Two-Player Zero-Sum Games .................... 14
   2.3 Solution Methods ............................................. 16
      2.3.1 Normal Form Games ...................................... 16
      2.3.2 Extensive Form Games ................................... 18
         2.3.2.1 Sequence Form Representation ....................... 19
         2.3.2.2 Solving Games in Sequence Form ................... 20
      2.3.3 Bayesian Games ........................................ 22
      2.3.4 Partially Observable Stochastic Games .................. 24

3 Families of Zero-Sum Bayesian Games 25
   3.1 Framework .................................................. 26
   3.2 Structure in the Value Function ................................ 26
   3.3 Discussion .................................................. 29

4 Zero-Sum POSGs 30
   4.1 Sufficient Statistics for Decision-Making ..................... 30
   4.2 Value Function of the Zero-Sum POSG ........................ 31
      4.2.1 Value in Terms of Past Joint Policies ................. 32
      4.2.2 Value in Terms of Plan-Time Sufficient Statistics .... 33
Abbreviations and Notation

AOH  Action-observation history
BG   Bayesian Game
Dec-POMDP  Decentralized Partially Observable Markov Decision Process
EFG  Extensive Form Game
NE   Nash Equilibrium
NFG  Normal Form Game
POMDP  Partially Observable Markov Decision Process
POSG  Partially Observable Stochastic Game

Single-shot games
θ   Joint type
θi  Individual type of agent i
θ−i  Types for all agents except i
Θi  Set of types for agent i

Multi-stage games
⃗θt  Joint action-observation history (AOH) at stage t
⃗θi  Individual AOH of agent i at stage t
⃗Θi  Set of AOHs for agent i at stage t

Policies and decision rules
δ(a | θ)  Probability of action a given stochastic decision rule δ conditioned on θ
π(a′ | ⃗θ)  Probability of action a′ given stochastic policy π conditioned on ⃗θ
Chapter 1

Introduction

“To know your enemy, you must become your enemy.”

Sun Tzu, The Art of War

Humans are quite capable when it comes to making decisions in competitive games, whether it concerns a task in a complex, dynamic environment, or a simple game of tic-tac-toe. We are, generally speaking, able to reason about a task and the consequences of our actions, but also about the opponent. Paraphrasing, the quote by Sun Tzu stated above tells us that ‘to know what your opponent will do, you must view the world from their perspective’. Of course, it is a bit extreme to compare a game of tic-tac-toe to warfare, but the idea is valid in any competitive game: to determine what the opponent will do next, we can put ourselves in the opponent’s shoes and try to answer the question ‘what would I do in this situation?’.

Computer systems nowadays are able to outperform humans in many competitive games using approaches based on this idea. A well-known example is the game of chess, in whichHowever, in the game of poker, in which a similar strategy should work, expert human players far outperform computers [Rubin and Watson, 2011]. An important reason for this is that the game is partially observable: the agents only hold some private information (their cards) and do not know the real state of the world. A factor that further complicates decision-making is that agents cannot only influence the future state of the environment through their own actions, but also what they will observe, as well as what other agents will observe. Thus, in order to win the game, the agents will have take their own uncertainty about the state of the environment into account, but also uncertainty regarding the opposing agents.
1.1 Decision-Making in Competitive Games

In so-called strictly competitive games, each agent wants to maximize the reward that his opponents are actively trying to minimize. Behaving rationally in such games typically requires the agent to follow a stochastic strategy, which specifies that actions should be taken with a certain probability. Following such a strategy, rather than a deterministic one, ensures that the agent cannot be exploited by the opponent, even if the opponent were to learn his strategy. In Texas hold-em poker, for example, a stochastic strategy could specify ‘if I get dealt a Jack of clubs and a Jack of spades, I bet with probability 0.7, and fold with probability 0.3’. If instead, an agent always bets if he receives these cards, the opponent could gain information from the chosen action and use that information to their advantage.

A subset of strictly competitive games is the set of two-player games in which the rewards for both agents sum to zero, appropriately named two-player zero-sum games. In this work, we will investigate the problem of finding the rational joint strategy (i.e., a tuple containing, for each agent, the strategy that maximizes their individual expected payoff) in two-player zero-sum games of incomplete information. While it is imaginable that each game requires a custom solution method, this is simply not feasible in practice. Instead, games are typically modeled using standard frameworks, so that standardized solution methods can be used.

Arguably the most general framework that can be used to model two-player zero-sum games is the zero-sum Partially Observable Stochastic Game (POSG), which describes the problem of decision-making under uncertainty in a multi-agent zero-sum game of one or more rounds. The uncertainty in the POSG stems from both the hidden state of the game, which is dynamic (i.e., it may change over time), and from the fact that agents do not observe the actions of the opponent. Furthermore, communication is not available, making this a decentralized decision-making problem. At every stage of the game, both agents simultaneously choose an action. Their choices affect the state according to a predefined (possibly stochastic) transition function. The agents then receive individual, private observations — typically a noisy signal about the state of the environment. Although the framework is able to model games that go on indefinitely, we will focus on the finite horizon case, meaning that the game ends after a finite number of rounds.

1.2 Contributions

In this work, we prove the existence of theoretical properties of two-player, zero-sum Partially Observable Stochastic Games of finite horizon. We take inspiration from recent work for collaborative settings which has shown that it is possible to summarize the past joint policy using so-called plan-time sufficient statistics [Oliehoek, 2013], which can be interpreted as the belief of a special type of Partially Observable Markov Decision Process.
Chapter 1. Introduction

Contributions

(POMDP) to which the collaborative Decentralized POMDP (Dec-POMDP) can be reduced [Dibangoye et al., 2013; MacDermed and Isbell, 2013; Nayyar et al., 2013]. Use of these statistics allows Dec-POMDPs to be solved using (adapted) solution methods for POMDPs that exploit the structural properties of the POMDP value function, leading to increases in scalability [Dibangoye et al., 2013].

In this thesis, we adapt the plan-time statistics from Oliehoek [2013] for use in the zero-sum POSG setting, with that hope that this makes reasoning about zero-sum games more tractable. In particular, we aim to provide insight into the structure the value function of the zero-sum POSG at every stage of the game, so that approaches that treat the POSG as a sequence of smaller problems may be used — an idea that has been applied successfully in the collaborative setting [Emery-Montemerlo et al., 2004; MacDermed and Isbell, 2013]. We give a value function formulation for the zero-sum POSG in terms of these statistics, and will try to answer the following questions:

1. Can we extend the structural results from the Dec-POMDP setting to the zero-sum POSG setting using plan-time sufficient statistics?
2. If so, can the structure of the value function be exploited in order to find the value of the zero-sum POSG?

We first consider a simple version of the problem: a zero-sum game of incomplete information that ends after one round, in which the probability distribution over hidden states is not known beforehand. We introduce a framework for such games called the Family of zero-sum Bayesian Games, and give a value function formulation in terms of the probability distribution over hidden states. We give formal proof that this value function exhibits a particular structure: it is concave in the marginal-space of the maximizing agent (i.e., the space spanning all distributions over private information of this agent), and convex in the marginal-space of the minimizing agent.

We then extend this structural result to the (multi-stage) zero-sum POSG setting. First, we show that the plan-time statistic as defined by [Oliehoek, 2013] provides sufficient information for rational decision-making in the zero-sum POSG setting as well. We give a value function formulation for the zero-sum POSG in terms of this statistic. This allows us to show that the final stage of the zero-sum POSG is equivalent to a Family of zero-sum Bayesian Games, thereby proving that the concave and convex properties hold as well. Even though the preceding stages of the zero-sum POSG cannot be modeled as Families of zero-sum Bayesian Games, we prove that the structural result found for the final stage holds for all other stages.

Furthermore, we show that the use of plan-time sufficient statistics allows for a reduction from a zero-sum POSG to a special type of zero-sum stochastic game in which the agents do not receive observations, which we call the Non-Observable Stochastic Game — indicating that theory for such stochastic games may extend to the zero-sum POSG setting directly.
We propose a method that aims to exploit the found concave and convex structure. Without going into too much detail, this method performs a heuristic search in a subspace of statistic-space that we call conditional-space through identification of promising one-stage policies. It computes value vectors at every stage of the game, which can be used to construct concave upper bounds and convex lower bounds on the value function. We propose two domains based on an existing benchmark problem from the collaborative setting, and compare the performance of our method to a random baseline and an existing solution method that solves the zero-sum POSG using sequence form representation [Koller et al., 1994]. As we find that our method does not outperform this existing solution method in terms of runtime or scalability, we analyze its shortcomings.

To validate the idea of performing a heuristic search in conditional-space, we use a different solution method called Nash memory for asymmetric games [Oliehoek, 2005] (‘Nash memory’ for short) as heuristic for the selection of one-stage policies. Motivation for this choice is that Nash memory iteratively identifies promising multi-stage policies, and that it guarantees convergence, i.e., it finds the rational strategies after a finite amount of iterations. Despite these properties, this heuristic search scales worse than the method that uses sequence form representation. It does, however, provide

Nonetheless, we hope that our theoretical findings open up the route for effective solution methods that search conditional-space, and we provide directions for future research.

1.3 Outline

This thesis is organized as follows. In chapter 2, we provide background on the game-theoretic frameworks commonly used to model multi-agent games, and we specify notation and terminology. We also discuss solution concepts for zero-sum games, and explain existing solution methods for the zero-sum variants of the various frameworks. In chapters 3 and 4, we present our theoretical results for respectively Families of zero-sum Bayesian Games and zero-sum POSGs. In chapter 5, we present our algorithm and give experimental results. Based on further analysis of the results, we perform a different experiment and give directions for future research in chapter 6. We discuss closely related literature in chapter 7. Chapter 8 summarizes and concludes this work.
Chapter 2

Background

The field of game theory studies the problem of decision-making in games played by multiple players or agents. In this chapter, we define the game-theoretic frameworks and solution concepts that are generally used to solve this problem.

We distinguish between collaborative games, in which it is in the agents’ best interest to cooperate, and non-collaborative games. A specific subset of non-collaborative games is the set of zero-sum games, a class that we will describe in more detail in section 2.2.3.

We formally define various game-theoretic frameworks that can be used to model multi-agent games in section 2.1. In section 2.2, we discuss established solution concepts and specify what it means to solve a zero-sum game. Finally, in section 2.3, we explain solution methods for zero-sum variants of the given frameworks. As the focus of this work will be on the zero-sum setting, we will not explain the solution methods for other categories of games.

Throughout this work, we make the following three assumptions.

**Assumption 1:** Agents are rational, meaning that they aim to maximize individual payoff [Osborne and Rubinstein, 1994].

**Assumption 2:** Agents have perfect recall, meaning that they are able to recall all past individual actions and observations.

**Assumption 3:** All elements of the game are common knowledge: there is common knowledge of $p$ if all agents know $p$, all agents know that all agents know $p$, they all know that they all know that they know $p$, ad infinitum [Osborne and Rubinstein, 1994].

There is literature on games in which one or more of these assumptions does not hold, for example, games where agents have imperfect recall [Piccione and Rubinstein, 1997], or games in which elements of the game are almost common knowledge [Rubinstein, 1989]. However, these games are outside the scope of this work.

---

1In the next section, we make clear what we mean by the term ‘elements of a game’.
2.1 Game-Theoretic Frameworks

A game-theoretic framework is a formal representation of a game. Essentially, it is a
description of the properties and rules of the game, but in a standard format, which enables
the use of standardized solution methods. We focus on two properties: whether the game is
one-shot or multi-stage, and whether there is partial observability or not. One-shot games
are games that end after one round, i.e., agents choose a single action and receive payoff
directly. Multi-stage games are, as the name implies, games of multiple stages or timesteps.
For example, rock-paper-scissors is a one-shot game, whereas chess is a multi-stage game.
Partial observability, in the game-theoretic sense, means that an agent does not observe
the state of the game directly. For example, in the game of chess, the state corresponds to
the positioning of the pieces on the board. It is a fully observable game, as the players
observe the positions of the pieces directly (and without noise). In Texas hold-em poker,
the state describes the cards that each player has in hand, the cards that are on the table,
and the cards are still in the deck. As the players only have partial information about this
state, it is a partially observable game.

The main focus of this work is on a two-player zero-sum variant of a framework known
as the Partially Observable Stochastic Game (POSG). We show various types of games
and the game-theoretic frameworks that can be used to model these games in Figure 2.1.1.
Note, that the zero-sum POSG framework, denoted as zs-POSG, is a specific instance of
the more general POSG model. Similarly, if the agents in a POSG have the same reward
function, the POSG reduces to a collaborative model for multi-stage games called the
Decentralized POMDP [Bernstein et al., 2002; Oliehoek, 2012].

In this section, we formally define some of the depicted game-theoretic frameworks,
starting with the least general one (the Normal Form Game, or NFG) and ending with the
most general one (the POSG). We do not make the distinction between zero-sum games
and other classes of games explicit yet, e.g., we define the POSG framework and not the
zero-sum POSG framework.
2.1.1 Normal Form Games

The simplest version of a multi-agent game is the fully observable one-shot game, which is typically modeled as a Normal Form Game (NFG), sometimes also referred to as strategic game. It is defined as follows.

**Game 2.1.1.** The *Normal Form Game* is a tuple \( \langle I, A, R \rangle \):

- \( I = \{1, \ldots, n\} \) is the set of agents,
- \( A = A_1 \times \ldots \times A_n \) is the set of joint actions \( a = \langle a_1, \ldots, a_n \rangle \),
- \( R : A \to \mathbb{R}^n \) is a reward function mapping joint actions to payoff for each agent.

In the NFG, each agent \( i \in I \) selects an action from \( A_i \), simultaneously. The agents then receive payoff according to the reward function \( R(a) \). By assumption 3, the elements of the NFG \(( I, A \) and \( R \)) are common knowledge to the agents.

A well-known example of a Normal Form Game is the game ‘Matching Pennies’. In this game, two agents must choose one side of a penny, and subsequently show it to the other agent. If the sides match, agent 1 wins. If not, agent 2 wins. The payoff matrix is shown in table 2.1.1, with the payoff shown as a tuple containing reward for agent 1 and 2, respectively. It is a strictly competitive game, i.e., if one agent wins the game, the other agent loses. More specifically, it is a so-called *zero-sum game*, as the sum of the two payoffs is zero for all joint actions. This concept will be explained in more detail in section 2.2.3.

Note, that the assumption of perfect recall (assumption 2) is obvious, as there are no past events in a one-shot game. The assumption of common knowledge (assumption 3), however, is crucial: if the agents do not know the payoff matrix, then the game cannot be modeled as a NFG.

While the NFG framework is explicitly defined for fully observable one-shot games, we will show that it is possible to convert partially observable games and even multi-stage games to Normal Form. However, the resulting Normal Form payoff matrix will typically be so large that it is impractical to do so.

2.1.2 Bayesian Games

If agents play a one-shot game with an underlying hidden state that affects the payoff, and one or more agents do not observe this hidden state directly, then we speak of a game of *imperfect information*. In such games, agents may receive private information about the hidden state, which we refer to as their *type*. Reasoning about games in which agents do not know the real payoff function, however, is generally difficult.

If the agents know a probability distribution over *joint types* (i.e., hidden states), then we can model the problem as a game of *incomplete information* instead, by introducing an additional agent, ‘nature’, that decides on the hidden state at the start of the game [Harsanyi, 1995]. In games of imperfect information, the agents do not observe all actions
and chance moves, meaning that they must reason about this missing information. This idea is captured in the Bayesian Game (BG) framework, which is defined as follows.

**Game 2.1.2.** The *Bayesian Game* is a tuple \(\langle I, \Theta, A, \sigma, R \rangle\):

- \(I = \{1, \ldots, n\}\) is the set of agents,
- \(\Theta = \Theta_1 \times \ldots \times \Theta_n\) is the set of joint types \(\theta = \langle \theta_1, \ldots, \theta_n \rangle\),
- \(A = A_1 \times \ldots \times A_n\) is the set of joint actions \(a = \langle a_1 \ldots a_n \rangle\),
- \(\sigma \in \Delta(\Theta)\) is the probability distribution over joint types,
- \(R: \Theta \times A \to \mathbb{R}^n\) is a reward function mapping joint types and joint actions to payoff for each agent.

Here, \(\Delta(\Theta)\) denotes the *simplex* over the set of joint types.

At the start of the game, ‘nature’ chooses a joint type \(\theta\) from \(\Theta\) according to \(\sigma\). Each agent \(i \in I\) observes his individual type \(\theta_i\), and picks action \(a_i\) from \(A_i\) (simultaneously). The agents then receive payoff according to the reward function as \(R(\theta, a)\).

A payoff matrix for a two-player Bayesian Game is shown in Table 2.1.2. The entries of the matrix contain the reward for agents 1 and 2, respectively. For example, if the agents receive the observation \(\bar{\theta}_1\) and \(\bar{\theta}_2\), and they choose the joint action \(\langle \bar{a}_1, \bar{a}_2 \rangle\), then agent 1 will receive a reward of 6, while agent 2 will receive a reward of 6.

This payoff matrix can be seen as a collection of Normal Form payoff matrices, one for each joint type. For example, if the joint type happens to be \(\langle \bar{\theta}_1, \bar{\theta}_2 \rangle\), then the agents are actually playing the Normal Form Game in the bottom right corner of Table 2.1.2. However, they do not know this when they make their decisions: P1 observes \(\bar{\theta}_1\), and therefore only knows that the agents are playing either of the games in the bottom row. P2 observes \(\bar{\theta}_2\), and therefore only knows that the agents are playing either of the games in the second column.

What makes decision-making in BGs difficult is that agents have to account for their own uncertainty over types and the uncertainty of the other agents: in the example scenario, P1 should take into account that P2 does not know the observed type of P1 \(\bar{\theta}_1\).

<table>
<thead>
<tr>
<th>P1</th>
<th>$\bar{\theta}_1$</th>
<th>$\bar{a}_1$</th>
<th>$\bar{a}_1'$</th>
<th>$\bar{a}_2$</th>
<th>$\bar{a}_2'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>head</td>
<td>(3, -3)</td>
<td>(6, -6)</td>
<td>(3, 3)</td>
<td>(1, 1)</td>
<td></td>
</tr>
<tr>
<td>tail</td>
<td>(-3, 3)</td>
<td>(-7, 7)</td>
<td>(-3, 3)</td>
<td>(-7, 7)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P1</th>
<th>$\bar{\theta}_1$</th>
<th>$\bar{a}_1$</th>
<th>$\bar{a}_1'$</th>
<th>$\bar{a}_2$</th>
<th>$\bar{a}_2'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>head</td>
<td>(3, -3)</td>
<td>(6, -6)</td>
<td>(3, 3)</td>
<td>(1, 1)</td>
<td></td>
</tr>
<tr>
<td>tail</td>
<td>(-3, 3)</td>
<td>(-7, 7)</td>
<td>(-3, 3)</td>
<td>(-7, 7)</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1.1: Payoff matrix for the Matching Pennies game.

Table 2.1.2: Payoff matrix for an example Bayesian Game two-player game with two types and two actions per agent.
2.1.3 Extensive Form Games

A game-theoretic framework that can be used to model \textit{sequential} games, i.e., games of more than one timestep, is the Extensive Form Game (EFG). In contrast to the one-shot Normal Form Game and Bayesian Game frameworks, the EFG allows us to model games of multiple stages. It represents the decision-making problem as a game tree: a directed, acyclic graph, in which vertices correspond to so-called \textit{decision nodes}, and edges correspond to the agents’ choices.

There are two types of non-terminal nodes in the Extensive Form game tree. The first type is the \textit{decision node}. At a decision node, a single agent selects an edge (corresponding to an action) that leads to a following node. The second type is the \textit{chance node}, in which the selected edge is chosen according to some prespecified probability distribution. We model chance nodes as decision nodes for a special agent ‘nature’. Terminal nodes in the game tree correspond to an ‘outcome’, which in turn corresponds to a payoff for all agents.

We restate that it is assumed that agents have \textit{perfect recall} (assumption 2). In the case of partial observability, for example when agents do not observe the choices made by other agents or ‘nature’, it may be that agent $i$ cannot discriminate between a particular set of decision nodes. We refer to such a set of nodes as an \textit{information set} for agent $i$.

The EFG framework is formally defined as follows.

\textbf{Game 2.1.3.} The \textit{Extensive Form Game} is a tuple $\langle I, K, B, A, H, R, k_{\text{root}} \rangle$:

- $I = \{\text{nature}, 1, \ldots, n\}$ is the set of $n + 1$ agents, including ‘nature’,
- $K = K^d \cup K^o$ is the set of all nodes in the game tree, where $K^d$ is the set of decision nodes, $K^o$ is the set of outcome nodes, and $K^d \cap K^o = \emptyset$,
- $B = B_1 \times \ldots \times B_n$ is a set containing for each agent $i \in I$ the set of decision nodes $b_i \in K^d$,
- $B_{\text{nature}} \subseteq K^d$ is the set of chance nodes, each of which specifies a probability distribution over the outgoing edges,
- $A_i^b$ is the set of edges specifying transitions from decision nodes $b_i$ to other decision nodes or outcome nodes, for all $b_i \in B_i$,
- $H = H_1 \times \ldots \times H_n$ is a set of information sets $h_i$ for each agent $i \in I$,
- $R : K^o \rightarrow \mathbb{R}^n$ is the reward function that specifies the payoff associated with an outcome node for every agent,
- $k_{\text{root}}$ is the root node.

The game starts at the root node $k_{\text{root}}$, which is a decision node for one of the agents (we say that chance nodes are decision nodes for the agent ‘nature’). The corresponding agent selects one of the outgoing edges, which is followed to the next node. This next node is a decision node of a different agent or nature. This process is repeated until a terminal node is reached, where the agents receive payoff according to the reward function.
Examples are shown in Figures 2.1.2a and 2.1.2b. In both games, two agents (P1 and P2) choose a single action, after which the game ends and they receive their individual payoff. At the leaf nodes, a tuple shows the payoff for P1 and P2 respectively. In the first example, the game tree represents a game in which agents choose their actions sequentially: P1 chooses first, and P2 observes this choice before selecting an action. In Figure 2.1.2b, P2 can no longer observe the choice made by P1. As a result, he cannot distinguish between the two decision nodes: both nodes are contained in a single information set for P2, indicated with dashed lines. Note, that the game tree in 2.1.2b is the Extensive Form representation of the 'Matching Pennies' game from section 2.1.1.

![Game Trees](image)

Figure 2.1.2: Game trees for two example Extensive Form Games.

### 2.1.4 Partially Observable Stochastic Games

The Partially Observable Stochastic Game (POSG) is a game-theoretic framework that describes the problem of rational decision-making under uncertainty in a multi-stage game. A key property of such a problem is that communication is not available, making this a decentralized decision-making problem. In the general POSG model, actions are selected simultaneously, and agents receive private observations.

**Game 2.1.4.** The *Partially Observable Stochastic Game* is a tuple \( \langle I, S, A, T, R, O, O, h, b^0 \rangle \):

- \( I = \{1, \ldots, n\} \) is the set of agents,
- \( S \) is the finite set of states \( s \),
- \( A = A_1 \times \ldots \times A_n \) is the set of joint actions \( a = \langle a_1, \ldots, a_n \rangle \),
- \( T \) is the transition function that specifies \( \Pr(s^{t+1} | s^t, a^t) \),
- \( R : S \times A \rightarrow \mathbb{R}^n \) is the reward function mapping states and actions to the payoff for each agent,
- \( O = O_1 \times \ldots \times O_n \) is the set of joint observations \( o = \langle o_1, \ldots, o_n \rangle \),
- \( O \) is the observation function that specifies \( \Pr(o^{t+1} | a^t, s^{t+1}) \),
- \( h \) is the horizon of the problem, which we assume to be a finite number,
- \( b^0 \in \Delta(S) \) is the initial probability distribution over states at \( t = 0 \).

On every stage \( t \) of the game, agents simultaneously select an action \( a^t_i \) from \( A_i \). The state is updated according the transition function \( T \). Based on the observation function,
each agent receives a private, individual observation $o_i^t \in O_i$. The agents accumulate the reward they receive at every stage of the game (reward is specified by the reward function $R$). This process repeats until the horizon is reached, after which agents receive their accumulated reward. By assumption 2, agents can collect their past actions and observations in the so-called Action-Observation History (AOH), $\vec{\theta}^t_i = \langle a_0^i, o_0^i, \ldots, a_{t-1}^i, o_t^i \rangle$. This is the only information the agents have about the game. The joint AOH is a tuple containing all individual AOHs $\vec{\theta} = \langle \vec{\theta}_1^t, \ldots, \vec{\theta}_n^t \rangle$.

It is possible to transform a POSG to Extensive Form and vice versa [Oliehoek and Vlassis, 2006]. It turns out that every pair $(\vec{\theta}_i^t, s^t)$ corresponds to a decision node for one of the agents in Extensive Form representation. If an agent $i$ cannot distinguish between two AOHs $(\vec{\theta}_i^t, \vec{\theta}_j^t)$ and $(\vec{\theta}_i^t, \vec{\theta}_j')$ where $i \neq j, \vec{\theta}_j^t \neq \vec{\theta}_j'$, then this individual AOH $\vec{\theta}_i^t$ induces exactly one information set. We will make use of this fact in section 2.3.4, where we explain how to solve a POSG by converting it to Extensive Form.

## 2.2 Solution Concepts

In Section 2.1 we defined the game-theoretic frameworks used to model various types of multi-agent games. In this section, we explain what it means to solve a game (in the game-theoretic sense). Generally, the solution of a game contains the rational joint strategy, that is, a tuple specifying the rational strategy for every agent.

### 2.2.1 Strategies

In any game, the goal of a rational and strategic agent is to find their rational strategy, i.e., is the strategy that maximizes their individual reward. Here, we give a formal definition of the concept. We divide strategies into three classes: pure, stochastic and mixed strategies.

**Definition** In an Extensive Form Game, an individual pure strategy for agent $i$, $\hat{\pi}_i$, is a mapping from information sets to actions. In the Normal Form Game, Bayesian Game, and Partially Observable Stochastic Game, a pure strategy specifies one action $a_i \in A_i$ for each situation agent $i$ can face.

**Definition** A individual mixed strategy $\mu_i$ specifies a probability distribution over pure strategies $\hat{\pi}_i$. The set of pure policies to which $\mu$ assigns positive probability is referred to as the support of $\mu_i$.

**Definition** In an Extensive Form Game, an individual stochastic strategy for agent $i$, $\pi_i$, is a mapping from information sets to probability distributions over actions. In the Normal Form Game, Bayesian Game, and Partially Observable Stochastic Game, a stochastic strategy specifies a probability distribution over actions $A_i$ for each situation agent $i$ can face.
From now on, we will refer to strategies in the one-shot setting as decision rules, denoted as $\delta$, while we will refer to strategies in multi-stage games as policies, and denote these as $\pi$. This distinction will prove useful in later chapters.

In the NFG, the game ends after the agents make a single decision, there is only one real ‘situation’: the start of the game. Therefore, an individual pure strategy is a decision rule $\delta_i$, and it maps to a single action. An individual stochastic decision rule is a mapping from actions to probabilities, denoted as $\hat{\delta}_i(a_i)$.

In the BG, each individual type $\theta_i \in \Theta_i$ induces a different situation for agent $i$, so a pure individual decision rule is a mapping from types to actions, denoted as $\delta_i(\theta_i)$. A stochastic decision rule is a mapping from types to a probability distribution over actions, denoted as $\delta_i(a_i | \theta_i)$. Note, that there are actually many more situations (one for each joint type $\theta$), but that agent $i$ does not observe the types of other agents, and therefore can only distinguish between situations in which their individual type is different.

In the POSG, the situation for agent $i$ is determined by his private information: the individual AOH $\vec{\theta}_t^i$. A pure individual policy is a mapping from individual AOHs to actions, denoted as $\pi_i(\vec{\theta}_t^i)$. A stochastic policy maps from individual AOHs to a probability distribution over actions, denoted as $\pi_i(a_i | \vec{\theta}_t^i)$. Interestingly, a policy in the POSG can be represented as a tuple of decision rules, one for every timestep: $\pi_i = (\delta_0^i, \ldots, \delta_{h-1}^i)$. We will further distinguish between the past individual policy, defined as the tuple of decision rules from stage 0 to $t$ as $\varphi^t_i = (\delta_0^i, \ldots, \delta_{t-1}^i)$, and what we call a partial individual policy, defined as the tuple of decision rules from stage $t$ to $h - 1$ as $\pi^t_i = (\delta_t^i, \ldots, \delta_{h-1}^i)$.

Joint strategies are defined as a tuple containing individual strategies for all agents, e.g., $\pi = (\pi_0, \ldots, \pi_n)$.

Any finite game-theoretic framework can be converted to a Normal Form Game by enumerating all pure policies available for the players. The Normal Form payoff matrix then specifies, for each pure joint policy, the expected payoff attained when following this policy. For example, to convert a two-player game to normal form, we let the rows of the payoff matrix correspond to pure policies of agent 1, and the columns to pure policies of agent 2. Entries in the payoff matrix are then the utilities associated with the resulting pure joint policies.

There are two disadvantages to this approach, however. First, we lose any information about the structure of the game. For example, if we convert an EFG in which action-selection is not simultaneous to normal form, we cannot infer who moves first from the resulting NFG. Second, the number of pure policies grows exponentially with the number of possible situations and actions. For example, in the finite-horizon POSG, the number of AOHs at a stage is already exponential in the number of joint actions and joint observations. This exponential blow-up generally prevents solution methods for NFGs to be applied to more complex games.
2.2.2 Nash Equilibria

The Nash Equilibrium (NE) is a solution concept for game-theoretic frameworks that specifies the rational joint strategy. Let a joint strategy be defined as a tuple containing all individual strategies. Let \( u_i \) be defined as the utility function for agent \( i \), that maps a joint strategy to a payoff. Let \( \hat{\Pi}_i \) be the (finite) set of pure joint policies for agent \( i \), and let \( \hat{\Pi} = \hat{\Pi}_0 \times \ldots \times \hat{\Pi}_n \). The pure NE is defined as follows.

**Definition** A pure Nash Equilibrium is a set of pure policies from which no agent has an incentive to unilaterally deviate:

\[
u_i(\langle \hat{\pi}_i, \hat{\pi}_{-i} \rangle) \geq \nu_i(\langle \hat{\pi'}_i, \hat{\pi}_{-i} \rangle) \quad \forall i \in I, \forall \hat{\pi'}_i \in \hat{\Pi}_i.
\]

Here, \( \hat{\pi}_i \in \hat{\Pi}_i \), and \( \hat{\pi}_{-i} = \{ \hat{\pi}_j : j \neq i, \forall j \in I \} \).

In other words, a pure Nash Equilibrium specifies a set of pure policies that guarantees the highest possible individual payoff for every agent.

This can be extended to the case of mixed strategies \( \mu_i \) as follows:

\[
u_i(\langle \mu_i, \mu_{-i} \rangle) = \sum_{\hat{\pi} \in \hat{\Pi}} \nu_i(\hat{\pi}) \prod_{j \in I} \mu_j(\pi_j).
\]

This is referred to as a mixed Nash Equilibrium\(^2\). Nash [1951] showed that any NFG with a finite number of agents and a finite number of actions always has at least one mixed NE.

A second, arguably more intuitive definition of a pure NE can be given in terms of best-response functions, which are defined as follows:

**Definition** The best-response function \( B_i \) for agent \( i \) is a mapping from a joint strategy \( \hat{\pi} = \langle \hat{\pi}_1, \ldots, \hat{\pi}_{i-1}, \hat{\pi}_{i+1}, \ldots, \hat{\pi}_n \rangle \) to a set of individual pure policies \( \hat{\Pi}_i \), from which agent \( i \) has no incentive to unilaterally deviate, given that the other agents follow the actions specified in \( \hat{\pi}_{-i} \):

\[
B_i(\hat{\pi}_{-i}) = \{ \hat{\pi}_i \in \hat{\Pi}_i : \nu_i(\langle \hat{\pi}_i, \hat{\pi}_{-i} \rangle) \geq \nu_i(\langle \hat{\pi'}_i, \hat{\pi}_{-i} \rangle), \forall \hat{\pi'}_i \in \hat{\Pi}_i \}
\]

A pure Nash Equilibrium is a tuple of pure strategies \( \hat{\pi}_i \) for which the property \( \hat{\pi}_i \in B_i(\pi_{-i}) \) \( \forall i \in I \) holds.

If a solution for a game exists, the solution is a Nash Equilibrium, and vice versa. Note, that a game can have multiple Nash Equilibria, but that when the action set is unbounded, continuous, or both, a solution may not exist [Dasgupta and Maskin, 1986]. As the focus of this work is on games that have finite action sets, we will not discuss unsolvable games in detail.

\(^2\)In the BG setting, the NE is referred to as the Bayesian Nash Equilibrium (BNE) [Harsanyi, 1995].
Chapter 2. Background Solution Concepts

2.2.3 Value of Two-Player Zero-Sum Games

Now that we have a clear definition of the solution of a game, we will show how it can be used to find the *value* in the strictly competitive setting. More specifically, we will focus on the two-player zero-sum game, which is a strictly competitive two-player game in which the rewards for both agents sum to zero. Assume that the reward function \( R \) can be split into components for agent 1 and agent 2 as \( R_1 \) and \( R_2 \). A game is zero-sum if the following holds:

\[
R_1(s^t, a^t) + R_2(s^t, a^t) = 0, \quad \forall s^t \in \mathcal{S}, a^t \in \mathcal{A}.
\]

By convention, let agent 1 be the maximizing player, and let agent 2 be the minimizing player. For the maximizing player, a rational strategy in a zero-sum game is a maxmin-strategy.

**Definition** A *maxmin-strategy* for agent 1 is the strategy that gives the highest payoff for agent 1, given that agent 2 aims to minimize it.

\[
\pi^*_1 \triangleq \arg\max_{\pi_1} \min_{\pi_2} u_1(\langle \pi_1, \pi_2 \rangle).
\]

Analogously, we can define a minmax-strategy for the agent 2.

**Definition** A *minmax-strategy* for agent 2 is the strategy that gives the lowest payoff for agent 1, given that agent 1 aims to maximize it.

\[
\pi^*_2 \triangleq \arg\min_{\pi_2} \max_{\pi_1} u_1(\langle \pi_1, \pi_2 \rangle).
\]

As the payoff for agent 2 is the additive inverse of the payoff for agent 1, a minmax-strategy from the perspective of agent 2 is a maxmin-strategy: by minimizing the payoff for agent 1, we are maximizing the payoff for agent 2 (and vice versa). Typically, only the reward for the maximizing agent is shown, as this is sufficient to determine the payoff for both agents.

A Nash Equilibrium in a two-player zero-sum game is a pair containing a maxmin-strategy and a minmax-strategy, \( \langle \pi^*_1, \pi^*_2 \rangle \). We can now find the maxmin- and minmax-value of the game as follows.

**Definition** The maxmin-value of a game is the value is the most payoff that the maximizing agent can ensure, without making any assumptions about the strategy of the minimizing agent:

\[
\max_{\pi_1} \min_{\pi_2} u_1(\langle \pi_1, \pi_2 \rangle).
\]

**Definition** The minmax-value of a game is the least payoff (for the maximizing agent) that the minimizing agent can ensure without making any assumptions about the strategy of the maximizing agent:

\[
\min_{\pi_2} \max_{\pi_1} u_1(\langle \pi_1, \pi_2 \rangle).
\]
We make use of the minmax-theorem\textsuperscript{3} by Von Neumann and Morgenstern [2007].

**Theorem 2.2.1.** In any finite, two-player zero-sum game, in any Nash equilibrium each player receives a payoff that is equal to both his maxmin-value and his minmax-value:

\[
\max_{\pi_1} \min_{\pi_2} u_1(\langle \pi_1, \pi_2 \rangle) = \min_{\pi_2} \max_{\pi_1} u_1(\langle \pi_1, \pi_2 \rangle).
\]

Let us define the value of a zero-sum game as follows.

**Definition** The value of the zero-sum game is the value attained when the maximizing agent follows a maxmin-strategy and the minimizing agent follows a minmax-strategy:

\[
V \triangleq u_1(\langle \pi_1^*, \pi_2^* \rangle) = \max_{\pi_1} \min_{\pi_2} u_1(\langle \pi_1, \pi_2 \rangle) = \min_{\pi_2} \max_{\pi_1} u_1(\langle \pi_1, \pi_2 \rangle).
\]

We illustrate the above concepts by finding Nash Equilibria and the corresponding value in two example Extensive Form Games shown in Figure 2.2.1 (these are identical to the example EFGs from section 2.1.3).

In Figure 2.2.1a, P1 selects A or B, which is observed by P2, who subsequently picks C or D. P1 only has a single decision node (the root node), and P2 can condition his choice on the observed action by P1. As such, every decision node corresponds to one information set. For the rational agent 2, the best response to A is D, and the best response to B is C, as these give P2 the highest payoff. As long as agent 2 plays these best-responses, the strategy for P1 is irrelevant, as \(u_1(\langle \pi_1^*, \pi_2^* \rangle) = -1, \forall \pi_1\). Therefore, every strategy for P1 is a maxmin-strategy. The strategy in which P2 plays the best responses is a minmax-strategy. The value of this game is \(V = u_1(\langle \pi_1^*, \pi_2^* \rangle) = -1\).

In Figure 2.2.1, where P2 can no longer observe the choices by P1, the rational strategies are different from those in the previous example. For P1, the maxmin-strategy assigns each action a probability 0.5: even if P2 knew that P1 followed such a strategy, P2 would not be able to exploit it, i.e. \(u_1(\langle \pi_1, \pi_2 \rangle) = 0, \forall \pi_1, \pi_2\). Furthermore, if P1 deviates from this strategy, higher probability will be assigned to either A or B, which P2 will surely exploit. Clearly, P1 has no incentive to deviate from this strategy: it is a rational strategy. By similar logic, the rational strategy for P2 assigns 0.5 probability to C and D. Following the rational strategies gives us a probability of 0.5 \(\times 0.5 = 0.25\) of ending in each terminal node, so the value of the game is \(V = u_1(\langle \pi_1^*, \pi_2^* \rangle) = \sum_{k^o \in K^o} \Pr(k^o|\langle \pi_1^*, \pi_2^* \rangle) \cdot R(k^o) = 0.25 \times 1 + 0.25 \times -1 + 0.25 \times 1 + 0.25 \times -1 = 0\).

\textsuperscript{3}Theorem 2.2.1 is sometimes referred to as the minimax-theorem.
### Chapter 2. Background Solution Methods

In section 2.2, we explained the Nash Equilibrium and how it relates to the value of the zero-sum game. We will now discuss methods that can be used to find the NE in the zero-sum setting, for each of the game-theoretic models described in section 2.1. In particular, we will show how a mixed NE can be found using Linear Programming, a technique that has its roots in linear optimization [Von Neumann and Morgenstern, 2007; Dantzig, 1998]. A Linear Program states an objective (maximization or minimization of a certain expression), and several linear constraints. Here, we give these objectives and constraints for the NFG, BG and EFG frameworks.

A problem arises from the fact that the incentives of the competing agents cannot easily be captured in a single objective function. Instead, we find maxmin- and minmax-strategies by solving two separate Linear Programs, from the perspective of the two agents respectively.

#### 2.3.1 Normal Form Games

In the zero-sum NFG setting, a stochastic decision rule for agent $i$ specifies a probability distribution over actions as $\delta_i(a_i), \forall a_i \in A_i$. Given decision rules for both agents $\delta_1, \delta_2$, and reward function $R(\langle a_1, a_2 \rangle) = R_1(\langle a_1, a_2 \rangle) = -R_2(\langle a_1, a_2 \rangle), \forall \langle a_1, a_2 \rangle \in A$, the $Q$-value for agent 1, which gives the expected payoff for a joint decision rule, can be calculated as:

$$Q_{NFG}(\langle \delta_1, \delta_2 \rangle) \triangleq \sum_{a_1} \delta_1(a_1) \sum_{a_2} \delta_2(a_2) R(\langle a_1, a_2 \rangle).$$

Note, $Q_{NFG}$ is equivalent to the utility function $u_1$. We now aim to find the rational joint decision rule $\delta^* = \langle \delta_1^*, \delta_2^* \rangle$, as this allows us to compute the value of the NFG:

$$V_{NFG} \triangleq \sum_{a_1} \delta_1^*(a_1) \sum_{a_2} \delta_2^*(a_2) R(a_1, a_2). \quad (2.3.1)$$
Chapter 2. Background Solution Methods

The Linear Programs we give below are based on [Shoham and Leyton-Brown, 2008, Chapter 4]. If \( \delta^*_2 \) is known, the decision rule \( \delta^*_1 \) can be found by solving the following LP:

\[
\begin{align*}
\max_{\delta_1} & \quad Q(\langle \delta_1, \delta^*_2 \rangle) \\
\text{subject to} & \quad \sum_{a_1} \delta_1(a_1) = 1 \\
& \quad \delta_1(\cdot) \geq 0
\end{align*}
\tag{2.3.2}
\]

However, \( \delta^*_2 \) is usually not known beforehand, as it depends on \( \delta^*_1 \). We solve this problem by noting that our game is zero-sum, and that a rational strategy for agent 1, \( \delta^*_1 \), is a maxmin-strategy. Let \( v_i \) be a free variable that represents the expected utility for agent \( i \) in equilibrium. As the game is zero-sum, we have \( v_1 = -v_2 \). Furthermore, by the minmax-theorem, \( v_1 \) is maximal in all scenarios where agent 1 follows a maxmin-strategy.

Obviously, agent 1 aims to maximize \( v_1 \). We add constraints specifying that \( v_1 \) must be equal to or lower than the expected payoff for every action \( a_2 \in A_2 \) (i.e., for all pure decision rules). Effectively, these constraints capture agent 2’s incentive to minimize \( R \): if an action \( a_2 \) exists that results in low reward for agent 1 \( R(a_1, a_2) \), agent 2 will always choose this action, effectively constraining \( v_1 \).

\[
\begin{align*}
\max_{\delta_1, v_1} & \quad v_1 \\
\text{subject to} & \quad \sum_{a_1} \delta_1(a_1) R(a_1, a_2) \geq v_1 \quad \forall a_2 \in A_2 \\
& \quad \sum_{a_1} \delta_1(a_1) = 1 \\
& \quad \delta_1(\cdot) \geq 0
\end{align*}
\tag{2.3.3}
\]

The LP used to find the rational decision rule for agent 2 can be constructed similarly\(^4\), except agent 2 aims to maximize \( v_1 \) (effectively minimizing \( v_2 \)).

\[
\begin{align*}
\min_{\delta_2, v_1} & \quad v_1 \\
\text{subject to} & \quad \sum_{a_2} \delta_2(a_2) R(a_1, a_2) \leq v_1 \quad \forall a_1 \in A_1 \\
& \quad \sum_{a_2} \delta_2(a_2) = 1 \\
& \quad \delta_2(\cdot) \geq 0
\end{align*}
\tag{2.3.4}
\]

\(^4\)In fact, the LP from the perspective of one agent turns out to be the dual of the LP for the opposing agent [Shoham and Leyton-Brown, 2008].
The solutions to these programs give the maxmin-value and the minmax-value. By the minmax theorem, these are equal. We find the value of the game to be $V_{\text{NFG}} = v_1 = -v_2$.

2.3.2 Extensive Form Games

As stated in 2.2.3, it is possible to convert an Extensive Form Game to Normal Form representation by enumerating all pure policies. An example is given in Figure 2.3.1, where the corresponding Normal Form payoff matrix $R$ is shown in table 2.3.1. The payoff shown in the table is the payoff for P1, which is the additive negation of the payoff for P2 by definition. The payoff matrix can be reduced in size by noting that there are pure policies that have the exact same effect on the outcome of the game, for example the policies $(l, L, l')$ and $(l, L, r')$ for P1. The matrix after removal of redundant rows and columns is appropriately named ‘reduced Normal Form’.

Entries in the Normal Form payoff matrix are the expected utility of a pure joint policy. This can found by taking the product of payoff at leaf nodes that can be reached via these pure policies, and the probability that these nodes are reached given the stochastic transitions specified by the chance nodes.

However, solving the resulting NFG is not always feasible, as the Normal Form representation of an EFG is exponential in the size of the game tree: every leaf node is reached by a combination of actions which form a pure policy, and thus adds a row or column to the payoff matrix. As the number of nodes grows, the NFG payoff matrix grows, and solving the game using the Linear Programs from section 2.3.1 quickly becomes intractable.

Even the reduced Normal Form payoff matrix is often too large. It turns out we can tackle this problem by using so-called sequence form representation, as introduced by Koller et al. [1994], for which solving the game is polynomial in the size of the game tree.

Figure 2.3.1: Game tree for an Extensive Form Game.
2.3.2.1 Sequence Form Representation

Sequence form representation, as introduced by Koller et al. [1994], is a representation for games that can be used to solve a zero-sum game in polynomial time in the size of the game tree. It is based on the notion that an agent can only contribute to part of the game tree, and that it is possible to compute how their contribution affects the probability of reaching certain leaf nodes regardless of the policy of the opponent or the influence of the environment. Instead of forming a mixed policy by assigning a probability distribution over the (many) pure policies, sequences of choices are assigned realization weights. These concepts are defined formally as follows.

**Definition** A sequence \( \sigma_i(p) \) for agent \( i \) is a tuple containing the information set and an action \( \langle h_i, a_i \rangle \) that lead to \( p \). More specifically, \( h_i \) is the last information set of agent \( i \) that is reached when we follow the edges on the path to \( p \), \( a_i \) is the action taken at \( h_i \) that corresponds to an edge that is in the path to \( p \).

A realization weight for a sequence reflects the probability that an agent makes the decisions contained in the sequence. The realization weight for such a sequence then gives us the probability that \( h_i \) will actually be reached, and that agent \( i \) chooses \( a_i \) at that point, given the policy of the agent.

**Definition** A realization weight of a sequence \( \sigma_i(p) \) for a given mixed policy \( \mu_i \), denoted as \( \mu_i(\sigma_i(p)) \), is the probability that agent \( i \) makes the decisions that are contained in the sequence leading to \( p \), assuming that the corresponding information sets are reached.

An exception is the sequence at the root node, which is depicted as \( \emptyset \), as there are no information sets and actions that lead to this node. The realization weight for this sequence is set to \( \mu_i(\emptyset) = 1 \).

Not every assignment of realization weights is valid: realization weights of continuations of a sequence must sum to the realization weight of that sequence. Let the sequence that leads to information set \( h_i \) be defined as \( \sigma_i(h_i) \). Let \( \langle h_i, a_{i,0} \rangle, \ldots, \langle h_i, a_{i,M} \rangle \) be the \( M \) continuations of the sequence \( \sigma_i(h_i) \). The realization weight for a sequence \( \sigma_i(h_i) \) can therefore be written as follows:

\[
\mu_i(\sigma_i(h_i)) = \sum_{m=0}^{M} \mu_i(\sigma_i(\langle h_i, a_{i,m} \rangle)).
\]

If the realization weights are known, we can use them to find the probability that an action \( a_{i,m} \) is chosen at information set \( h_i \). This allows us to convert a mixed policy to a stochastic policy, assuming that \( a_{i,m} \) is an action that can be chosen at \( h_i \):

\[
\pi_i(a_i|h_i) = \Pr(a_i|h_i, \mu_i) = \frac{\mu_i(\sigma_i(\langle h_i, a_{i,m} \rangle))}{\mu_i(\sigma_i(h_i))}. \tag{2.3.5}
\]
2.3.2.2 Solving Games in Sequence Form

Let $\beta(p)$ denote the product of probabilities that ‘nature’ chooses the edges that correspond to the path from the root node to node $p$. Koller et al. [1994] show that we can compute the probability of reaching a node $p$, given a pair of mixed policies $(\mu_1, \mu_2)$, and the product of probabilities that ‘nature’ chooses the edges corresponding to the path to $p$, as follows:

$$\Pr(p | \mu_1, \mu_2, \beta) = \mu_1(\sigma_1(p)) \mu_2(\sigma_2(p)) \beta(p).$$

(2.3.6)

Factorization of (2.3.6) shows that the contribution of either agent to the probability of reaching $p$ is independent of the contribution of the opponent and the environment, confirming the notion stated earlier.

We now aim to compute the realization weights that correspond to a rational policy, and retrieve the policy using (2.3.5). We will collect realization weights in a vector $\vec{\rho}_1$ that contains a realization weight for every sequence for agent 1 (similarly, $\vec{\rho}_2$ is defined for agent 2). Its entries satisfy the following constraints:

$$\vec{\rho}_i(\sigma) \geq 0 \quad \forall \sigma_i,$$

$$\vec{\rho}_i(\emptyset) = 1,$$

$$\vec{\rho}_i(\sigma_i(h_i)) = \sum_{m=0}^{M} \vec{\rho}_i(\sigma_i((h_i, a_i,m))) \quad \forall h_i \in H_i,$$

(2.3.7)

For the example given in Figure 2.3.1, there are seven possible sequences for agent 1: $\emptyset, l, r, rL, rR, l'$ and $r'$. Agent 2 has five possible sequences: $\emptyset, p, q, p'$ and $q'$. Finding $\vec{\rho}_i^*$ through Linear Programming requires us to set the aforementioned constraints. For example, the realization weight of sequence ‘$r'$’, $\vec{\rho}_1^*(r)$, must be equal to $\vec{\rho}_1^*(rL) + \vec{\rho}_1^*(rR)$.

The matrices and equalities in Figure 2.3.2 capture all three constraints in (2.3.7).

Let $K_{[\sigma_1, \sigma_2]}'$ be the set of all leaf nodes that can be reached if the agents follow the decisions specified in the sequences $\sigma_1$ and $\sigma_2$. Entries in the sequence form payoff matrix $P$ for agent 1 are defined as the sum of reward corresponding all reachable nodes $k$, multiplied by the probability that nature takes the choices leading to this node:

$$P(\sigma_1, \sigma_2) = \sum_{p \in K_{[\sigma_1, \sigma_2]}'} \beta(p)R(p).$$

The sequence form payoff matrix for agent 1, for the example tree given in Figure 2.3.1, is given in Figure 2.3.3. As the game is zero-sum, the payoff matrix for agent 2 is the additive inverse of the payoff matrix of agent 1. Note, that for the current example the

5Koller et al. refer to the vector $\vec{\rho}_i$ as the realization plan for agent $i$. 
sequence form payoff matrix is larger than the Normal Form payoff matrix in Table 2.3.1, but that it is a lot more sparse.

\[
E = \begin{bmatrix}
\emptyset & l & r & r_L & r_R & l' & r'
\end{bmatrix} = \begin{bmatrix}
1 \\
-1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 1 & 0 \\
-1 & 0 & 0 & 0 & 0 & 1 & 1
\end{bmatrix}
\]

\[
e = \begin{bmatrix}
\emptyset & p & q & p' & q'
\end{bmatrix} = \begin{bmatrix}
0 \\
l \\
r \\
r_L \\
r_R
\end{bmatrix}
\]

\[
F = \begin{bmatrix}
\emptyset & p & q & p' & q'
\end{bmatrix} = \begin{bmatrix}
1 \\
-1 & 1 & 1 & 0 & 0 \\
-1 & 0 & 0 & 1 & 1
\end{bmatrix}
\]

\[
f = \begin{bmatrix}
\emptyset & p & q & p' & q'
\end{bmatrix} = \begin{bmatrix}
1 \\
0 \\
0 \\
-3 & 0 & 0 & 0 \\
0 & -2 & -3 & 0
\end{bmatrix}
\]

\[
E\hat{\rho}_1 = e \quad (2.3.8)
\]

\[
F\hat{\rho}_2 = f \quad (2.3.9)
\]

We can now rewrite the Linear Programs used to solve zero-sum Normal Form Games to a more general sequence form LP. Let \(v_1\) and \(v_2\) be two vectors of unbound variables, with the same dimensions as respectively \(e\) and \(f\). An intuitive interpretation of the elements of these vectors is that they contain the contribution of the opponent to the expected payoff of agent \(i\), for every information set of agent \(i\). The very first element, corresponding to the ‘root’ information set, is then the expected payoff of the system. In the following LP, the objective function captures agent 1’s incentive to maximize expected payoff and the constraints specified in (2.3.7).

\[
\max_{\hat{\rho}_1, v_2, f} \quad -v_2^T f
\]

subject to

\[
\hat{\rho}_1^T (-P) - v_2^T F \quad \leq 0
\]

\[
E\hat{\rho}_1 = e
\]

\[
\hat{\rho}_1 \quad \geq 0
\]

Figure 2.3.2: Constraint matrices and constraints for the sequence form Linear Program.

Figure 2.3.3: Sequence form payoff matrix for P1 for the EFG in Figure 2.3.1.
The LP from the perspective of the opposing agent is the following\(^6\).

\[
\begin{align*}
\min_{\tilde{\rho}_2, v_1, e} & \quad e^\top v_1 \\
\text{subject to} & \quad -P\tilde{\rho}_2 + E^\top v_1 \geq 0 \\
& \quad F\tilde{\rho}_2 = f \\
& \quad \tilde{\rho}_2 \geq 0
\end{align*}
\]  

(2.3.11)

Realization weight vectors corresponding to rational policies, \(\tilde{\rho}_1^*\) and \(\tilde{\rho}_2^*\), can be found by solving LPs 2.3.10 and 2.3.11, respectively. The solutions to these LPs give the maxin- and minmax-value. By the minmax-theorem, these values are equal, and they give us the value of the Extensive Form Game: \(V_{\text{EFG}} = e^\top v_1 = \tilde{\rho}_1^T P\tilde{\rho}_2^* = -f^\top v_2\). From the realization weight vectors we can obtain the probability of taking an action given an information set, which allows us to construct a rational stochastic policy for both agents using (2.3.5):

\[\pi_i^*(a_i|h_i) = \frac{\tilde{\rho}_i^*(\sigma_i(h_i, a_i))}{\tilde{\rho}_i^*(\sigma_i(h_i))}, \forall h_i \in H_i, a_i \in A_i.\]

### 2.3.3 Bayesian Games

A stochastic decision rule in a Bayesian Game is a mapping from types to probability distributions over actions, denoted as \(\delta(a|\theta)\). Given a joint decision rule, which is a tuple containing decision rules for both agents \(\delta = \langle \delta_1, \delta_2 \rangle\), the Q-value of the zero-sum BG is:

\[Q_{\text{BG}}(\delta) \triangleq \sum_\theta \sigma(\theta) \sum_a \delta(a|\theta)R(\theta, a).\]  

(2.3.12)

If we know the rational joint decision rule \(\delta^*\), then we can define the value of the BG as:

\[V_{\text{BG}} \triangleq Q_{\text{BG}}(\delta^*) = \sum_\theta \sigma(\theta) \sum_a \delta^*(a|\theta)R(\theta, a).\]  

(2.3.13)

This definition of the value holds for nonzero-sum Bayesian Games as well, under the condition that a Nash Equilibrium (and thus a rational joint decision rule \(\delta^*\)) exists. As we are playing a zero-sum game, we know that agent 1 is trying to maximize the value that agent 2 is minimizing. This allows us to redefine (2.3.13) to a more specific form that makes the incentives of the two agents explicit in its definition:

\[V_{\text{BG}} \triangleq \max_{\delta_1} \min_{\delta_2} Q_{\text{BG}}(\langle \delta_1, \delta_2 \rangle).\]  

(2.3.14)

By the minmax-theorem, (2.3.14) is equal to its counterpart \(V_{\text{BG}}(\langle \delta_1, \delta_2 \rangle) = \min_{\delta_2} \max_{\delta_1} Q_{\text{BG}}(\langle \delta_1, \delta_2 \rangle)\). A rational joint decision rule is then a tuple containing a maximizing decision rule \(\delta_1\) and a minimizing decision rule \(\delta_2\).

\(^6\)Koller et al. [1994] show that LP 2.3.11 is the dual to LP 2.3.10.
It is possible to find a rational joint decision rule in a zero-sum Bayesian Game using techniques for Normal Form Games. To do so, we transform the BG payoff matrix to Normal Form by treating all individual types $\theta_i$ as independent agents. The resulting NFG can be solved using standard solution methods (for example, using the technique from section 2.3.1), which results in a stochastic decision rule specifying a probability distribution over actions (or alternatively, a mixed decision rule) for each individual type. Combining the (Normal Form) decision rules corresponding to the types of agent $i$ then gives us the rational (Bayesian) decision rule $\delta^*_i$.

For example, converting a two-player zero-sum BG where both agents have two types and two actions to Normal Form results in a NFG of 4 ‘agents’, each of which chooses between two actions. To find the Bayesian decision rule for agent 1, we solve the NFG of four agents, and combine the decisions made by the ‘agents’ corresponding to $\theta_1$ and $\bar{\theta}_1$ to form a single decision rule $\delta_1$.

For large number of types, however, the resulting Normal Form payoff matrix is usually high-dimensional, and solving the NFG quickly becomes infeasible. A zero-sum BG can be converted to Extensive Form instead, which allows us to solve the game using sequence form representation. An example Bayesian Game in Extensive Form is given in Figure 2.3.3, with the corresponding BG payoff matrix shown in 2.3.2 (it is a zero-sum game, and only payoff for P1 is shown). It turns out that every type $\theta_i$ in the BG corresponds to exactly one information set for agent $i$ in the EFG. Therefore, the sequence form payoff matrix is exactly the BG payoff matrix but with one row and column (both filled with zeros) added for the ‘root’ sequences, as no outcome nodes can be reached from the root node before both agents have chosen an action (as seen in Figure 2.3.3). Once the BG is converted to sequence form, the rational joint decision rule and the value can be found using the Linear Programs from section 2.3.2.

![Figure 2.3.4: Extensive Form representation of an example Bayesian Game.](image)

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$a_2$</td>
</tr>
<tr>
<td>$\pi_1$</td>
<td>$\pi_2$</td>
</tr>
<tr>
<td>$a_1$</td>
<td>$-4$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$-2$</td>
</tr>
<tr>
<td>$\bar{\pi}_1$</td>
<td>$2$</td>
</tr>
</tbody>
</table>

Table 2.3.2: Example BG payoff matrix.
2.3.4 Partially Observable Stochastic Games

Like the BG and EFG, a Partially Observable Stochastic Game can be converted to a Normal Form Game by enumerating the pure policies. However, solving the resulting NFG quickly becomes intractable, as the number of policies grows exponentially in the number of AOHs and actions. Like the zero-sum Bayesian Game, we can instead opt to convert the POSG to an EFG and subsequently use sequence form representation to solve the game [Oliehoek and Vlassis, 2006]. We will show how to convert the two-player zero-sum POSG to sequence form directly.

In accordance with section 2.3.2.2, we construct a total of five matrices: two constraint matrices for each agent and one sequence form payoff matrix. The size of the constraint matrices depends on the number of information sets and sequences per agent. For example, if five information sets and ten possible sequences exist for agent 1, then the corresponding constraint matrices $E$ and $e$ will be of size $5 \times 10$ and $5 \times 1$ respectively. We will express these in terms of components of the POSG.

The information sets for agent $i$, $H_i$, are by definition the sets of joint AOHs that it cannot distinguish between. As explained in section 2.1.4, this means that every individual AOH for agent $i$ induces one information set $h_i$ in the EFG.

The sequences for agent 1 are all possible combinations of individual AOHs and individual actions. Let the sequence form payoff matrix contain a row for every sequence of agent 1 and a column for every sequence of agent 2. The entries of this matrix correspond to the immediate reward attained at every stage when following the actions captured in the two sequences, multiplied by relevant observation probabilities (the actions by ‘nature’). That is, for sequences $\langle \vec{\theta}_1^t, a_1^t \rangle$ and $\langle \vec{\theta}_2^t, a_2^t \rangle$, the corresponding entry in the sequence form payoff matrix is:

$$P(\langle \vec{\theta}_1^t, a_1^t \rangle, \langle \vec{\theta}_2^t, a_2^t \rangle) = R(\langle \vec{\theta}_1^t, \vec{\theta}_2^t \rangle, \langle a_1^t, a_2^t \rangle) \cdot \beta(\langle \vec{\theta}_1^t, \vec{\theta}_2^t \rangle),$$

where $\beta$ returns the product of observational probabilities for a joint AOH as $\beta(\vec{\theta}) = \prod_{t=0}^{t} \Pr(o^{t+1} | \vec{\theta}, a^t)$.

While solving the zero-sum POSG this way is generally efficient (polynomial in the size of the game tree [Koller et al., 1994]), even sequence form Linear Programss will be unsolvable for games of large horizon, as we will show empirically in chapter 6. In the following chapters, we provide new theory that, we hope, may open up the route for solution methods that allow for rational decision-making in zero-sum POSGs of large horizon as well.
Chapter 3

Families of Zero-Sum Bayesian Games

The Bayesian Game framework describe the problem of decision-making under uncertainty in a one-shot game with hidden state, selected according to a probability distribution that is common knowledge among the agents. This probability distribution, \( \sigma \), is a static element of the BG, and cannot be affected by the agents.

As we will show in chapter 4, a similar probability distribution called a plan-time sufficient statistic exists in the POSG setting, with the important difference that the agents can influence this statistic through their decisions. Obviously, a rational and strategic agent in the zero-sum POSG aims to reach a statistic that will eventually result in the highest individual payoff. Therefore, if we are to find a solution to a zero-sum POSG using these statistics, we must try to answer the following question: What is the statistic at the next timestep that maximizes individual payoff?

Before answering that question for the zero-sum POSG, we show that in the one-shot case, a distribution over information that allows the agent to maximize individual payoff exists. In section 3.1, we introduce an framework for one-shot games of incomplete information in which the probability distribution over joint types is not set, which we name the Family of Bayesian Games. Essentially, it is a collection of Bayesian Games for which all elements but the type distribution are equal.

We define its value function in terms of the type distribution. In section 3.2, we prove formally that this value function exhibits a concave and convex structure. In chapter 4, we will show that the final stage of the zero-sum POSG is equivalent to family of zero-sum Bayesian Games, indicating that at the final stage, the value function of the zero-sum POSG exhibits the same concave and convex properties. Subsequently, we extend this structural result to all stages of the zero-sum POSG.
Chapter 3. Families of Zero-Sum Bayesian Games

3.1 Framework

Let $\Delta(\Theta)$ be the simplex over the set of joint types $\Theta$. Let $\sigma \in \Delta(\Theta)$ be a probability distribution over joint types, equivalent to the type distribution in the Bayesian Game from section 2.1.2. We define the Family of Bayesian Games framework as follows.

**Game 3.1.1.** A *Family of Bayesian Games*, defined as a tuple $\langle I, \Theta, A, R \rangle$, is the set of Bayesian Games for which all elements but the type distribution $\sigma$ are identical, for all $\sigma \in \Delta(\Theta)$:

- $I = \{1, \ldots, n\}$ is the set of agents,
- $\Theta = \Theta_1 \times \ldots \times \Theta_n$ is the set of joint types $\theta = \langle \theta_1, \ldots, \theta_n \rangle$,
- $A = A_1 \times \ldots \times A_n$ is the set of joint actions $a = \langle a_1, \ldots, a_n \rangle$,
- $R : \Theta \times A \to \mathbb{R}^n$ is a reward function mapping joint types and joint actions to payoff for each agent.

Let $\mathcal{F}$ be a Family of zero-sum Bayesian Games. Its value function $V^*_\mathcal{F}$ can be defined in terms of the type distribution:

$$V^*_\mathcal{F}(\sigma) \triangleq \sum_{\theta} \sigma(\theta) \sum_a \delta^*(a|\theta) R(\theta, a). \quad (3.1.1)$$

Here, $\delta^* = (\delta^*_1, \delta^*_2)$ is the rational joint decision rule (as defined in section 2.2). Note that evaluation of (3.1.1) for a particular $\sigma$ gives the value for the BG in $\mathcal{F}$ that has type distribution $\sigma$. We generalize the definitions for the Q-value and value in the zero-sum BG setting, (2.3.14) and (2.3.14), as follows:

$$Q_{\mathcal{F}}(\sigma, \delta) \triangleq \sum_{\theta} \sigma(\theta) \sum_a \delta(a|\theta) R(\theta, a), \quad (3.1.2)$$

$$V^*_\mathcal{F}(\sigma) \triangleq \max_{\delta_1} \min_{\delta_2} Q_{\mathcal{F}}(\sigma, \langle \delta_1, \delta_2 \rangle). \quad (3.1.3)$$

This redefinition of the value function will prove useful in the next section, where we show that the value function exhibits a particular structure.

3.2 Structure in the Value Function

We will give formal proof that the value function of a Family of zero-sum Bayesian Games exhibits a particular concave and convex structure in different subspaces of $\sigma$-space. Let us define best-response value functions that give the best-response value to a decision rule of the opposing agent:

$$V^{BR1}_{\mathcal{F}}(\sigma, \delta_2) \triangleq \max_{\delta_1} Q_{\mathcal{F}}(\sigma, \langle \delta_1, \delta_2 \rangle), \quad (3.2.1)$$

$$V^{BR2}_{\mathcal{F}}(\sigma, \delta_1) \triangleq \min_{\delta_2} Q_{\mathcal{F}}(\sigma, \langle \delta_1, \delta_2 \rangle). \quad (3.2.2)$$
By the minmax-theorem [Von Neumann and Morgenstern, 2007], (3.1.3), (3.2.1) and (3.2.2), the following holds:

\[ V^*_x(\sigma) = \min_{\delta_2} V^{BR1}_x(\sigma, \delta_2) = \max_{\delta_1} V^{BR2}_x(\sigma, \delta_1). \]  

(3.2.3)

Let us decompose \( \sigma \) into a marginal term \( \sigma_{m1} \) and a conditional term \( \sigma_{c1} \):

\[ \sigma_{m1}(\theta_1) \triangleq \sum_{\theta_2} \sigma((\theta_1, \theta_2)), \]

(3.2.4)

\[ \sigma_{c1}(\theta_2|\theta_1) \triangleq \frac{\sigma((\theta_1, \theta_2))}{\sum_{\theta'_2} \sigma((\theta_1, \theta'_2))} = \frac{\sigma_{m1}(\theta_1)}{\sigma_{m1}(\theta_1)}. \]

(3.2.5)

The terms \( \sigma_{m2} \) and \( \sigma_{c2} \) are defined similarly. By definition, we have:

\[ \sigma((\theta_1, \theta_2)) = \sigma_{m1}(\theta_1)\sigma_{c1}(\theta_2|\theta_1) = \sigma_{m2}(\theta_2)\sigma_{c2}(\theta_1|\theta_2). \]

(3.2.6)

For the sake of concise notation, we will write \( \sigma = \sigma_{i,m}\sigma_{i,c} \). Let \( \vec{\sigma}_{1,m} \) be the vector notation of a marginal. Each entry in this vector corresponds to the probability for one type (as specified by the marginal) \( \sigma_{i,m}(\theta_i) \).

Let us refer to the simplex \( \Delta(\Theta_i) \) containing marginals \( \sigma_{m,i} \) as the marginal-space of agent \( i \). We define a value vector that contains the reward for agent 1 for each individual type \( \theta_1 \), given \( \sigma_{c,1} \) and given that agent 2 follows decision rule \( \delta_2 \):

\[ r_{[\sigma_{c,1},\delta_2]}(\theta_1) \triangleq \max_{a_1} \sum_{\theta_2} \sigma_{c1}(\theta_2|\theta_1) \sum_{a_2} \delta_2(a_2|\theta_2) R((\theta_1, \theta_2), (a_1, a_2)). \]

(3.2.7)

The vector \( r_{[\sigma_{c,2},\delta_1]} \) is defined similarly.

Now that we have established notation, we will show that the best-response value functions defined in (3.2.1) and (3.2.2) are linear in their respective marginal-spaces. In fact, they correspond exactly to the previously defined value vector.

**Lemma 3.2.1.** (1) \( V^{BR1}_x \) is linear in \( \Delta(\Theta_1) \) for all \( \sigma_{c,1} \) and \( \delta_2 \), and (2) \( V^{BR2}_x \) is linear in \( \Delta(\Theta_2) \) for all \( \sigma_{c,2} \) and \( \delta_1 \). More specifically, we can write the best-response value functions as the inner product of a marginal \( \sigma_{m,i} \) and the vector \( r_{[\sigma_{c,i},\delta_i]} \):

1. \( V^{BR1}_x(\sigma_{m1}\sigma_{c1}, \delta_2) = \vec{\sigma}_{m1} \cdot r_{[\sigma_{c,1},\delta_2]} \),

(3.2.8)

2. \( V^{BR2}_x(\sigma_{m2}\sigma_{c2}, \delta_1) = \vec{\sigma}_{m2} \cdot r_{[\sigma_{c,2},\delta_1]} \).

(3.2.9)

**Proof** The proof is listed in Appendix A.

Using this result, we prove that \( V^*_x \) exhibits concavity in \( \Delta(\Theta_1) \) for every \( \sigma_{c,1} \), and convexity in \( \Delta(\Theta_2) \) for every \( \sigma_{c,2} \).
Theorem 3.2.2. $V^*_F$ is (1) concave in $\Delta(\Theta_1)$ for a given conditional distribution $\sigma_{c,1}$, and (2) convex in $\Delta(\Theta_2)$ for a given conditional distribution $\sigma_{c,2}$. More specifically, $V^*_F$ is respectively a minimization over linear functions in $\Delta(\Theta_1)$ and a maximization over linear functions in $\Delta(\Theta_2)$:

1. $V^*_F(\sigma_{m,1}\sigma_{c,1}) = \min_{\delta_2} [\bar{\sigma}_{m,1} \cdot r_{[\sigma_{c,1},\delta_2]}]$
2. $V^*_F(\sigma_{m,2}\sigma_{c,2}) = \max_{\delta_1} [\bar{\sigma}_{m,2} \cdot r_{[\sigma_{c,2},\delta_1]}]$.

Proof We prove item 1 using the result of Lemma 3.2.1:

$$V^*_F(\sigma_{m,1}\sigma_{c,1}) \overset{(3.2.2)}{=} \min_{\delta_2} V^*_F^{BR_1}(\sigma_{m,1}\sigma_{c,1},\delta_2) \overset{(3.2.8)}{=} \min_{\delta_2} [\sigma_{m,1} \cdot r_{[\sigma_{c,1},\delta_2]}].$$

The proof for item 2 is analogous to that of item 1.

To illustrate these results, we have visualized the (concave and convex) value function for an example Family of zero-sum Bayesian Games in Figure 3.2.1, by computing the value at various points in $\sigma$-space and connecting these using a wireframe. The payoff matrix is given in Table 3.2.1. In this example, each agent has two types and two actions. As the game is zero-sum, only the payoff for agent 1 is shown — the payoff for agent 2 is its additive inverse. We observe that the value function is a saddle point function in $\Delta(\Theta)$, in line with our theoretical results. Of course, if we had used an example BG with more than two types per agent, $\Delta(\Theta)$ would be high-dimensional, and visualizing the value function is not straightforward.
3.3 Discussion

An important implication of the result of Theorem 3.2.2 is that it indicates that agents have an incentive to prefer a certain probability distribution over types. The result is intuitive: if the information distribution $\sigma$ is located at the ‘corners’ of $\Delta(\Theta_1)$, agent 2 will be very certain of the type of agent 1, as all probability mass is on one type $\theta_1$. This makes it easy for agent 2 to exploit agent 1. As agent 2 is the minimizing agent, this results in low value. Inversely, if $\sigma$ is located at the center of $\Delta(\Theta_1)$, agent 2 will be uncertain about the type of agent 1. This makes it more difficult to exploit agent 1, and will therefore result in higher value.

While the found structure is not directly useful in the Family of Bayesian Games as the agents are not able to influence the information distribution, we will show, in chapter 4, that the proof can be extended to the zero-sum POSG setting. In that setting, the agents can influence the probability distribution over information — a multi-agent belief called the \textit{plan-time sufficient statistic} — at timestep $t$ through their decisions at earlier timesteps. Identifying that some statistics are ‘better’ than others (i.e., they result in higher individual payoff than other statistics) is only the first step towards rational decision-making in the zero-sum POSG, however. In order to exploit the founds structure, we will need to determine where the ‘best’ statistics are located, and also find the joint policy that allows these statistic to be reached.
Chapter 4

Zero-Sum POSGs

In this chapter, we will extend the structural results of chapter 3 (i.e., the value function of the Family of Bayesian Games is concave and convex) to the zero-sum POSG setting. We take inspiration from recent work for Dec-POMDPs, which introduces so-called plan-time sufficient statistics, essentially a joint belief over concatenations of private information [Oliehoek, 2013]. Use of this statistic allows for a reduction of the Dec-POMDP to a special type of POMDP [MacDermed and Isbell, 2013], leading to increases in scalability [Dibangoye et al., 2009].

We adapt the statistic for use in the zero-sum POSG setting, and show that a similar reduction to a special type of stochastic game is possible. Furthermore, we show that the final stage of the zero-sum POSG, when defined in terms of the new statistic, is equivalent to a Family of zero-sum Bayesian Games, which means that at the final stage, the value function of the zero-sum POSG exhibits the concave and convex properties found in chapter 3. We give formal proof that the value function of the zero-sum POSG exhibits these properties at all stages of the game. In later chapters, we attempt to exploit this structure in order to find the Nash Equilibrium and value of the zero-sum POSG.

4.1 Sufficient Statistics for Decision-Making

To explain the usefulness of a plan-time sufficient statistic, we will first explain what makes something a sufficient statistic for decision-making. Consider the fully observable single-agent setting, typically modeled as a Markov Decision Process. If the agent wants to behave rationally in this setting, it is sufficient to know just the state of the game, as it satisfies the Markov property: the current state provides sufficient information to determine, given an action, the conditional probability distribution over future states. As such, the agent is able to reason about the consequences of his actions, and can compute how his current decision affects his future value. In this setting, the state is a sufficient statistic for decision-making.
In partially observable games, the state is typically hidden, so it can no longer act as a sufficient statistic. Instead, the agent holds a belief over states: a probability distribution that specifies, for each state, the agent’s belief that he is in that state. During the game, it is altered based on the received observations and chosen actions, according to a deterministic belief-update [Kaelbling et al., 1998]. As such, the belief contains all the information the agent needs to make decisions at the current stage, and allows the agent to reliably determine the next belief. Therefore, it is a sufficient statistic.

Decentralized multi-agent games of incomplete information provide additional complexity, as each agent recalls only individual actions and observations, and generally does not know the Action-Observation History (AOH) of the opponent (counterexamples include games where limited communication is possible [Oliehoek et al., 2007; Spaan et al., 2008]). It is possible to compute a probability distribution over AOHs, as we will show. However, to find this belief, an agent will have to plan for all agents, i.e., compute the rational joint policy. Assuming that we know the initial belief $b^0$ and the rational joint policy $\pi^*$, the conditional probability on an AOH is defined as

$$\Pr(\vec{\theta}|b^0, \pi^*) = \Pr(\langle a^0, o^1, \ldots, a^{t-1}, o^t \rangle|b^0, \pi^*) = \prod_{k=1}^{t} \Pr(a^{t-k}|\pi^*, \vec{\theta}^{t-k}) \Pr(o^{t-k+1}|a^{t-k}, \vec{\theta}^{t-k}).$$

Recent work on collaborative multi-agent games proposes to use this probability distribution over AOHs as a statistic for decision-making in games with stochastic policies [Oliehoek, 2013]. In the next sections, we will adapt this same statistic for use in the zero-sum POSG.

### 4.2 Value Function of the Zero-Sum POSG

In this section, we give two formulations for the value function of the zero-sum POSG, one in terms of the past joint policy (in section 4.2.1) and one in terms of a plan-time sufficient statistic (in section 4.2.2). These formulations are relatively straightforward extensions from previous work on Dec-POMDPs [MacDermed and Isbell, 2013; Oliehoek, 2013]. In section 4.2.3 we show that, using plan-time sufficient statistics, the zero-sum POSG model can be reduced to a centralized stochastic game with hidden state and without observations, which we refer to as the Non-Observable Stochastic Game (NOSG). Furthermore, they allow us, in section 4.2.5, to prove that the value function of the zero-sum POSG exhibits a particular structure.

---

[1] Oliehoek [2013] shows that a probability distribution over observation-histories and states is also sufficient in the Dec-POMDP setting. This statistic is typically more compact than the distribution over AOHs. However, it is not sufficient in the zero-sum POSG setting.

---

31
4.2.1 Value in Terms of Past Joint Policies

In order to facilitate the formulation in the next subsection, we first express the value of the POSG at a stage $t$ in terms of a past joint policy $\varphi^t$ (as defined in section 2.1.4), attained when all agents follow the joint decision rule $\delta^t$, and assuming that in future stages agents will act rationally, i.e., they follow a rational joint future policy $\pi^{t+1} = (\delta^{t+1} \ldots \delta^{h-1})$.

Conceptually, this enables us to treat the problem of finding a rational joint policy as a series of smaller problems, namely identification of a rational joint decision rule $\delta^t$ at every stage. However, as we will show, a circular dependency exists: selection of $\delta^t$ is dependent on the future rational decision rule $\delta^{t+1}$, which in turn is dependent on $\varphi^t$ and thus on $\delta^t$.

We define the Q-value function at the final stage $t = h - 1$, and give an inductive definition of the Q-value function at preceding stages. We then define the value function at every stage. Let the reward function in terms of a joint AOH and joint action be defined as

$$R(\vec{\theta}, a_t) \triangleq \sum s_t \Pr(s_t | \vec{\theta}, b_0) R(s_t, a_t).$$

(4.2.1)

For the final stage $t = h - 1$, the Q-value function reduces to this immediate reward, as there is no future value:

$$Q^*_{h-1}(\varphi^{h-1}, \vec{\theta}^{h-1}, \delta^{h-1}) \triangleq R(\vec{\theta}^{h-1}, \delta^{h-1}).$$

(4.2.2)

Note, that this formulation includes the past joint policy $\varphi^{h-1}$, but that it is not dependent on $\varphi^{h-1}$ directly. Rather, it depends on the joint AOH $\vec{\theta}^{h-1}$. Furthermore, the state is not explicitly contained in this definition, as knowing the AOH is sufficient to determine the state [Oliehoek, 2012].

Given an AOH and decision rule at stage $t$, it is possible to find a probability distribution over AOHs at the next stage, as an AOH at $t + 1$ is the AOH at $t$ concatenated with action $a^t$ and observation $o^{t+1}$:

$$\Pr(\vec{\theta}^{t+1} | \vec{\theta}, a^t) = \Pr((\vec{\theta}, a^t, o^{t+1}) | \vec{\theta}, a^t) = \Pr(o^{t+1} | \vec{\theta}, a^t) \delta^t(a^t | \vec{\theta}).$$

For all stages except the final stage $t = h - 1$, the value at future stages is propagated to the current stage using (4.2.3):

$$Q_t^*(\varphi^t, \vec{\theta}^t, \delta^t) \triangleq R(\vec{\theta}^t, \delta^t) + \sum_{a^t} \sum_{o^{t+1}} \Pr(\vec{\theta}^{t+1} | \vec{\theta}^t, a^t) Q_{t+1}^*(\varphi^{t+1}, \vec{\theta}^{t+1}, \delta^{t+1}),$$

(4.2.3)

$$Q_t^*(\varphi^t, \delta^t) \triangleq \sum_{\vec{\theta}} \Pr(\vec{\theta}^0 | \varphi^t) Q_t^*(\varphi^t, \vec{\theta}, \delta^t).$$

(4.2.4)
We use (4.2.4) to find rational decision rules for both agents. Consistent with (4.2.3), we show how to find $\delta_t^{t+1}$ (of course, $\delta_t^t$ can be found similarly):

$$\delta_t^{t+1} = \arg\max_{\delta_t^{t+1}} \left[ Q_{t+1}^*(\varphi_t^{t+1}, \langle \delta_t^t, \delta_t^{t+1} \rangle) \right],$$

(4.2.5)

$$\delta_t^{t+2} = \arg\min_{\delta_t^{t+1}} \left[ Q_{t+1}^*(\varphi_t^{t+1}, \langle \delta_t^t, \delta_t^{t+1} \rangle) \right].$$

(4.2.6)

For a given $\varphi^{h-1}$, a rational joint decision rule $\delta^{h-1}$ can be found by performing a maximinimization over immediate reward (using (4.2.2), (4.2.5) and (4.2.6)). Evaluation of $Q_{h-1}^*(\varphi^{h-1}, \delta^{h-1})$ gives us the value at stage $t = h - 1$, and (4.2.3) propagates the value to the preceding stages. As such, rationality for all stages follows by induction. We can now define the value function in terms of the past joint policy as:

$$V_t^*(\varphi^t) = \max_{\delta_t^t} \min_{\delta_t^{t+1}} Q_t^*(\varphi^t, \langle \delta_t^t, \delta_t^{t+1} \rangle).$$

(4.2.7)

By (4.2.2), (4.2.5) and (4.2.6), $\delta^t$ is dependent on $\delta^{t+1}$, and thus on the rational future joint policy. However, $\delta^{t+1}$ can only be found if past joint policy $\varphi^{t+1}$, which includes $\delta^t$, is known. This circular dependency on both the future and past joint policy makes multi-agent decision-making in POSGs a difficult problem. Furthermore, in the Dec-POMDP it is possible to find an exact solution using dynamic programming because we then search in the finite space of pure joint policies [Seuken and Zilberstein, 2007]. In the zero-sum POSG case, we search in the infinitely large space of stochastic policies, rendering the use of such approaches impossible.

### 4.2.2 Value in Terms of Plan-Time Sufficient Statistics

Even though there are infinitely many past joint policies in the zero-sum POSG, we do not expect that their effects on the game at a particular stage are completely arbitrary. In fact, in this section we propose to replace the dependence of the value function on past joint policies by a plan-time sufficient statistic, $\sigma^t$, that summarizes many past joint policies. This new statistic potentially breaks the circular dependency discussed in the previous subsection: with decision rule selection at stage $t$ dependent on the new plan-time statistic rather than the past joint policy, agents may be able to determine the rational partial policy $\pi^{t*}$ if they know the statistic, regardless of choices made on stages 0 to $t$. However, as we will show, there are infinitely many statistics, meaning that we cannot use brute-force approaches (i.e., determine a rational partial policy $\pi^{t*}$ for all statistics $\sigma^t$). Therefore, as decision-making on stage $t$ depends on the statistic $\sigma^t$, we still need to determine which statistic $\sigma^t$ will be reached by a rational past joint policy $\varphi^{t*}$.

As we will show in section 4.2.5, the value function exhibits a certain concave and convex shape in statistic-space, a property that we will try to exploit in later chapters.
The plan-time sufficient statistic is formally defined as follows.

**Definition** The plan-time sufficient statistic for a general past joint policy \( \varphi_t \), assuming \( b^0 \) is known, is a distribution over joint AOHs: \( \sigma^t(\tilde{b}^t) \triangleq \Pr(\tilde{b}^t | b^0, \varphi_t) \) [Oliehoek, 2013].

In the collaborative Dec-POMDP case, these plan-time sufficient statistics fully capture the influence of the past joint policy. We will prove that this is also true in the zero-sum POSG case, by showing that use of these statistics allows for redefinition of the equations from Section 4.2.1. We aim to express the value for a given decision rule \( \delta_t \) in terms of a plan-time sufficient statistic, given that the agents act rationally at later stages. We first define the update rule for plan-time sufficient statistics:

\[
\sigma^{t+1}(\tilde{b}^{t+1}) \triangleq \Pr(o^{t+1} | \tilde{b}^t, a^t) \delta^t(a^t | \tilde{b}^t) \sigma^t(\tilde{b}^t). \quad (4.2.8)
\]

At the final stage \( t = h - 1 \), the Q-value function reduces to the immediate reward, as there is no future value:

\[
Q^*_h-1(\sigma^{h-1}, \tilde{b}^{h-1}, \delta^{h-1}) \triangleq R(\tilde{b}^{h-1}, \delta^{h-1}). \quad (4.2.9)
\]

We then define the Q-value for all other stages as:

\[
Q^*_t(\sigma^t, \tilde{b}^t, \delta^t) \triangleq R(\tilde{b}^t, \delta^t) + \sum_{\tilde{a}^t} \sum_{\tilde{o}^t} \Pr(\tilde{b}^{t+1} | \tilde{b}^t, \delta^t) Q^*_t+1(\sigma^{t+1}, \tilde{b}^{t+1}, \delta^{t+1})). \quad (4.2.10)
\]

\[
Q^*_t(\sigma^t, \delta^t) \triangleq \sum_{\tilde{b}^t} \sigma^t(\tilde{b}^t) Q^*_t(\sigma^t, \tilde{b}^t, \delta^t). \quad (4.2.11)
\]

We use (4.2.11) to find rational decision rules for both agents:

\[
\delta^{t+1}_1 = \operatorname{argmax}_{\delta^{t+1}_1} Q^*_t+1(\sigma^{t+1}, (\delta^{t+1}_1, \delta^{t+1}_2)), \quad (4.2.12)
\]

\[
\delta^{t+1}_2 = \operatorname{argmin}_{\delta^{t+1}_2} Q^*_t+1(\sigma^{t+1}, (\delta^{t+1}_1, \delta^{t+1}_2)). \quad (4.2.13)
\]

Let us formally prove the equivalence of (4.2.4) and (4.2.11).²

**Lemma 4.2.1.** \( \sigma^t \) is a sufficient statistic for decision-making in the zero-sum POSG, i.e. \( Q^*_t(\sigma^t, \tilde{b}^t, \delta^t) = Q^*_t(\varphi^t, \tilde{b}^t, \delta^t), \forall t \in 0 \ldots h - 1, \forall \tilde{b}^t \in \Delta(\mathcal{G}^t), \forall \delta^t. \)

**Proof** The proof is listed in Appendix A. \( \square \)

We define the value function for a two-player zero-sum POSG in terms of \( \sigma^t \):

\[
V^*_t(\sigma^t) \triangleq \max_{\delta^t_1, \delta^t_2} Q^*_t(\sigma^t, (\delta^t_1, \delta^t_2)). \quad (4.2.14)
\]

²This proof is a straightforward extension of proof of sufficiency in the Dec-POMDP setting.
Note the similarities between (4.2.14) and (4.2.7).

Although we have now identified the value at a single stage of the game, we cannot implement a backward inductive approach directly, as decisions on stages before stage $t$ affect the reached statistic $\sigma^t$. However, given statistic $\sigma^t$, it is possible to compute $\pi^{t*}$ without knowing $\varphi^t$, for example by converting the game induced by this statistic to a Normal Form Game (i.e., enumerating all pure partial policies $\hat{\pi}_1^t$ and $\hat{\pi}_2^t$ in order to construct a Normal Form payoff matrix) and solving it using Linear Programming.

### 4.2.3 Reduction to Non-Observable Stochastic Game

As discussed earlier in this chapter, a recent development in the field of Dec-POMDPs is that this (decentralized) model can be reduced to a special case of POMDP (a non-observable MDP, or NOMDP) [MacDermed and Isbell, 2013; Nayyar et al., 2013; Dibangoye et al., 2013; Oliehoek and Amato, 2014], which allows POMDP solution methods to be employed in the context of Dec-POMDPs. The proposed plan-time statistics for Dec-POMDPs [Oliehoek, 2013] precisely correspond to the belief in this centralized model.

Since we have shown that it is possible to generalize the sufficient plan-time statistics to zero-sum POSGs, it is reasonable to expect that a similar reduction is possible. Here we present this reduction to a special type of stochastic game, to which we refer as a Non-Observable Stochastic Game (NOSG). We do not provide the full background of the reduction for the Dec-POMDP case, but refer the reader to [Oliehoek and Amato, 2014]. The difference between that reduction and the one we present here is that the zero-sum POSG is reduced to a zero-sum stochastic game where the joint AOH acts as the state, whereas the Dec-POMDP is reduced to a POMDP.

**Definition** A plan-time Non-Observable Stochastic Game for a zero-sum POSG is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \hat{O}, R, \hat{b})$:

- A set of two agents $I = \{1, 2\}$,
- A set of augmented states $\mathcal{S}$. Each state $s^t$ corresponds to a joint AOH $\bar{\theta}^t$,
- A continuous action-space $\mathcal{A}_1$ for agent 1, containing stochastic decision rules $\delta_1^t$,
- A continuous action-space $\mathcal{A}_2$ for agent 2, containing stochastic decision rules $\delta_2^t$,
- A set of shared observations $\hat{O} = \{\text{NULL}\}$ that only contains the NULL observation,
- A transition function as specified in (4.2.3): $\hat{T}(s^{t+1}|s^t, a^t) = \Pr(\bar{\theta}^{t+1}|\bar{\theta}^t, \delta^t)$,
- A shared observation function $\hat{O}$ that specifies that observation NULL is received with probability 1,
- A reward function that corresponds to the reward function in the zero-sum POSG as specified in (4.2.1): $R(s^t, a^t) = R(\bar{\theta}^t, \delta^t)$,
- The initial belief over states $\hat{b}^0 \in \Delta(\mathcal{S})$. 

35
In the NOSG model, the agents conditions its choice on belief over the augmented states \( \hat{b} \in \Delta(\hat{S}) \), which corresponds to the belief over joint AOHs captured in the statistic \( \sigma^t \in \Delta(\Theta^t) \). As such, a value function formulation for the NOSG can be given in accordance with \((4.2.14)\). Note, that while the NULL observation is shared, the state and action contain entries for both agents in the original zero-sum POSG.

A zero-sum POSG can also be converted to a POMDP by fixing the policies of one agent [Nair et al., 2003], which leads to a model where the information state \( b(s, \theta_j) \) is a distribution over states and AOHs of the other agent. In contrast, our NOSG formulation maintains a belief over joint AOHs, and does not require fixing the policy of any agent. That is, where the approach of Nair leads to a single-agent model that can be used to compute a best-response (which can be employed to compute an equilibrium, see, e.g., [Oliehoek et al., 2006]), our conversion leads to a multi-agent model that can potentially be used to compute a Nash equilibrium directly.

Furthermore, the NOSG formulation shows that properties of zero-sum stochastic games with shared observations also hold for the zero-sum POSG. For example, Ghosh et al. [2004] show that for any zero-sum stochastic game with shared observations, a reduction to a completely observable model is possible and that in the infinite horizon case a solution (a value and rational joint policy) must exist. As the NOSG is a specific case of such a stochastic game (with one shared NULL observation), these results extend to the plan-time NOSG for our finite-horizon zero-sum POSG. One of the conditions for the games Ghosh considers, however, is that the action-spaces for both agents are metric and compact spaces. For finite-horizon zero-sum POSGs the plan-time NOSG will fulfill these requirements, but we do not prove that this will also hold in the infinite-horizon zero-sum POSG. Nonetheless, our reduction shows that some of the properties established by Ghosh et al. for a limited subset of zero-sum Stochastic Games, in fact extend to the finite-horizon zero-sum POSGs setting we consider.

### 4.2.4 Equivalence Final Stage POSG and Family of Bayesian Games

We have already shown, in chapter 3, that the value function of a Family of zero-sum Bayesian Games exhibits concavity and convexity in terms of the marginals parts of the type distribution \( \sigma \) for respectively agent 1 and 2. We formally prove that this result directly extends to the final stage of the zero-sum POSG \( t = h - 1 \).

First, we give an inductive definition of the value function of the zero-sum POSG that makes it easier to prove the aforementioned equivalence. Let \( Q_t^R \) be defined as a function that returns immediate reward function for a given statistic and joint decision rule:

\[
Q_t^R(\sigma^t, \delta^t) \triangleq \sum_{\bar{\delta}^t} \sigma^t(\bar{\theta}^t) R(\bar{\theta}^t, \delta^t) \{ \text{vec. not.} \} \delta^t \cdot R_{\delta^t}.
\]

\[(4.2.15)\]
Here, $\bar{\sigma}^t$ is vector where every entry corresponds to the probability of a joint AOH, $\sigma^t(\bar{\theta})$, and $R_\delta$ is a vector containing reward for every joint AOH, attained when agents follow the given joint decision rule $\delta^t$. Let $V_{t}$ be defined as the value for a given statistic $\sigma^t$, given that the agents follow the partial joint policy $\pi^t$:

$$V_{t}(\sigma^t, \pi^t) \triangleq \begin{cases} Q_{t}^R(\sigma^t, \delta^t) & \text{if } t = h - 1 \\ Q_{t}^R(\sigma^t, \delta^t) + V_{t+1}(U_{ss}(\sigma^t, \delta^t), \pi^{t+1}) & \text{otherwise.} \end{cases} \quad (4.2.16)$$

Here, $U_{ss}$ is the statistic update function derived from (4.2.8). Note, that through the statistic update, the future value is dependent on the decision rule $\delta^t$. Using (4.2.16), we define the value function of the zero-sum POSG as a maxminimization over joint partial policies $\pi^t$. By the minmax-theorem [Von Neumann and Morgenstern, 2007], the minmax-value and maxmin-value of the game induced by a statistic (i.e., the problem of selecting rational partial policies $\pi^t_1$ and $\pi^t_2$ for a given $\sigma^t$) is equal to the maxmin-value:

$$V^*_t(\sigma^t) \triangleq \max_{\pi^t_1} \min_{\pi^t_2} V_{t}(\sigma^t, (\pi^t_1, \pi^t_2)) = \min_{\pi^t_2} \max_{\pi^t_1} V_{t}(\sigma^t, (\pi^t_1, \pi^t_2)). \quad (4.2.17)$$

Using these definitions, we prove the equivalence of the final stage of a zero-sum POSG and a Family of zero-sum Bayesian Games.

**Lemma 4.2.2.** For a Family of zero-sum Bayesian Games $F$, if

1. joint actions $a \in A$ in the BG are equal to joint actions of the POSG,
2. joint types $\theta \in \Theta$ correspond to AOHs $\bar{\theta}^h-1 \in \bar{\Theta}^h-1$,
3. the initial distribution over types $\sigma$ is equal to $\sigma^h-1$,

then the value function at the final stage of the zero-sum POSG, $V^*_{h-1}$, and the value function of the family of Bayesian Games, $V^*_F$, are equivalent.

**Proof** The proof is listed in Appendix A. \qed

By the results of Theorem 3.2.2 (i.e. that the value function $V^*_F$ exhibits concavity and convexity in marginal-spaces of agent 1 and 2 respectively) and Lemma 4.2.2, $V^*_{h-1}$ is concave in $\Delta(\bar{\Theta}^h-1)$ for all $\sigma^h-1_c$, and convex in $\Delta(\bar{\Theta}^h-1_2)$ for all $\sigma^h-1_c$.2.

Even though the final stage is equal to a family of Bayesian Games, our approach is substantially different from approaches that represent a POSG as a series of BGs [Emery-Montemerlo et al., 2004] and derivative works [Oliehoek et al., 2008]. In fact, all other stages ($0$ to $h-2$) cannot be represented as a Family of BGs as defined in section 3.1.

We will show this using the previously established Q-value function definitions. Consider the definition of the Q-value of the zero-sum POSG for stages $t = 0, \ldots, h - 2$:

$$Q^*_t(\sigma^t, \delta^t) \{4.2.16\} Q_{t}^R(\sigma^t, \delta^t) + V_{t+1}(U_{ss}(\sigma^t, \delta^t)).$$
If the agents follow two different joint decision rules $\delta_t$ and $\delta_t''$ at a statistic $\sigma_t$, this will likely result in two different statistics at the next stage $\sigma_{t+1} \neq \sigma_{t+1}''$ via the statistic update function $U_{ss}$, and therefore, a different future value $V_{t+1}^*$. This value is largely dependent on the rational policy at that stage, $\pi_{t+1}^*$, which may be completely different for the two different statistics. Therefore, with the exception of the final stage of the zero-sum POSG, for which there is no future value, the relation between a decision rule and the value is non-linear.

Now consider the Q-value of the Bayesian Game. Let $\vec{\delta}$ be a vector, where each entry corresponds to an entry in the decision rule $\delta(a|\theta)$. Let $R$ be a vector where each entry corresponds to the reward $R(\theta, a)$ multiplied by the probability that a type is reached, $\sigma(\theta)$. We rewrite the Q-value definition as an inner product, from which we observe that in the Bayesian Game, the relation between the decision rule and the value is linear:

$$Q_{BG}(\delta) = \sum_\theta \sigma(\theta) \sum_a \delta(a|\theta) R(\theta, a) = \sum_\theta \sum_a \delta(a|\theta) \sigma(\theta) R(\theta, a) = \vec{\delta} \cdot R_{\sigma}. $$

As the relation between a $\delta_t$ and the value in the zero-sum POSG is non-linear, but the relation between a $\delta$ and value in the Bayesian Game is linear, stages 0 to $h - 2$ of the zero-sum POSG can not be modeled as Families of zero-sum Bayesian Games. Nevertheless, we show that the value function at the preceding stages exhibits the same concave and convex properties as the value function at the final stage of the zero-sum POSG.

### 4.2.5 Structure in the Value Function

Similar to the decomposition of the distribution over types in BGs (in equations 3.2.4 and 3.2.5), the plan-time sufficient statistic $\sigma^t$ can be decomposed into a marginal term $\sigma_{m,i}^t$ and conditional term $\sigma_{c,i}^t$ as follows:

$$\sigma_{m,1}^t(\vec{\theta}_1) \triangleq \sum_{\vec{\theta}_2} \sigma^t(\langle \vec{\theta}_1, \vec{\theta}_2 \rangle),$$

$$\sigma_{c,1}^t(\vec{\theta}_2|\vec{\theta}_1) \triangleq \frac{\sigma^t(\langle \vec{\theta}_1, \vec{\theta}_2 \rangle)}{\sum_{\vec{\theta}_2} \sigma^t(\langle \vec{\theta}_1, \vec{\theta}_2 \rangle)} = \frac{\sigma_{m,1}^t(\vec{\theta}_1)}{\sigma_{m,1}^t(\vec{\theta}_1)}. $$

Counterparts for the opposing agent, $\sigma_{m,2}^t$ and $\sigma_{c,2}^t$, exist. By definition, we have:

$$\sigma^t(\langle \vec{\theta}_1, \vec{\theta}_2 \rangle) = \sigma_{m,1}^t(\vec{\theta}_1) \sigma_{c,1}^t(\vec{\theta}_2|\vec{\theta}_1) = \sigma_{m,2}^t(\vec{\theta}_2) \sigma_{c,2}^t(\vec{\theta}_1|\vec{\theta}_2).$$

We will write $\sigma^t = \sigma_{m,i}^t \sigma_{c,i}^t$ for short. Let $\vec{\sigma}_{m,i}^t$ be the vector notation of the marginal statistic $\sigma_{m,i}^t$. Its entries are the probability of individual AOHs, given by $\sigma_{m,i}^t(\vec{\theta}_1)$. 

38
We will prove formally that the value function of the two-player zero-sum POSG exhibits two properties: It is, at every stage $t$, concave in marginal-space for agent 1, $\Delta(\vec{\Theta}_1^t)$, and convex in marginal-space for agent 2, $\Delta(\vec{\Theta}_2^t)$. We give an inductive definition of the value in terms of a statistic and a joint partial policy, $V_t$, that makes the distinction between the immediate reward and the value propagated from future stages explicit. Then, we define best-response value functions $V_t^{BR1}$ and $V_t^{BR2}$, and show that they are linear in the respective marginal-spaces. Using this result, we prove that $V_t^*$ exhibits concave and convex properties.

Figure 4.2.1 provides intuition on how the best-response value functions relate to the concave and convex value function. Each vector corresponds to exactly one best-response value function for a given opponent policy. These vectors can be used to construct a concave value function in $\Delta(\vec{\Theta}_1^t)$, or a convex function in $\Delta(\vec{\Theta}_2^t)$. As we will show, selecting a ‘slice’ in statistic-space corresponding to a single conditional $\sigma_{c,1}^t$ guarantees concavity, while selecting a single $\sigma_{c,2}^t$ guarantees convexity. Note, that this visualization is similar to the visualization of value function of the Family of zero-sum Bayesian Games shown in Figure 3.2.1.

Best-response value functions in terms of $\sigma^t$ and individual policies $\pi_i^t$ are defined as $V_t^{BR1}$ and $V_t^{BR2}$, similar to (3.2.1) and (3.2.2), for agent 1 and agent 2 respectively. By (4.2.17) and the minmax-theorem, the following holds (similar to (3.2.3)):

$$V_t^*(\sigma^t) = \min_{\pi_2^t} V_t^{BR1}(\sigma^t, \pi_2^t) = \max_{\pi_1^t} V_t^{BR2}(\sigma^t, \pi_1^t).$$  \hspace{1cm} (4.2.21)

Let us define a vector that contains the value (immediate reward and future value) for agent 1 for each individual AOH $\vec{\theta}_1^t$, given that agent 2 follows the partial policy $\pi_2^t$:
we can write these functions as the inner products of a marginal σ linear in the respective marginal-spaces.

\[ \nu_{\sigma^{t+1}_{c,1}, \sigma^{t+1}_{c,2}}(\vec{\theta}^{t+1}_1) = \max_{a_1} \left[ \sum_{\vec{\theta}^{t+1}_2} \sigma^t_{c,1}(\vec{\theta}^{t+1}_2|\vec{\theta}^t_1) \sum_{a_2} \delta^t_2(a^t_2|\vec{\theta}^t_2) \left( R((\vec{\theta}^t_1, \vec{\theta}^t_2), (a^t_1, a^t_2)) + \sum_{a'_1, a'_2} \Pr((\langle o^{t+1}_1, o^{t+1}_2|\langle \vec{\theta}^{t+1}_1, \vec{\theta}^{t+1}_2\rangle, (a^t_1, a^t_2))\nu_{\sigma^{t+1}_{c,1}, \sigma^{t+1}_{c,2}}(\vec{\theta}^{t+1}_1, a^t_1, o^{t+1}_1) \right) \right] \quad (4.2.22) \]

Note, that this is a recursive definition, as \( \vec{\theta}^{t+1}_1 = \langle \vec{\theta}^t_1, a^t_1, o^{t+1}_1 \rangle \), and that \( \delta^t_1 \) is not present in the equation. The reason that \( \delta^t_1 \) can be omitted is that conditional \( \sigma^{t+1}_{c,1} \) is independent of the selected decision rule of agent 1, \( \sigma^t_1 \). We prove this formally.

**Lemma 4.2.3.** Conditionals \( \sigma^{t+1}_{c,1} \) are independent of the decisions of agent i and \( \sigma^t_{m,i} \):

1. Conditional \( \sigma^{t+1}_{c,1} \) depends only on \( \sigma^t_{c,1} \) and the decision rule of agent 2, \( \sigma^t_2 \).
2. Conditional \( \sigma^{t+1}_{c,2} \) depends only on \( \sigma^t_{c,2} \) and the decision rule of agent 1, \( \sigma^t_1 \).

**Proof** The proof is listed in Appendix A.

We have established in Lemma 4.2.2 that the value function at the final stage \( t = h - 1 \) is equivalent to the value function of a Family of zero-sum BGs. Thus, the zero-sum POSG value vector from (4.2.22) reduces to the value vector for a Family of zero-sum BGs from (3.2.7) when we make the substitutions of Lemma 4.2.2:

\[ \nu_{\sigma^{h-1}_{c,1}, \sigma^{h-1}_{c,2}}(\vec{\theta}^{h-1}_1) = r_{\sigma^{h-1}_{c,1}, \sigma^{h-1}_{c,2}}(\vec{\theta}^{h-1}_1). \quad (4.2.23) \]

Intuitively, this also makes sense, as at the final stage the future value is zero and the partial policy \( \pi^{h-1}_2 \) only contains a single decision rule \( \delta^{h-1}_2 \). The value vector \( \nu_{\sigma^{h-1}_{c,2}, \pi^t_1} \) is defined similar to (4.2.22), and contains an entry for every AOH \( \vec{\theta}^t_2 \) in \( \hat{\Theta}^t_2 \).

We show that the best-response value functions \( V^{BR1}_t \) and \( V^{BR2}_t \) are linear in their respective marginal-spaces.

**Lemma 4.2.4.** (1) \( V^{BR1}_t \) is linear in \( \Delta(\hat{\Theta}^t_1) \) for a given \( \sigma^t_{c,1} \) and \( \pi^t_2 \), and \( V^{BR2}_t \) is linear in \( \Delta(\hat{\Theta}^t_2) \) for a given \( \sigma^t_{c,2} \) and \( \pi^t_1 \), for all stages \( t = 0, \ldots, h - 1 \). More specifically, we can write these functions as the inner products of a marginal \( \sigma^t_{m,i} \) and a vector:

\[ 1. \; V^{BR1}_t(\sigma^t_{m,1}, \sigma^t_{c,1}, \pi^t_2) = \sigma^t_{m,1} \cdot \nu_{\sigma^t_{c,1}, \pi^t_2}; \quad (4.2.24) \]
\[ 2. \; V^{BR2}_t(\sigma^t_{m,2}, \sigma^t_{c,2}, \pi^t_1) = \sigma^t_{m,2} \cdot \nu_{\sigma^t_{c,2}, \pi^t_1}; \quad (4.2.25) \]

**Proof** We prove this by induction. By the result of Lemma 4.2.2, we know the value function at stage \( t = h - 1 \) to be equivalent to that of a family of BGs. As such, the result of Lemma 3.2.1, which shows that a best-response value function in the Family of zero-sum Bayesian Games is linear in the corresponding marginal-space, acts as a base case for the proof. The full proof is listed in Appendix A.

\[ \square \]
Using this result, we prove that $V_t^*$ exhibits concavity in $\Delta(\tilde{\Theta}_1^t)$ at every stage for every $\sigma_{c,1}^t$, and convexity in $\Delta(\tilde{\Theta}_2^t)$ for every $\sigma_{c,2}^t$.

**Theorem 4.2.5.** $V_t^*$ is (1) concave in $\Delta(\tilde{\Theta}_1^t)$ for a given $\sigma_{c,1}^t$, and (2) convex in $\Delta(\tilde{\Theta}_2^t)$ for a given $\sigma_{c,2}^t$. More specifically, $V_t^*$ is respectively a minimization over linear functions in $\Delta(\tilde{\Theta}_1^t)$ and a maximization over linear functions in $\Delta(\tilde{\Theta}_2^t)$:

1. $V_t^*(\sigma_{m,1}^t, \sigma_{c,1}^t) = \min_{\pi_2} \left[ \sigma_{m,1}^t \cdot \nu[\sigma_{c,1}^t, \pi_2^t] \right]$,
2. $V_t^*(\sigma_{m,2}^t, \sigma_{c,2}^t) = \max_{\pi_1} \left[ \sigma_{m,2}^t \cdot \nu[\sigma_{c,2}^t, \pi_1^t] \right]$.

**Proof** Filling in the result of Lemma 4.2.4 gives:

$$V_t^*(\sigma_{m,1}^t, \sigma_{c,1}^t) \stackrel{(4.2.21)}{=} \min_{\pi_2} V_t^{BR1}(\sigma_{m,1}^t, \sigma_{c,1}^t, \pi_2^t) \stackrel{(4.2.24)}{=} \min_{\pi_2} \left[ \sigma_{m,1}^t \cdot \nu[\sigma_{c,1}^t, \pi_2^t] \right].$$

The proof for item 2 is analogous to that of item 1.

The importance of this theorem is that it may enable the development of new solution methods for these classes of games. To draw the parallel, many POMDP solution methods successfully exploit the fact that a POMDP value function is piecewise-linear and convex in belief-space [Pineau et al., 2006; Spaan and Vlassis, 2005] (which is similar to the statistic-space in the zero-sum POSG), and recently such results have been extended to the decentralized cooperative case [MacDermed and Isbell, 2013; Dibangoye et al., 2013]. Therefore, we hope to exploit the found structure by treating the zero-sum POSG as a sequence of smaller problems: the selection of a rational joint decision rule $\delta_t^*$ at every stage. In the next chapter, we will propose a method that makes use of the found structure.
Chapter 5

Exploiting the Identified Structure

The theoretical results from chapter 4 imply that within ‘slices’ of statistic-space that correspond to a single conditional \( \sigma_{c,i}^t \), it could be possible to approximately compute the value function \( V_t^* \) using a piecewise-linear and concave or piecewise-linear and convex function (as has been visualized in Figure 4.2.1). Therefore, using the value vectors as defined in (4.2.22), it may be possible to adapt point-based solution methods for POMDPs [Spaan and Vlassis, 2005] or Dec-POMDPs [MacDermed and Isbell, 2013] for the zero-sum POSG setting.

First, however, we need to identify for which conditionals we want to find new value vectors. Obviously, we want to keep the number of these conditionals as small as possible, as it is likely that computing value vectors is computationally expensive. Furthermore, we do not want to invest time and resources in computing value vectors for a conditional, only to find out later that this conditional will never be reached by a rational past policy.

We will show in section 5.1, it is possible to perform a tree-search in the space of conditionals. In subsequent sections, we propose a heuristic method that guides this search, and one random method that acts as baseline for performance. Performance of these methods will be tested on two different domains in section 5.4. We then provide discussion on the results, and analyze the shortcomings of the proposed heuristic.

As a baseline for performance, we convert the zero-sum POSG to sequence form representation and solve the resulting problem. Koller et al. [1994] show that it is possible to solve this game in polynomial time in the size of the game tree, which, in the case of the zero-sum POSG, is linearly related the number of AOHs (see also section 2.3.4). Although this time complexity is hard to beat, zero-sum POSGs of large horizon quickly become intractable when using sequence form representation, as the size of the game tree is usually highly exponential in the number actions and observations. — indicating that there is a need for alternative approaches. We will show this empirically in chapter 6.
5.1 Trees of Conditional Statistics

In chapter 4, we established an inductive value function definition for the zero-sum POSG. Nevertheless, exploiting the found concave and convex structure in order to find a Nash Equilibrium and corresponding value is not straightforward.

For the Dec-POMDP setting, approximate approaches that treat the problem as a series of Bayesian Games exist [Emery-Montemerlo et al., 2004]. These can make reasoning about large-horizon problem more tractable, as solving multiple one-stage games may be computationally cheaper than solving one multi-stage game. However, such approaches can not be directly extended to the zero-sum POSG setting when the value function is defined in terms of the plan-time sufficient statistic. One of the main reasons for this is that there are infinitely many statistics. If we assume that each statistic induces a one-shot game, then it is impossible to do brute-force evaluation, as this would require us to solve infinite amount of one-shot games.

Furthermore, we have to be careful about the way the one-shot games are represented, as the problem of selecting a rational decision rule for a given statistic is not exactly equivalent to the problem of decision-making in a Bayesian Game: in the zero-sum POSG, a change in decision rule at stage $t$ is non-linearly related to the change in value, unlike in the BG (as discussed in more detail in section 4.2.4). If we are to find the rational decision rule $\delta^t_*$ for a statistic $\sigma^t$, then we will need to evaluate $Q^t_1(\sigma^t, \delta^t)$, which involves evaluation of $V^*_t(\sigma^{t+1})$, and therefore $V^*_t$, and so on until $t = h - 1$. It is likely computationally expensive to perform such computations often. Therefore, it is not practical to search in statistic-space.

Instead, we will search the space of conditionals $\sigma^t_{c,i}$. Not only do we have some guarantees about the structure of the value function for a single conditional (it is concave or convex in marginal-space), we know by the result of Lemma 4.2.3 that conditional $\sigma^t_{c,i}$ is not affected by the decisions of agent $i$. Let us define a function $U_c$ that captures this property. It returns the conditional at the next timestep $t + 1$ given a conditional at the current timestep and the decision rule of the opposing agent. For agent 1, it is defined as $\sigma^{t+1}_{c,1} \triangleq U_c(\sigma^t_{c,1}, \delta^t_2)$. Analogously, an update function for conditional $\sigma^t_{c,2}$ given $\delta^t_1$ exists.

Using these function, it is possible to construct a search tree $T_i$ where nodes correspond to conditionals, $\sigma^t_{c,i}$, and edges correspond to decision rules of the opposing agent, $\delta^t_{i-1}$. Obviously, at the final stage of the zero-sum POSG, there are no future nodes. Instead, each decision rule $\delta^h_{i-1}$ corresponds to exactly one value vector, as defined in 4.2.23. As we have shown in section 4.2.1, such a value vector describes the best-response value function against the given decision rule $\delta^h_{i-1}$. A visualization of such a tree is given in Figure 5.1.1. We will refer to the tree of conditionals $\sigma^t_{c,1}$ with edges corresponding to $\delta^t_2$ as ‘$T_1$’. Analogously, we refer to the tree of conditionals $\sigma^t_{c,2}$ as ‘$T_2$’. Note, that it
is not impossible for two different paths to reach the same node: the tree is a so-called Directed Acyclic Graph (DAG). This observation, while interesting, is not directly useful in this chapter, as the method we define does not exploit this fact. Instead, we will discuss possible implications in section 6.2.1.

Through the backup operation given in (4.2.22), value vectors at stage \( t = h - 1 \) can be back-propagated from node \( \sigma_{c,i}^{h-1} \) to the parent node \( \sigma_{c,i}^{h-2} \). The resulting value vector will give the best-response value against the policy \( \pi_{c,i}^{h-2} = (\delta_{c,i}^{h-2}, \delta_{c,i}^{h-1}) \), where \( \delta_{c,i}^{h-2} \) is the decision rule that corresponds to the edge that connects the leaf node \( \sigma_{c,i}^{h-1} \) to its parent node. This process is repeated until the root node \( \sigma_{c,i}^0 \) is reached, resulting in a value vector that gives the best-response value against a full policy \( \pi_{-i} \) (this policy is encoded by the path from the root node to the leaf node).

Let \( \Pi_{\sigma_{c,i}^t} \) be the set individual partial policies encoded by the paths from node \( \sigma_{c,i}^t \) to leaf nodes. At every node in \( T_1 \), we can collect the value vectors, each of which corresponds to a partial individual policy \( \pi_{c,i}^t \). As agent 2 minimizes over these policies, the collected value vectors describe the following approximation of the value function.

\[
\nabla_t (\sigma_{m,1} \sigma_{c,1}^t) \triangleq \min_{\pi_{c,1}^t} \left[ \bar{\sigma}^t_{m,1} \cdot \nu_{[\sigma_{c,1}^t, \pi_{c,1}^t]} \right], \quad (5.1.1)
\]

Let us prove formally that \( \nabla_t \) is an upper bound on the value function \( V_t^* \).

**Lemma 5.1.1.** \( V_t^*(\sigma^t) \leq \nabla_t (\sigma^t) \forall \sigma^t \in \Delta(\Theta^t) \).

**Proof** By the result of Theorem 4.2.5, we have \( V_t^*(\sigma_{m,1} \sigma_{c,1}^t) = \min_{\pi_{c,1}^t} \left[ \bar{\sigma}^t_{m,1} \cdot \nu_{[\sigma_{c,1}^t, \pi_{c,1}^t]} \right] \). This is a minimization over the set of all stochastic individual partial policies starting at stage \( t \).

Let us refer to this set as \( \Pi_{\sigma_{c,1}^t}^2 \). Obviously, we have \( \Pi_{\sigma_{c,1}^t} \subseteq \Pi_{\sigma_{c,1}^t}^2 \). Therefore:

\[
\min_{\pi_{c,1}^t \in \Pi_{\sigma_{c,1}^t}^2} \left[ \bar{\sigma}^t_{m,1} \cdot \nu_{[\sigma_{c,1}^t, \pi_{c,1}^t]} \right] \leq \min_{\pi_{c,1}^t \in \Pi_{\sigma_{c,1}^t}} \left[ \bar{\sigma}^t_{m,1} \cdot \nu_{[\sigma_{c,1}^t, \pi_{c,1}^t]} \right],
\]

\[
V_t^*(\sigma_{m,1} \sigma_{c,1}^t) \leq \nabla_t (\sigma_{m,1} \sigma_{c,1}^t). \quad \Box
\]

Analogously, a lower bound on the value function \( V_t^* \) is defined for every conditional \( \sigma_{c,2}^t \) (corresponding to a node in \( T_2 \)) as a maximization over partial policies of agent 1:

\[
\nabla_t (\sigma_{m,2} \sigma_{c,2}^t) \triangleq \max_{\pi_{c,2}^t \in \Pi_{\sigma_{c,2}^t}} \left[ \bar{\sigma}^t_{m,2} \cdot \nu_{[\sigma_{c,2}^t, \pi_{c,2}^t]} \right], \quad (5.1.2)
\]

For \( T_1 \), we show the process of computing and subsequently back-propagating the value vectors in Figure 5.1.2. As there are space of conditionals is continuous, there are infinitely many nodes. As such, the search tree will be infinitely wide, rendering a brute-force search through \( T_1 \) impossible. Heuristic methods, which guide the search in the hope that only a
The subset of all nodes should be evaluated, may prove useful here. In the next section, we propose such a method.

The idea of using value vectors to find bounds on the value function is similar to notions from online planning algorithms for POMDPs Smith and Simmons [2004]; Ross et al. [2008] that minimize an upper bound on the (piecewise-linear) convex POMDP value function by choosing actions that maximize the value given by the upper bound. If the value associated with such an action turns out to be lower than previously thought, the corresponding value vector will be pushed down. A different value vector may then become the maximizing value vector, and the process repeats itself. As such, these methods guarantee that the upper bound will eventually converge to the value function.

However, in the zero-sum POSG case, we aim to minimize the concave upper bound instead. If the value vectors are somewhat representative of the actual value function, then this need not be a problem. However, the upper bound \( V_{t+1} \) is concave in \( \sigma_{m,1} \)-space, and agent 2 can not directly influence the marginal \( \sigma_{m,1} \). The most effective play for agent 2 is therefore to find a conditional \( \sigma_{c,1}^t \) for which the value function has the lowest peak — this ensures that even if the uncertainty about the AOH of agent 1 increases, the value is kept as low as possible. If we only let agent 2 select conditionals where the bound is already low, we have no guarantee that the bounds at different conditionals (at which the upper bound is relatively high) eventually converge to the real value.

![Figure 5.1.1: A tree of conditional nodes \( \sigma_{c,i}^t \), where edges correspond to decision rules of the opposing agent, \( \delta_{-i} \).](image)

45
Chapter 5. Exploiting the Identified Structure

Approaches

Figure 5.1.2: A visualization of the process of back-propagating value vectors from the final stage to the root node. First, we find the value vectors at stage $h-1$ through heuristic search in the tree of conditionals $T_1$. Then, we backup these value vectors to the root node, collecting value vectors in every node $\sigma^t_c$ along the way. A minimization over the found vectors describes a piecewise linear and concave (convex for $T_2$) function for every node: the upper bound $V_0$ on the value function.

5.2 Approaches

In the previous section, we showed that it is possible to construct two search trees (one for each agent), where nodes correspond to conditionals $\sigma^t_c$ and edges correspond to decision rules of the opposing agent $\delta^-_t$, and discussed how we can compute value vectors. The question we aim to answer in this section is: for which conditionals do we want to compute a value vector?

We propose two methods that perform a heuristic search in the space of conditionals. The first method performs random exploration of the search tree, and acts a baseline for performance. The second method uses a heuristic of our design to select new decision rules, which, through the update function $U_c$, allow us to find new conditional nodes. We compare their performance in terms of runtime and accuracy, where we provide ground truth for the value using sequence form representation.
5.2.1 Approach ‘Random’: Sampling Decision Rules at Random

The first method we propose constructs search trees $T_1$ and $T_2$ through random sampling of decision rules. On every iteration, it performs randomized depth-first search, and adds nodes with a predefined probability (controlled by a parameter $\alpha$). In many ways, it is similar to random sampling of full policies, which is obviously a naive method of solving the problem. However, it does provide a baseline for performance, and allows us to empirically validate the idea of constructing a tree of conditionals.

We explain the steps for $T_1$. On every iteration, we perform a randomized depth first search. At every node we encounter during the search, we are presented with two choices: either we expand the current node (with probability $\alpha$), or we move to one of the child nodes (with probability $1 - \alpha$). Exceptions are nodes that have no children yet, at which we have no choice but to expand. By ‘expand’, we mean sampling of a (new) decision rule $\delta_{t+1}$, from which we obtain the next conditional as $\sigma_{t+1}$.

This vector is propagated to the first stage using (4.2.22), where it corresponds to a individual policy for the opponent, $\pi_{-i}$. Initially, the search tree $T_1$ contains only a root node, corresponding to the conditional $\sigma_0$. As it does not have any child nodes (yet), the first operation is always to ‘expand’ the current node.

The main procedure is given in algorithm 5.1. It performs a randomized depth-first search in the tree of conditionals, guided by a given parameter $\alpha$. The method chooses whether we move to a next node in the tree (with probability $\alpha$), or if we should ‘Expand’ the current node (with probability $1 - \alpha$).

The ‘Expand’ procedure is given in algorithm 5.2. It takes as input a node of a tree and the horizon $h$, and returns a value vector at stage $t = h - 1$. It picks a new decision rule using the procedure ‘getNewDecisionRule’, which samples a new decision rule at random. The node is updated using the conditional update function $U_c$. Of course, if a new-found decision rule has already been added to the node, we move to the next node using the

\[ \nu_{\sigma_{c,1}^{h-1}, \sigma_{c,2}^{h-1}}(\bar{\theta}_1^{h-1}) = \max_{a_1^{h-1}} \left[ \sum_{\bar{\theta}_2^{h-1}} \sum_{a_2^{h-1}} \delta_{t}^{h-1}(\bar{\theta}_2^{h-1} | \bar{\theta}_1^{h-1}) \sigma_{c,1}^{h-1}(a_2^{h-1} | \bar{\theta}_2^{h-1}) R((\bar{\theta}_1^{h-1}, a_1^{h-1}), (a_2^{h-1}, a_2^{h-1})) \right]. \] (5.2.1)
corresponding edge instead of adding duplicate decision rules. Note, that the Expand procedure is recursive, and that the process is repeated until stage \( h - 1 \) is reached.

The resulting value vector is propagated to the first stage using the Backup procedure, given in algorithm 5.3. This procedure collects the value vectors at every stage of the tree, and prunes any dominated vectors. This allows us to stop the Backup procedure if the resulting value vectors are dominated, thus saving runtime.

As stated earlier, the Random method selects decision rules (and indirectly, conditionals) at random, which is not so different from random policy sampling. The method we propose in the next subsections aims to select decision rules in a more intelligent manner.

**Algorithm 5.1** SolvePOSGConcaveConvex: Randomized depth first search

**Input:** horizon \( h \), tree-traversal parameter \( \alpha \), maximum runtime

**Output:** Upper bound \( V \) and lower bound \( V_0 \) on the value.

1: Initialize \( T_1 = \sigma_{c,1}^0, T_2 = \sigma_{c,2}^0 \) \# initialize trees with root nodes
2: \( i = 1 \) \# start with agent 1
3: \( \text{node} = \sigma_{c,i}^0 \) \# initialize ‘node’ as root
4: while time left do
5:   if random(0,1) > \( \alpha \) and not node.children.empty then
6:     \( \text{node} = \text{random.choice(node.children)} \)
7: else
8:   \( \sigma_{c,-i}^{h-1}, v_{\sigma_{c,-i}^{h-1},\pi_{-i}^{h-1}} = \text{Expand}(	ext{node}, h) \) \# get terminal node, value vector
9:   Backup(\( \sigma_{c,i}^{h-1}, v_{\sigma_{c,i}^{h-1},\pi_{-i}^{h-1}} \)) \# backs up value vector
10:  \( i = \text{modulo}(i, 2) + 1 \) \# switch agent
11:  \( \text{node} = \sigma_{c,i}^0 \) \# restart search: set node = root
12: end if
13: end while
14: \( V = \min(\sigma_{c,1}^0.\text{valueVectors}) \)
15: \( V_0 = \max(\sigma_{c,2}^0.\text{valueVectors}) \)
16: return \((V, V_0)\)

**Algorithm 5.2** Expand: Recursive expansion of a given node

**Input:** node \( \sigma_{c,i}^{h-1} \), horizon \( h \)

**Output:** node \( \sigma_{c,i}^h \), value vector \( v_{\sigma_{c,i}^h,\pi_{h-1}}^t \)

1: \( \delta_{-i}^t = \text{getNewDecisionRule()} \)
2: if \( t < h - 1 \) then
3:   \( \sigma_{c,i}^{t+1} = U_t(\sigma_{c,i}^t, \delta_{-i}^t) \)
4:   node.addChildAndVertex(\( \sigma_{c,i}^{t+1}, \delta_{-i}^t \))
5:   \( \text{node} = \sigma_{c,i}^{t+1} \)
6: return \( \text{Expand}(	ext{node}, h) \)
7: else
8:   \( v_{\sigma_{c,i}^{t+1},\pi_{-i}^{t+1}} = \text{computeValueVector}(	ext{node}, \delta_{-i}^{h-1}) \) \# using (5.2.1)
9: return \( \text{node}, v_{\sigma_{c,i}^{t+1},\pi_{-i}^{t+1}} \)
10: end if

48
Algorithm 5.3 Backup: Recursive backup of value vectors

| Input:  | Node $\sigma_{c,1}^{t+1}$, Value vector $v|_{\sigma_{c,1}^{t+1}, \pi_{-c}^t}$ |
|---------|---------------------------------------------------------------|
| 1:      | $v|_{\sigma_{c,1}^{t+1}, \pi_{-c}^t} = \text{backupVector}(v|_{\sigma_{c,1}^{t+1}, \pi_{-c}^t})$ # using (4.2.22) |
| 2:      | node = node.parent |
| 3:      | if $v|_{\sigma_{c,1}^{t+1}, \pi_{-c}^t}$ not dominated by node.valueVectors then |
| 4:      | node.valueVectors.add($v|_{\sigma_{c,1}^{t+1}, \pi_{-c}^t}$) |
| 5:      | node.valueVectors.pruneDominated() |
| 6:      | if $t > 0$ then |
| 7:      | Backup(node, $v|_{\sigma_{c,1}^{t+1}, \pi_{-c}^t}$) |
| 8:      | end if |
| 9:      | end if |

5.2.2 Approach ‘Informed’: Informed Decision Rule Selection

Of course, the Random method will only act as baseline for performance, as sampling decision rules at random is not a smart way to find the Nash Equilibrium. The heuristic we propose here uses the best policies found at runtime as a heuristic for decision rule selection. The new decision rule selection method implements the ‘getNewDecisionRule’ method in the Expand procedure (in algorithm 5.2). Apart from that, the method is identical to the Random method in all aspects.

The basic idea is to select decision rules that are good responses to the decision rules encoded in the tree of the opposing player. For example, we want to select new decision rules $\delta_2^{t+1}$ that secure the least payoff for agent 1 against the decision rules $\delta_1^t$ that correspond to edges in $T_2$. Motivation for this choice is that at runtime, we know the approximate value associated with following these decision rules through the corresponding value vectors: every node in $T_2$ has collected one or more value vectors that describe a lower bound on the value function. This lower bound is our current approximation of the value function, so we let agent 2 select the decision rule $\delta_2^{t+1}$ that minimizes the lower bound. Analogously, new decision rules for agent 1, to be added to $T_2$, are found by maximizing the upper bound.

We explain the heuristic using an abstract example. Assume that we have, at runtime, partially constructed $T_1$ and $T_2$ (in parallel), and that we are currently performing a randomized depth first search in tree $T_1$. We have traversed the tree $T_1$ to a node $\sigma_{c,1}^t$, where we decide (with probability $\alpha$) to ‘Expand’. Thus, we now have to identify a new edge, corresponding to a decision rule $\delta_2^t$, to add to the current node. The situation is visualized in Figure 5.2.1 and Figure 5.2.2, where conditionals and decision rules are annotated with their index. For example, $\delta_{2,(0)}^{t+1}$ is the first decision rule of agent 2 at stage $t+1$, $\delta_{2,(1)}^{t+1}$ is the second one.
This path corresponds to a single past individual policy \( \varphi_2 \), encoded by the decision rules on the edges. We assume that agent 2 follows this past policy — otherwise, the current node would not have been reached.

Contained in \( T_2 \), there are a finite amounts of paths from root node \( \sigma_{c,2}^0 \) to nodes at stage \( t \), \( \sigma_{c,2}^t \). Each of these path corresponds to a past individual policy \( \pi_1^t \). We constrain agent 1 to choose one of these past policies. Together with the past individual policy for agent 2, \( \varphi_2^t \), we can compute all reachable statistics \( \sigma^t \) through successive statistic updates.

We then let agent 1 maximize over decision rules \( \delta_1^t \), each of which corresponds to a next node, \( \sigma_{c,2}^{t+1} \). For every \( \sigma_{c,2}^{t+1} \), we have a convex lower bound on the value function described by the collected value vectors. We now aim to choose \( \delta_2^t \), so, that a marginal \( \sigma_{m,2}^t \) that minimizes this lower bound is reached, taking into account that agent 1 can maximize over conditionals \( \sigma_{c,2}^{t+1} \).

Let us define this formally. Let \( \Phi_{T_2} \) be the set of all past individual policies \( \varphi_1^t \), corresponding to paths in \( T_2 \) from the root node to nodes at stage \( t \). Let \( \sigma_{c,\varphi_1^t} \) be the conditional \( \sigma_{c,2}^{t+1} \) reached by such a past individual policy through successive conditional updates, e.g., \( \sigma_{c,\varphi_1^t} = \sigma_{c,\delta^t_1} = U_c(U_c(\sigma_{c,2}^t, \delta_1^t), \delta_1^t) \). Let \( D_{\sigma_{c,\varphi_1^t}} \) be the set of decision rules \( \delta_1^t \) corresponding to the edges of the node in \( T_2 \) that has conditional \( \sigma_{c,\varphi_1^t} \). Let \( \sigma_{t, [\varphi_1^t, \varphi_2^t]} \) be the statistic reached when the agents follow the past joint policy \( \varphi^t = (\varphi_1^t, \varphi_2^t) \). Let \( \bar{Q}_t \) be defined as the lower bound on the Q-value as:

\[
\bar{Q}_t(\sigma^t, \delta^t) = Q^R_t(\sigma^t, \delta^t) + V_s(U_{ss}(\sigma^t, \delta^t)) \tag{5.2.2}
\]

An upper bound on the Q-value, \( \bar{Q}_t \), is defined similarly. The decision \( \delta_2^t \) rule we add as edge to the node \( \sigma_{c,1}^t \) is then the decision rule that minimizes the lower bound, given that agent 1 tries to maximize it:

\[
\delta_2^t = \arg \max_{\delta_2^t} \max_{\varphi_1^t \in \Phi_{T_2}} \max_{\delta_1^t \in D_{\sigma_{c,\varphi_1^t}}} \left[ \bar{Q}_t(\sigma^t, \delta^t); \left( \delta_1^t, \delta_2^t \right) \right]. \tag{5.2.3}
\]
Let $c_2$ be an unconstrained variable that represents the contribution of agent 2 towards the payoff of agent 1, according to the lower bound $\underline{Q}_t$. That is, maximizing $c_2$ maximizes the lower bound, not the value function. We then solve (5.2.3) using the following Linear Program.

$$\begin{align*}
\min_{\delta_2, c_2} & \quad c_2 \\
\text{subject to} & \quad \underline{Q}_t(\sigma^t_{[\varphi_1, \varphi_2]}, (\delta^t_1, \delta^t_2)) \geq c_2 & \forall \varphi_1 \in \Phi_{T_2}, \forall \delta^t_1 \in D_{\sigma_{c, \varphi_1}} \\
& \quad \sum_{a_2 \in A_2} \delta_2(a_2 | \bar{\theta}_2) = 1 & \forall \bar{\theta}_2 \in \bar{\Theta}_2^t \\
& \quad \delta_2(\cdot | \bar{\theta}_2) \geq 0 & \forall \bar{\theta}_2 \in \bar{\Theta}_2^t
\end{align*}$$

(5.2.4)

Similarly, $c_1$ is an unconstrained variable that represents the contribution of agent 2 towards the payoff of agent 1, according to the upper bound $\overline{Q}_t$. Agent 1 would like to maximize this contribution, which is captured in the following Linear Program.

$$\begin{align*}
\max_{\delta_1, c_1} & \quad c_1 \\
\text{subject to} & \quad \overline{Q}_t(\sigma^t_{[\varphi_1, \varphi_2]}, (\delta^t_1, \delta^t_2)) \leq c_1 & \forall \varphi_2 \in \Phi_{T_1}, \forall \delta^t_2 \in D_{\sigma_{c, \varphi_2}} \\
& \quad \sum_{a_1 \in A_2} \delta_1(a_1 | \bar{\theta}_1) = 1 & \forall \bar{\theta}_1 \in \bar{\Theta}_1^t \\
& \quad \delta_1(\cdot | \bar{\theta}_1) \geq 0 & \forall \bar{\theta}_1 \in \bar{\Theta}_1^t
\end{align*}$$

(5.2.5)

The decision rule selection procedure that uses these LPs is given in algorithm 5.4. As stated earlier, it implements the ‘getNewDecisionRule’ method on the

**Algorithm 5.4** informedDecisionRuleSelection

**Input:** node $\sigma_{c,i}^t$, tree of conditionals $T_i$

**Output:** New decision rule for the opposing agent $\delta_{-i}^t$

1: Determine $P_i^t$ and $D_{\sigma_{c,i}^t}$ from $T_i$.

2: if $i \equiv 1$ then

3: \( \varphi_2^t = \text{getPastPolicy}(\sigma_{c,1}^t) \) \# encoded by path to root node $\sigma_{c,1}^0$

4: Solve $\delta_2^t = \arg\max_{\delta_2^t} \max_{\varphi_1^t \in \Phi_1^t, \delta_1^t \in D_{\sigma_{c, \varphi_1}^1}} \left[ \underline{Q}_t(\sigma^t_{[\varphi_1^t, \varphi_2^t]}, (\delta_1^t, \delta_2^t)) \right] \) \# using LP 5.2.4

5: \( \text{return} \ \delta_2^t \)

6: else

7: \( \varphi_1^t = \text{getPastPolicy}(\sigma_{c,2}^t) \)

8: Solve $\delta_1^t = \arg\max_{\delta_1^t} \min_{\varphi_2^t \in \Phi_2^t, \delta_2^t \in D_{\sigma_{c, \varphi_2}^2}} \left[ \overline{Q}_t(\sigma^t_{[\varphi_1^t, \varphi_2^t]}, (\delta_1^t, \delta_2^t)) \right] \) \# using LP 5.2.5

9: \( \text{end if} \)

10: \( \text{return} \ \delta_1^t \)
Chapter 5. Exploiting the Identified Structure

5.3 Domains

We propose two multi-agent zero-sum games, which will be used to test our algorithms. Both games are designed with tractability in mind, meaning that they have a fairly low number of states, actions, and observations. They provide a challenge in that the rational joint policy for most horizons is non-deterministic. In line with the zero-sum POSG model, action selection is simultaneous, and agents receive noisy private observations.

5.3.1 Competitive Tiger Problem

The competitive tiger problem is a game with two states, 4 x 4 joint actions, and 3 x 3 joint observations. In this problem, two identical agents compete for treasure. The agents stand before two doors. Behind one door they will find treasure. Behind the other door, a tiger awaits them. The agents receive noisy observations about the position of the tiger, but they never observe the state directly. The main goal in this game is to open the correct door before the opponent does, or alternatively, to block the opponent that tries to open the correct door. The game is visualized in Figure 5.3.1a. It is an adaptation of a classic benchmark problem for Dec-POMDPs called the Dec-Tiger problem, introduced by Nair et al. [2003], in which two agents must cooperate in order to get the treasure.

The game has two states ‘tiger left’ and ‘tiger right’. On initialization, either of these states is picked at random. The state remains the same until either agent opens a door, after which the state is reset.

Both agents can choose to ‘open left’, ‘open right’, ‘listen’ or they can ‘block’ the opposing agent to prevent him from opening a door.

Agents will only receive indicative observations if they explicitly choose to ‘listen’. Both agents have a probability 0.85 of hearing the position of tiger correctly (they receive either ‘hear tiger left’ or ‘hear tiger right’), and probability 0.15 of receiving incorrect information. If an agent chooses any other action, he receives a dummy observation, which does not provide any information about the state. By explicitly adding this dummy observation and subsequently pruning unreachable AOHs, we can significantly decrease the number of AOHs that we must consider, as we will show in section 5.4. We show a matrix specifying observation probabilities in Table B.1.

The game is symmetrical (i.e., we do not give one agent an advantage over the other), and agents will only receive reward if they choose different actions. For agent 1, the rewards are specified as follows. Blocking correctly is rewarded +1, blocking incorrectly is rewarded -1. Opening the correct door is rewarded 2. Opening the wrong door results in -4 reward. Note that if one agent blocks a door, then this negates the effects of the opponent’s ‘open door’ action: the state remains the same, and the agents will only receive reward based on whether the ‘block’ action was correct or not. For example, if agent 1 saves the life of
agent 2 through a ‘block’ action, agent 1 will be penalized for doing so, while agent 2 will live to see another day (and is rewarded for letting agent 1 block when he should not have). The game is zero-sum, so for all states and actions, the reward for agent 2 is the additive inverse of the reward for agent 1. We show a matrix specifying rewards in Table B.1.

5.3.2 Adversarial Tiger Problem

The second game we propose is also an adaptation of the cooperative tiger problem, where instead of having two agents open doors, we let the tiger be a rational and strategic agent. The agent in front of the doors, which we will refer to as ‘adventurer’, must open one of two doors, both of which lead to treasure. The tiger aims to guard both treasures, however, he can only guard one at a time. Obviously, the adventurer aims to obtain treasure.

Unlike in the competitive tiger problem, there is now some distance between the rooms, so moving between these doors will take one timestep. The game can be modeled as a game of two states, that specify whether agents are in front of the ‘same door’ or at the ‘other door’. This state representation is slightly more compact than having the state contain the positions of both agents explicitly. Similar to the competitive tiger problem, the initial state of the game is chosen at random. The game is visualized in Figure 5.3.1b.

The adventurer chooses between three actions: ‘move’, ‘listen’ or ‘open door’. The tiger chooses between two actions: ‘move’ and ‘listen’. Both the adventurer and the tiger can move freely between the doors, but they each have a 0.1 chance of failing, in which case their position will not change (e.g., the probability that both moves fail is $0.1 \times 0.1 = 0.01$). If an agent chooses any other action, his position does not change.

The agents receives noisy observations if he performs the ‘listen’ action. There are two observations per agent, specifying whether the agent hears the opponent behind the ‘same door’ or the ‘other door’. An agent that moves receives the observation ‘other door’ regardless of the state, thereby not gaining any information. If an agent listens when the opponent moves, then the agent has a 0.6 probability of observing the resulting state, while the opponent will receive the ‘other’ observation, thereby not gaining any information. If both agents ‘listen’, then they have 0.7 probability of observing the correct state and 0.1 probability on any other observation. If the adventurer opens the door, the tiger will observe the resulting state, while the adventurer will receive an observation specifying that the opponent is behind ‘other door’ with probability 1.0. We show a matrix specifying observation probabilities in Table B.2.

If the adventurer opens the correct door, he receives a positive reward of 3. Opening the wrong door results in a reward of -5. To give the adventurer an extra incentive to open the door, he receives -1 reward on every timestep where he chooses ‘move’ or ‘listen’. We show a matrix specifying rewards in Table B.2. In the next section, we will show empirically that the tiger has the advantage in this game.
5.4 Experiments

We test the algorithms from section 5.2 on both test domains, and provide discussion on the found results. Before that, we show how the number of AOHs can be reduced.

In both domains, some observational probabilities are zero. As a result, not every combination of actions and observations results in a valid AOH. For example, if an agent in the competitive tiger problem chooses to open a door, he will never receive the observation ‘hear tiger left’. Therefore, all AOHs that contain the sequence ⟨‘open right’, ‘hear tiger left’⟩ can be skipped. Doing so decreases runtime of methods that have a dependence on the number of AOHs. For example, sequence form solving [Koller et al., 1994] benefits greatly from this exploit, as the number of information sets and sequences are both directly dependent on the number of AOHs (as we explained in section 2.1.4). In our methods, finding a best-response to a given decision rule — a step that is necessary in the backup of value vector, see (4.2.22) — involves looping over all AOHs at that stage. We show the number of reachable AOHs for both domains in Table 5.4.1. At timestep 0, we define there to be a single joint AOH, which is an empty tuple ⟨⟩. This AOH is always reachable.

![Diagram of competitive and adversarial tiger problems](image)

(a) The competitive tiger problem. (b) The adversarial tiger problem.

Figure 5.3.1: Visualization of the domains. Yellow circles represent agents.

<table>
<thead>
<tr>
<th>Problem</th>
<th>AOHs</th>
<th>timestep</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Competitive tiger</td>
<td>reachable</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>total</td>
<td>144</td>
</tr>
<tr>
<td>Adversarial tiger</td>
<td>reachable</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>total</td>
<td>24</td>
</tr>
</tbody>
</table>

Table 5.4.1: Reachable versus total amount of joint AOHs.
First, we test the effect of the parameter $\alpha$ on the performance of the solution methods. We perform five runs (each with a maximum runtime of 30 minutes), for both the competitive tiger problem and the adversarial tiger problem, horizon 3. Experiments were performed on a notebook with quad core Intel i5 processor and 8GB RAM. Figure 5.4.1 shows the mean and standard deviation of the resulting upper and lower bounds, where ‘Random’ corresponds to the method in section 5.2.1, ‘Informed’ refers to the informed decision selection from section 5.2.2.

We observe that for the Random method, the effect of a change in $\alpha$ is typically not very significant. An exception is $\alpha = 1$, where we see a sudden decrease in performance, for both the competitive tiger problem and the adversarial tiger problem. This is not surprising, as setting $\alpha$ to 1 ensures that only one decision is sampled at the root node. If this initial decision rule is bad, then the resulting bound will be loose. Interestingly, for the adversarial tiger problem, the Random method is able to let the upper bound converge to the value for almost all $\alpha$. As the tightness of the upper bound is determined by the quality of the decision rules $\delta_t^i$, this indicates that for horizon 3, random sampling allows us to find a rational strategy for agent 2 (the tiger). This indicates that it is likely that for the tiger, many rational strategies exist. Based on these experimental results, we set $\alpha = 0.3$ for both problems.

For the Informed method, results are slightly different. In the competitive tiger problem, we observe in Figure 5.4.1b that $\alpha = 0.7$ seems to give the best results. However, there is a decline in performance as $\alpha$ becomes larger, which is why we opt to set $\alpha = 0.4$. For the adversarial tiger problem, for which the results are shown in Figure 5.4.1d, we observe that the lower bound found by the Informed method seems to reach a local minimum for all $\alpha$: although the lower bound has not converged to the value, each run stops improving as the lower bound reaches -3. As the standard deviation of the upper bound is quite large, it is safe to assume that the results of the informed method depend heavily on the initialization (i.e., which decision rule is selected at random at the very first iteration). We choose to set $\alpha = 0.5$ for this problem as well, as visually, it seems to give good results.

We test the Random and Informed methods on the two given domains, for horizon 2 to 6. Experiments were assigned a maximum runtime of 10, 30, 60, 120, 240 minutes for the different horizons, respectively (chosen empirically). Results are averaged over five runs. As sequence form finds the value exactly, the corresponding experiments are only performed once, and we show the same value for the upper and lower bound.

The experimental results are given in Table 5.4.2 and Table 5.4.3. ‘Sequence form’ indicates that we solved the game using sequence form representation [Koller et al., 1994] (see also section 2.3.4). The tables give the found upper and lower bounds, where the ‘X’ indicates that no results could be find within the allotted time. For the symmetric competitive tiger problem, the value of the game is 0 for all horizons. In the asymmetric
adversarial tiger problem, this is not the case: the tiger has a small advantage.

We observe that for the competitive tiger problem of horizon 2, the Informed method does not outperform the Random method, as the distance between the bounds found by the Informed method is larger than the distance between the bounds found by the Random method. For games of horizon 3 and 4 however, the Informed method performs better. The decreasing performance of the Random method was to be expected: as the horizon grows, the policies become exponentially larger, and it is less likely that the random method selects good decision rules. We find that the competitive tiger problem of horizon 5 and 6 cannot be solved by the Informed method — solving the LP at every stage turned out to be too time-consuming. Looking at the number of joint AOHs in these problems (Table 5.4.1), this is not surprising, but regrettable nonetheless.

For the adversarial tiger problem, the Random method outperforms the Informed method at every horizon. Unlike the competitive tiger problem, the adversarial tiger problem of horizon 5 is solvable, but here too the resulting bounds are less tight than those found by the Random method. We conclude from this that while the Informed method may result in better bounds for specific problems, it does not scale well. For both problems, we find that solving the game using sequence form representation is possible for almost all horizons — the exception being the horizon 6 competitive tiger problem.

<table>
<thead>
<tr>
<th>horizon</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>0.27</td>
<td>-0.29</td>
<td>1.32</td>
<td>-1.35</td>
<td>3.01</td>
</tr>
<tr>
<td>Informed</td>
<td>1.08</td>
<td>-0.99</td>
<td>1.21</td>
<td>-1.12</td>
<td>1.97</td>
</tr>
<tr>
<td>Sequence form</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.4.2: Experimental results for the competitive tiger problem.

<table>
<thead>
<tr>
<th>horizon</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>-1.59</td>
<td>-1.63</td>
<td>-2.24</td>
<td>-2.58</td>
<td>-2.84</td>
</tr>
<tr>
<td>Informed</td>
<td>-1.41</td>
<td>-2.00</td>
<td>-1.68</td>
<td>-3.00</td>
<td>-2.21</td>
</tr>
<tr>
<td>Sequence form</td>
<td>-1.60</td>
<td>-1.60</td>
<td>-2.24</td>
<td>-2.24</td>
<td>-3.00</td>
</tr>
</tbody>
</table>

Table 5.4.3: Experimental results for the adversarial tiger problem.
To illustrate the rate of convergence of the different methods, we have visualized the upper and lower bounds for the competitive tiger problem and the adversarial tiger problem in Figure 5.4.2. Results were based on a single run.

As the competitive tiger problem is symmetric, we observe the upper and lower bound are almost equally far from the Nash Equilibrium. We observe that for the (asymmetric) adversarial tiger problem, it is generally easier to find a tight upper bound than it is to find a tight lower bound. In this game, the tightness of the upper bound depends on the policies of the tiger, which suggests that giving the tiger the advantage affects how well it can exploit the opponent.

The Random baseline, as expected, does not find the value of either game, but the bounds it finds converge steadily to the Nash Equilibrium. The competitive tiger problem is more complex than the adversarial tiger problem due to a larger number of possible actions and observations. The fact that for the competitive tiger problem, the gap between the bounds is much larger than the gap in the adversarial tiger problem confirms this. Although for both games the Random method is guaranteed to converge as we approach the limit (infinite decision rules sampled), it is unlikely that the Nash Equilibrium is found in a reasonable amount of time.

The Informed method works significantly better than Random in the competitive tiger problem, as the resulting gap between the bounds is smaller. For the adversarial tiger problem however, no new decision rules were found after a few improvements were made — the bounds quickly reach a local minimum and maximum, where, despite a gap between the upper and lower bounds, no new decision rules are selected. Clearly, in this problem, the Informed method does not work as well as the Random method. We will discuss possible causes and the shortcomings of the Informed method in section 5.5.
### Chapter 5. Exploiting the Identified Structure

**Experiments**

Figure 5.4.1: The effect of the parameter $\alpha$ on the tightness of the upper and lower bound, for both the competitive tiger problem and the adversarial tiger problem for horizon 3. Mean and variance are based on 5 runs, at discrete $\alpha$. Left figures show results for the random method, right for the Informed method. Top figures show results for the competitive tiger problem, bottom figures for the adversarial tiger problem.

#### Table: Effect of $\alpha$ on the bounds

<table>
<thead>
<tr>
<th>Parameter $\alpha$</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2.5</td>
<td>0.1</td>
</tr>
<tr>
<td>2.0</td>
<td>0.2</td>
</tr>
<tr>
<td>1.5</td>
<td>0.3</td>
</tr>
<tr>
<td>1.0</td>
<td>0.4</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>0.0</td>
<td>0.6</td>
</tr>
</tbody>
</table>

#### Graphs:

- **(a) Competitive tiger, Random method.**
- **(b) Competitive tiger, Informed method.**
- **(c) Adversarial tiger, Random method.**
- **(d) Adversarial tiger, Informed method.**
Chapter 5. Exploiting the Identified Structure

Experiments

Figure 5.4.2: Typical convergence rate for the two methods on the test problems, horizon 4. Shown are the lower bound $V$ and the upper bound $\overline{V}$ as function of the elapsed time. The Nash Equilibrium is shown as a dotted line, it is 0 for the competitive tiger problem, and -3 for the adversarial tiger problem. Left figures show results for the random method, right for the Informed method. Top figures show results for the competitive tiger problem, bottom figures for the adversarial tiger problem.
5.5 Shortcomings of the Informed Method

As the Informed method does not work well, we investigate its shortcomings. We identified two problems with this approach that are likely to cause bad performance. First, we choose decision rules that minimize a bound, rather than the real value, which may lead to inaccuracies. Second, the chosen decision rules are only secure against a finite set of opponent policies, and there are no guarantees that they will eventually be secure against all opponent policies.

5.5.1 Decision Rule Selection Based on Bounds instead of Value

Consider the example scenario displayed in Figure 5.2.1 and Figure 5.2.2. During ‘informed’ sampling of decision rules, the decision rule that we would like to add to $T_1$ is the following:

$$
\delta_{t^*}^2 = \arg\min_{\delta_1^t} \max_{\delta_2^t} \left[ Q_t^*(\sigma^t, \langle \delta_1^t, \delta_2^t \rangle) \right].
$$

However, as we generally do not know $Q_t^*$ at runtime, we instead find a decision rule that minimizes the lower bound on the $Q$-value. The problem is that the resulting decision rule is not necessarily the decision rule that minimizes the value. This is visualized in Figure 5.5.1a. Value vectors for one conditional $\sigma_{t+1}^{c,2}$ give a lower bound on the value function, $V_{t+1}^{c,2}$, in red. The real value function $V_{t+1}^{*}$ is shown in green. The minimizing agent would like to ensure that he reaches the marginal $\sigma_{m,2}^{t+1}$ that results in the minimal value, indicated here with a green dotted line. To do so, he will have to make decisions on all preceding stages $0, \ldots, t$ that, through statistic-updates, allow this marginal to be reached. However, these decisions are based on the lower bound, which will — in this example scenario — eventually result in a very different marginal.

We expect that this effect can be reduced somewhat if we ensure that the bounds are within a certain distance $\epsilon$ of the value function, as shown in Figure 5.5.1b. Of course, ensuring that the bounds are tight is a always good idea, but our procedure for decision selection will especially benefit from it as it is more likely that $V_{t+1}^{*}$ will then exhibit a structure that is similar to $V_{t+1}^{*}$. Currently, however, we do not have a method that ensures that the bounds are close, and we will discuss this in more detail in section 6.2.

5.5.2 Convergence Not Guaranteed

Another potential cause for the bad performance of the Informed method is that the found decision rules are only secure against a subset of all policies. At runtime, only a finite amount of stochastic opponent policies is available, e.g., one policy $\pi_1$ for each path in the tree $T_2$ from root node to a leaf node. These are collected in the set $\Pi_{\sigma_{c,2}}$. To find a maxmin-strategy, we compute a decision rule $\delta^*_2$ that cannot be exploited, i.e., it is secure
Chapter 5. Exploiting the Identified Structure Shortcomings of the Informed Method

Figure 5.5.1: Example showing value vectors that describe a lower bound $V_{t+1}$, with above it the value function $V^*_{t+1}$ for a single conditional $\sigma^{t+1}_{m,2}$.

(a) Example in which the minimizing marginal is different for $V_{t+1}$ and $V^*_{t+1}$.

(b) The same example, except now we ensure that the bounds are $\epsilon$-close.

against all (pure) best-response policies. If we have no guarantee that $\Pi_{\sigma^{0}_{c,2}}$ will eventually contain all best-response policies, then we cannot guarantee that a strategy that is secure against the policies in $\Pi_{\sigma^{0}_{c,2}}$ is a maxmin-strategy. If we cannot guarantee that the found policy for agent 2 is rational, then we have no guarantee that the upper bound $\overline{V}_t$ converges to the value function $V^*_t$ either. Similar reasoning holds for the lower bound $\underline{V}_t$. 
Chapter 6

Further Analysis and Future Work

As discussed in the previous chapter, the ‘Informed’ method we proposed did not work well, partially because it does not guarantee that the upper and lower bounds converge to the value function. To investigate if the notion of solving the zero-sum POSG using heuristic search in the trees of conditionals is valid nonetheless and whether the fault lies with the heuristic or not, we repeat the experiments using a different heuristic that does guarantee convergence.

We explain this heuristic in more detail in section 6.1.1. In section 6.1.2, we discuss the experimental results. Based on these results and our insights, we give directions for future research in section 6.2.

6.1 Approach ‘Coevolution’

We propose to use an existing method called Nash Memory for asymmetric games (‘Nash memory’ for short) [Oliehoek, 2005] as heuristic for selection of decision rules. The main motivation for this choice is that this method guarantees convergence (i.e., a Nash Equilibrium is found) after a finite number of iterations, unlike the Informed method we proposed in the previous chapter.

Nash memory is based on the concept of coevolution, a field of research that has its roots in evolutionary algorithms. This class of algorithms iteratively generates a population of candidate solutions, in which each individual member is subjected to one or more arbitrary fitness tests. The fittest individuals then serve as the basis for the next generation of candidates. Coevolutionary algorithms differ from standard evolutionary algorithms in that they keep track of two populations: one population contains candidate solutions, the other contains ‘tests’, i.e., individuals that are used to determine the fitness of individuals from the first population [Ficici, 2004]. In the context of Nash Equilibria, a candidate solution can correspond to a policy.
6.1.1 Algorithm

The method *Nash Memory for asymmetric games* is an adaptation of the less general *Nash memory for symmetric games* [Ficici and Pollack, 2003]. It finds the maxmin-strategy and minmax-strategy, i.e., the policies that are secure against all opponent policies, by iteratively computing mixed policies using incrementally updated populations of pure policies. Let us define this formally. We restate the method as defined in [Oliehoek, 2005], and notation and terminology are largely similar.

Let \( N_i \) and \( M_i \) be two mutually exclusive sets of pure policies, defined for both agents \( i \in [1, 2] \). \( N_i \) is the support of the mixed policy \( \mu_{i,N_i} \), which is the policy that secures the highest payoff on the current iteration. \( M_i \) contains policies that are not in the support of \( \mu_{i,N_i} \). Let \( H \) be a heuristic that is used to find new policies in order to update \( N_i \) and \( M_i \). In our case, this heuristic will return best-response policies, as we will show.

At initialization, we let \( M_i = \emptyset \), and \( N_i \) is initialized as a set containing a single pure policy (chosen randomly). \( \mu_{i,N_i} \) is initialized as a mixture that that assigns probability 1 to the pure policy in \( N_i \). On every iteration, a set of test-policies \( \mathcal{L}_i \) is returned by the heuristic \( H \) and evaluated against \( \mu_{i,N_i} \). Assuming that the search heuristic delivers a single policy for both players, \( L_1 \) and \( L_2 \), we can check if the found mixed policies are secure by computing the expected payoff \( u \) for the compound policy \( L = (L_2, L_1) \) against the compound policy \( \mu_N = (\mu_{1,N_1}, \mu_{2,N_2}) \):

\[
u(L, \mu_N) \triangleq u_1((L_2, \mu_{2,N_2})) + u_2((L_1, \mu_{1,N_1})), \tag{6.1.1}\]

If \( u(L, \mu_N) > 0 \), then the found pair of mixed strategies is not secure against the found test policies. For both agents, we compute a new mixed policy for agent \( i \) as a mixture over policies in \( N_i \cup M_i \cup T_{-i} \) that is secure against all policies in \( N_{-i} \cup M_{-i} \cup T_i \). Of course, \( N_i \) and \( M_i \) are updated accordingly. The process is repeated until convergence, where \( u(T, (\mu_{1,N_1}, \mu_{2,N_2})) = 0 \). This is different from Nash memory for symmetric games, which calculates the best mixed policies from one set of encountered policies \( M \cup N \cup W \) [Ficici and Pollack, 2003].

Oliehoek et al. [2005] show that convergence is guaranteed if the two sets are updated using policies that are the best-response against the given mixed policy, as in the limit, the two populations will contain all pure policies — computing a mixture over all pure policies that is secure against all pure opponent policies gives the maxmin- or minmax-strategy [Thie and Keough, 2011]. More specifically, on every iteration \( T_2 \) will contain the pure policy \( \hat{\pi}_1 \) that is a best-response to \( \mu_{2,N_2} \), i.e., the pure policy that gives the highest payoff for agent 1 if agent 2 commits to playing \( \mu_{2,N_2} \). Similarly, \( T_1 \) contains the best-response

---

1 The set \( \mathcal{L}_i \) is denoted \( T_i \) in [Oliehoek, 2005], but we already use the notation \( T_i \) to refer to the trees of conditionals of agent \( i \).
policy \( \hat{\pi}_2 \) against \( \mu_1, \mathcal{N}_1 \). In the worst case, the number of iterations is linear in the number of pure policies [Ficici, 2004]. This worst case occurs when the security-level policies for both agents have all pure policies in their support.

We use \textit{Nash memory for asymmetric games} as a heuristic for decision rule selection. At every iteration of the coevolution scheme, we convert the found mixed security-level policies to stochastic policies\(^{2}\). The decision rules contained in this stochastic policy \( \pi_i = (\delta_i^0, \ldots, \delta_i^{h-1}) \) are added in the corresponding tree of conditionals, and nodes (corresponding to conditionals) are added where necessary.

The possible benefit of combining Nash memory with our search tree construction method is twofold. First, if Nash memory finds a stochastic policy \( \pi_1 = (\delta_1^0, \ldots, \delta_1^{h-1}) \) and we have already added the decision rule \( \delta_1^0 \) to the tree \( T_2 \), then we can simply follow the corresponding edge in the tree to the conditional \( \sigma_{c,2}^1 = U_c(\sigma_{c,2}^0, \delta_1^0) \), and we do not have to recompute the effect of following \( \delta_1^0 \). Second, the best-response needed by Nash memory is implicitly calculated in the value vector backup operation. We repeat the value vector definition below (for \( i = 1 \)).

\[
\nu_{[\sigma_{c,1}, \pi_2]}(\theta_1^t) \{4.2.22\} = \max_{a_1^t} \left[ \sum_{\theta_2^t} \sigma_{c,1}^t(\theta_2^t | \theta_1^t) \sum_{a_2^t} \delta_2^t(a_2^t | \theta_2^t) \left( R(\langle \theta_1^t, \theta_2^t \rangle, \langle a_1^t, a_2^t \rangle) + \sum_{o_2^{t+1}} \Pr(\langle o_2^{t+1} \rangle | \langle \theta_1^t, \theta_2^t \rangle, \langle a_1^t, a_2^t \rangle) \nu_{[\sigma_{c,1}, \pi_2]}(\theta_1^{t+1}, \theta_2^{t+1}) \nu_{[\sigma_{c,1}, \pi_2]}(\theta_1^{t+1}, \theta_2^{t+1})) \right) \right] (6.1.2)
\]

To find the value vector \( \nu_{[\sigma_{c,1}, \pi_2]} \), a maximization over actions \( a_1^t \) is performed for every AOH \( \theta_1^t \). By collecting these maximizing actions at every stage, we can construct a best-response policy \( \hat{\pi}_1 \) that specifies an action \( a_1^t \in \mathcal{A}_i \) for every AOH \( \theta_1^t \in \mathcal{\Theta}_i \).

The Expand (algorithm 5.2) and Backup (algorithm 5.3) procedures from the previous chapter are used, with some differences. We give the Expand procedure a stochastic policy of the opposing agent as input, from which it retrieves decision rules. For example, when expanding a node \( \sigma_{c,1}^t \) using a given stochastic policy \( \pi_2 \), the getNewDecisionRule call at the first line of the Expand procedure returns the decision rule \( \delta_2^t \) contained in \( \pi_2 \).

We let the Backup procedure collect the best-responses, so that we may construct a best-response policy. As we need a best-response at every stage of the game, the procedure can not be stopped if we find that the current value vector is dominated.

We emphasize that our Coevolution approach does not provide additional improvements over Nash memory: it follows the policies specified by Nash memory directly, and only performs best-response calculation and caching of (stochastic) policies. These steps could

\(^2\)Conversion from mixed to stochastic policies is performed using realization weights as described in [Oliehoek, 2005, Chapter 7], see also (2.3.5).
have been performed using different methods (alternatives for best-response calculation are discussed in more detail in [Oliehoek et al., 2005]).

The Coevolution procedure is given in algorithm 6.1. For both agents, a secure mixed policy \( \mu_i \) is converted to a stochastic policy \( \pi_i \). This stochastic policy serves as input for the Expand procedure. Using the resulting value vector, a best-response policy is calculated, which is used to find the next secure mixed policy.

Note, that we always expand the root node. Effectively, this is similar to setting \( \alpha = 0 \) in the Informed or Random method. We could have instead opted to traverse the partially constructed search trees first, and disregard part of the stochastic policy \( \pi_i \) specified by Nash memory (i.e., if we choose to expand node \( \sigma'_{c,t} \), we have already made decisions at stages \( 0, \ldots, t - 1 \), so we can disregard that part of \( \pi_i \)). However, the decisions at stages \( t, \ldots, h - 1 \) that are contained in \( \pi_i \) are only secure if agent \( i \) follows the policy at earlier stages as well. That is, a partial policy \( \pi_i^t \) that maximizes payoff for agent \( i \), if agent \( i \) follows \( \varphi_i^t \), does not necessarily maximize payoff if the agent follows different past policy \( \varphi_i'^t \) — and it likely results in lower payoff.

Of course, it may be that the found policy contains ‘good’ decision rules nonetheless, i.e., decision rules that give high payoff regardless of the decisions made in the past. However, we did not investigate this, and will instead provide discussion in section 6.2.

### 6.1.2 Experiments

We test the Coevolution approach on the competitive tiger and adversarial tiger domains. We compare its performance to the performance of our implementation of Nash memory for asymmetric games [Oliehoek, 2005], and the performance of sequence form solving [Koller et al., 1994]. The results are given in Table 6.1.1 and Table 6.1.2. The table gives the found upper bound \( \overline{V} \) and lower bound \( \underline{V} \). An ‘X’ indicates that no results could be find within the allotted time (10, 30, 60, 120, 240 minutes for the different horizons respectively, chosen empirically).

While the Coevolution approach is clearly superior to the methods proposed in section 5.2, it does not scale as well as sequence form representation does. Interestingly, our Coevolution approach is slightly faster than our implementation of Nash memory, indicating that our implicit best-response calculation (during the backup of value vectors) may be relatively fast way to find the best-response policy. Note, however, that the runtime of Nash memory could possibly be improved using faster best-response computation methods (as stated earlier, see Oliehoek et al. [2005]).

Nevertheless, for both Coevolution and Nash memory, the competitive tiger problem of horizon 5 and horizon 6 are not solvable. For the adversarial tiger problem, horizon 6 is unsolvable. The bottleneck in both approaches is the computation of the best-response policy, as this involves computing the value of every pair of AOHs and actions — clearly,
there is an exponential blowup. Our experimental results confirm this, indicating that the heuristic is not suited for large-horizon problems.

To empirically validate this, we let the Coevolution, Nash memory and sequence form methods run until convergence and measure the runtime for horizons 2 to 6. Results are shown in Table 6.1.3. The table shows the mean value of three runs. For sequence form solving, the time needed to construct the sequence form payoff matrix is included. As expected, runtimes for Coevolution and Nash memory are considerably higher for larger horizons than sequence form solving, and we observe a seemingly exponential trend.

We visualize the rate of convergence of the Coevolution method and Nash memory in Figure 6.1.2. Nash memory as specified in [Oliehoek et al., 2005] keeps track of the payoffs $u_1(⟨L_2, µ_2, N_2⟩)$ and $u_2(⟨L_2, µ_2, N_2⟩)$. To enable the comparison with the results of the Coevolution, Random and Informed methods, we have instead visualized the values $u_1(⟨µ_2, N_2, L_2⟩)$ and $u_1(⟨µ_1, N_1, L_1⟩)$. As expected, the bounds found by Coevolution and Nash memory converge faster than the Random method and the Informed method (shown in Figure 5.4.2), and Coevolution converges slightly faster than Nash memory. For the sake of completeness, we investigate the number of value vectors found by the Coevolution, Random and Informed methods. As it is likely that computing one value vector is already quite expensive, we would like to know how many value vectors are necessary in order to find the value. The median number of found value vectors at the first stage $t = 0$, based on five runs, is given in Table 6.1.4. We omitted the results for problems of horizon 6, as the Coevolution and Informed methods were not able to find any value vectors for these problems. Once again, results are based on five runs, with maximum runtime of 10, 30, 60 and 120 minutes for the horizons 2, 3, 4 and 5 respectively. As the Random and Informed methods do not converge within the allotted time, but Coevolution is able to converge in a finite number of iterations for most problems, the number of value vectors found by Coevolution is sometimes drastically lower than for the other methods. Obviously, random sampling of decision rules is faster than Informed selection, so the number of value vectors is typically much higher. Interestingly, the number of value vectors found by the Random method declines rapidly after horizon 4.

Given these experimental results, we believe that there are directions for future research that may be worth investigating. We will discuss these in more detail in section 6.2.
Table 6.1.1: Experimental results for the competitive tiger problem.

<table>
<thead>
<tr>
<th>Method</th>
<th>horizon</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Informed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coevolution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nash memory</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence form</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.1.2: Experimental results for the adversarial tiger problem.

<table>
<thead>
<tr>
<th>Method</th>
<th>horizon</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Informed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coevolution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nash memory</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence form</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.1.3: Runtime of the Coevolution, Nash memory and sequence form approaches in seconds, until convergence. The ‘X’ indicates that the program ran out of working memory or exceeded the time limit of 24 hours.
Chapter 6. Further Analysis and Future Work

Approach ‘Coevolution’

<table>
<thead>
<tr>
<th>Method</th>
<th>horizon 2</th>
<th>horizon 3</th>
<th>horizon 4</th>
<th>horizon 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\sigma^{0}_{c,1}$</td>
<td>$\sigma^{0}_{c,2}$</td>
<td>$\sigma^{0}_{c,1}$</td>
<td>$\sigma^{0}_{c,2}$</td>
</tr>
<tr>
<td>Random</td>
<td>1.19 · 10^5</td>
<td>1.43 · 10^5</td>
<td>8.87 · 10^4</td>
<td>8.73 · 10^4</td>
</tr>
<tr>
<td>Informed</td>
<td>3.24 · 10^3</td>
<td>3.12 · 10^3</td>
<td>31</td>
<td>34</td>
</tr>
<tr>
<td>Coevolution</td>
<td>6</td>
<td>6</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Random</td>
<td>3.32 · 10^5</td>
<td>3.51 · 10^5</td>
<td>6.70 · 10^5</td>
<td>8.21 · 10^5</td>
</tr>
<tr>
<td>Informed</td>
<td>1.92 · 10^4</td>
<td>3.23 · 10^4</td>
<td>5.26 · 10^3</td>
<td>7.43 · 10^3</td>
</tr>
<tr>
<td>Coevolution</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 6.1.4: Median number of value vectors at stage 0. Top three rows are results for the competitive tiger problem, bottom three rows for the adversarial tiger problem. Bold numbers indicate convergence.

Algorithm 6.1: SolvePOSGCoevolution: Coevolution as heuristic

**Input:** horizon $h$, maximum runtime

**Output:** Upper bound $\bar{V}$ and lower bound $\underline{V}$ on the value.

1. Initialize $\mathcal{M}_1, \mathcal{M}_2$ as $\emptyset$.
2. Initialize $\mathcal{N}_1, \mathcal{N}_2$ with a randomly selected pure policies $\tilde{\pi}_1, \tilde{\pi}_2$.
3. Initialize $\mu_1, \mu_2, \mathcal{N}_1, \mathcal{N}_2$ as mixtures over the single policy in $\mathcal{N}_1, \mathcal{N}_2$.
4. **while** time left **do**
5. **for** $i \in (1, 2)$ **do**
6. $\pi_i = \text{mixed2stochastic}(\mu_i)$ # using realization weights, see (2.3.5)
7. $\sigma^{h-1}_{c,-i}, v^{h-1}_{[\sigma^{h-1}_{c,-i}, \pi_i]} = \text{Expand}(\sigma^{0}_{c,-i}, h, \pi_i)$
8. $L_i = \text{Backup}(\sigma^{h-1}_{c,-i}, v^{h-1}_{[\sigma^{h-1}_{c,-i}, \pi_i]})$ # retrieve best-response $\hat{\pi}_{-i}$
9. **end for**
10. $L = \langle L_2, L_1 \rangle$ # construct compound policies
11. $\mu_{\mathcal{N}} = (\mu_1, \mathcal{N}_1, \mu_2, \mathcal{N}_2)$
12. **if** $u(L, \mu_{\mathcal{N}}) > 0$ **then** # convergence criterion, see (6.1.1)
13. **break**
14. **end if**
15. $\mathcal{N}_1 = \mathcal{N}_1 \cup \mathcal{M}_1 \cup \{L_2\}$
16. $\mathcal{N}_1 = \mathcal{N}_2 \cup \mathcal{M}_2 \cup \{L_1\}$
17. $\mu_{\mathcal{N}_1} = \text{findSecurePolicy}(\mathcal{N}_1, \mathcal{N}_2)$ # mixture over $\mathcal{N}_1$ secure against $\mathcal{N}_2$
18. $\mu_{\mathcal{N}_2} = \text{findSecurePolicy}(\mathcal{N}_2, \mathcal{N}_1)$
19. Update $\mathcal{N}_1, \mathcal{M}_1, \mathcal{N}_2, \mathcal{M}_2$
20. **end while**
21. $\bar{V} = \min(\sigma^{0}_{c,1}.\text{valueVectors})$
22. $\underline{V} = \max(\sigma^{0}_{c,2}.\text{valueVectors})$
23. **return** $(\bar{V}, \underline{V})$
Chapter 6. Further Analysis and Future Work

Figure 6.1.1: Typical convergence rate for the Coevolution method.

Figure 6.1.2: Typical convergence rate for the different methods, showing the lower bound $V_0$ and the upper bound $V$ as function of the elapsed time. The Nash Equilibrium is shown as a dotted line. It is 0 for the competitive tiger problem, and -3 for the adversarial tiger problem. Left figures show results for the Coevolution method, right for Nash memory. Top figures show results for the competitive tiger problem, bottom figures for the adversarial tiger problem.

6.2 Future Work

In the previous chapter, we attempted to validate the idea of designing algorithms that treat the zero-sum POSG as a series of problems — namely, the identification of a rational decision rule at each stage — rather than trying to solve a zero-sum POSG using a single LP. Motivation for this choice is that such approaches have already been used in the Dec-POMDP setting successfully [Emery-Montemerlo et al., 2004; MacDermed and Isbell, 2013]. We gave further empirical motivation for this idea in sections 6.1.2 and 6.1.2, where we showed that using sequence form representation to solve a zero-sum POSG becomes intractable for large horizon problems: $h > 5$ for the competitive tiger problem, $h > 6$ for the adversarial tiger problem. However, the ‘Informed’ method we proposed did not work well. Based on its shortcomings, we proposed to use an existing solution method as
heuristic, but observed that this new heuristic does not scale well. Based the experimental results, we discuss alternative heuristics and give directions for future research.

6.2.1 Exploiting Properties of Conditionals

It is not impossible for two sequences of decision rules to result in the same conditional. This means that a tree of conditionals is actually a Directed Acyclic Graph (DAG), i.e., a special type of tree in which different paths can lead to the same node. For example, if a conditional always assigns zero-probability to a specific AOH \( \sigma_{t+1}^c,1 | \vec{\theta}_{t+1}^2, \vec{\theta}_1^t \) = 0, \( \forall \vec{\theta}_1^t \in \tilde{\Theta}_1^t \), then decision rules that differ only in the assignment of probabilities \( \delta_{t+1}^2(a_{t+1}^2 | \vec{\theta}_2^t) \) will all result in the same conditional \( \sigma_{t+1}^c,1 \):

\[
\sigma_{c,1}^{t+1}(\vec{\theta}_{2}^{t+1}|\vec{\theta}_{1}^{t+1}) = \sigma_{c,1}(\vec{\theta}_{2}^{t}|\vec{\theta}_{1}^{t})\delta_{2}^{t}(a_{2}^{t}|\vec{\theta}_{2}^{t})Pr(o_{t+1}^{2}|a_{2}^{t},\vec{\theta}_{1}^{t+1}) = 0 \cdot \delta_{2}^{t}(a_{2}^{t}|\vec{\theta}_{2}^{t})Pr(o_{t+1}^{2}|a_{2}^{t},\vec{\theta}_{2}^{t},\vec{\theta}_{1}^{t+1}) = 0.
\]

We do not explicitly exploit the DAG structure in our algorithms. Given two arbitrary conditionals at consecutive stages, \( \sigma_{c,i}^t \) and \( \sigma_{c,i}^{t+1} \), it may be possible to infer the decision rule that satisfies the condition \( \sigma_{c,i}^{t+1} = U_c(\sigma_{c,i}^t, \delta_{c,i}^{t+1}) \). If such a decision rule exists and it is valid, then we can add an edge connecting the nodes \( \sigma_{c,i}^t \) and \( \sigma_{c,i}^{t+1} \). This allows value vectors at the node \( \sigma_{c,i}^{t+1} \) to be back-propagated to two different parent nodes, and therefore, through two different paths to the root node. This will (likely) result in different value vectors at stages \( t-1 \) to 0 as well. Essentially, such a method makes use of the notion that the rational partial policy \( \pi_{i}^{t+1} \) for two nodes \( \sigma_{c,i}^t \) and \( \sigma_{c,i}^{t+1} \) may be identical.

We believe that it may also be beneficial to determine if a small change in conditional-space implies a small change in policy-space, too. If so, approximately rational policies (and therefore, value vectors) for many similar conditionals can readily be found by computing the concave or convex value function at a single conditional. We have not been able to provide any results on how the value of similar conditionals might be related, but we find it unlikely that the relation between the value functions at two such conditionals is completely arbitrary.

6.2.2 Alternative Heuristics

We shortly discuss heuristics that we considered, but dismissed.

6.2.2.1 Bounds as Heuristics

For methods that iteratively construct upper and lower bounds, the distance between the upper and lower bounds seems like an obvious heuristic: a part of the statistic-space where the distance between the bounds is the largest, is the part that needs improvement. This
notion has successfully been applied in the POMDP setting [Smith and Simmons, 2004; Kocsis and Szepesvári, 2006]. If the searches in \( T_1 \) and \( T_2 \) are performed simultaneously, then we may be able to determine (at runtime) how tight the bounds on the value function are. However, it is difficult (if not impossible) to compute the distance between the upper and lower bound, as the continuous bounds are not defined in the same subspace of statistic-space: we only know continuous bounds for particular conditionals, and these bounds are only continuous in marginal-space.

Although it is possible to compute the distance at discrete statistics, such a procedure can only be performed for statistics that can be realized given the current set of conditionals. That is, there must be some \( \sigma_{c_1}^t \) and \( \sigma_{c_2}^t \) in the search tree so that \( \sigma^t = \sigma_{c_1}^t \sigma_{m_1}^t = \sigma_{c_2}^t \sigma_{m_2}^t \). Evaluation of newly sampled statistics (random or evenly spaced) should give some insight into the distance between the bounds, but it is likely that this underestimates the distance between the bounds. Unless we have some way of computing the bound for the entire statistic-space, it is difficult to give any guarantees about the quality of the found solution.

6.2.2.2 Adaptive \( \alpha \)

A shortcoming of the ‘Informed’ method is the parameter that guides the randomized depth-first search, \( \alpha \), is static. This parameter specifies whether the current node should be expanded by adding a new decision rule (with probability \( \alpha \)), or if we should instead move to one of its child nodes (with probability \( 1 - \alpha \)). In general, a reasonable \( \alpha \) will lie anywhere between 0 and 1.

Experimental results indicated that for the two domains we designed, results are fairly similar for all \( \alpha \geq 0.8 \). For other problems, however, it may be more beneficial to add nodes at the first stage \( t = 0 \) than it is to compute value vectors at the final stage. For example, when a partially constructed tree \( T_2 \) contains conditionals \( \sigma_{c_2}^t \) that will not be reached if agent 1 follows the rational decision rule \( \delta_0^1 \), then we would rather not traverse the tree, but instead add new decision rules \( \delta_1^0 \). Of course, whether a found decision rule is rational or not is generally not known until we have found the full rational policy, which makes it difficult to tell if a selected \( \alpha \) is ‘good’ or not.

Therefore, we argue that it may be beneficial to make \( \alpha \) adaptive. It can be seen as a ‘momentum’ parameter that specifies how often a node should be expanded, for example by specifying probability. At every node \( \sigma_{c,i}^t \), we decide whether we expand with probability \( 1 - \alpha_{\sigma_{c,i}^t} \), where \( 0 \leq \alpha_{\sigma_{c,i}^t} \leq 1 \). If we observe that expanding at node \( \sigma_{c,i}^t \) often results in dominated value vectors, we decrease the momentum for this node, as we would like to ensure that we do not expand again. Inversely, if we find that adding new decision rules at a node almost always results in improvements (i.e., dominating value vectors), we should increase \( \alpha \) for this node: we do not want to waste time and resources on the existing child
nodes of this node, and should keep adding new child nodes in the hope that this results in tighter bounds.

A disadvantage of this approach is that it is likely that introducing an adaptive $\alpha$ creates the need for several other hyperparameters that control the rate at which $\alpha$ is updated, among other settings. Fine-tuning these parameters is typically a difficult process. For example, let us assume that we are currently at the node $\sigma_{c1}^0$. If we find that adding new decision rules $\delta_2^0$ at this node consistently fails to improve the bound, then we should consider lowering the momentum at this node, $\alpha_{\sigma_{c1}^0}$. However, if our parameter settings specify that the momentum should be lowered slowly, it will take several iterations before the method figures out that it is useless to add new decision rules at this node. Inversely, if the badly selected decision rules were due to bad luck, and the momentum is lowered too aggressively, then our method may get stuck as it never tries to expand that node again.

Of course, we can further constrain the momentum parameter with threshold parameters $t_{\text{min}} < \alpha t_{\text{max}}$ (so that regardless of momentum updates, a node will be expanded with probability $t_{\text{min}}$, and we move to a next node the tree with probability $1 - t_{\text{max}}$), but this does not solve the problem entirely and introduces additional parameters.

### 6.2.2.3 Coevolution Combined with Tree-Traversal

Our Coevolution approach is largely based on the solution method *Nash memory for asymmetric games* [Oliehoek, 2005]. As Nash memory does not use the constructed bounds, only the best-response policy found during the value vector backup operation (as described in section 6.1.1), the Coevolution approach does not make a substantial improvement over the approach it was based on, and follows the policies specified by Nash memory exactly. Therefore, the constructed trees of conditionals only act as containers for the value vectors.

Similar to how the Random and Informed methods traverse the partially constructed search trees of conditionals, it may be useful let Coevolution traverse the tree as well, before deciding to expand a node. We can discard part of a policy found by Nash memory, $\pi_1 = \langle \delta_1^1, \ldots, \delta_t^{h-1} \rangle$, that is not needed. For example, if we have reached a node $\sigma_{c2}^t$ and we would like to expand it, we could select decision rules $\delta_1^1, \ldots, \delta_t^{h-1} \in \pi_1$, disregarding the past individual policy $\varphi_t^1 = \langle \delta_1^0, \ldots, \delta_t^{t-1} \rangle$. Effectively, however, this causes decision rules $\delta_1^t$ that are secure at a particular conditional $\sigma_{c2}^t$ to be added to a different conditional $\sigma_{c2}^{t'}$. We find it unlikely that a decision rule selected specifically for one conditional will still give high value at a different conditional. Of course, it may be the case that the found decision rule works well regardless of past decisions.
6.2.3 Outlook

We argue that there is a need for alternative solution methods for zero-sum POSGs, as the current state-of-the-art for such games — conversion to sequence form and solving the resulting problem using realization weights [Koller et al., 1994] — is not able to solve games of large horizon. In particular, we advocate the use of methods that solve the zero-sum POSG as a sequence of smaller problems, namely, the identification of rational decision rules at every stage. We showed that the concave and convex structure of the value function of the zero-sum POSG enables the design of methods that perform a heuristic search in conditional-space. The heuristics we propose in this work, however, do not work well. The first heuristic (‘Informed’, section 5.2.2) does not guarantee convergence, among other problems. The second heuristic (‘Coevolution’, section 6.1.1), based on an existing solution method, does not scale well. We argue that research towards alternative heuristics is needed, and emphasize that such heuristics should improve on at least one of the two aspects — a heuristic should guarantee convergence or be computationally cheap, and preferably both.
Chapter 7

Related Work

As far as the author is aware, this thesis is the first work that uses plan-time sufficient statistics as defined by Oliehoek [2013] in the zero-sum Partially Observable Stochastic Game setting — the exception being [Wiggers et al., 2015], which presents the findings of chapters 3 and 4. However, it is not the first to identify a statistic for zero-sum games of incomplete information, nor is it the first to identify structure in the value function of such games. As we will discuss in this chapter, the existing statistics are often less compact than the one we use, and the games under consideration are often less general than the zero-sum POSG. Furthermore, as the statistic we use originates in the Dec-POMDP literature, there are works from the collaborative setting that are closely related to ours.

We first discuss literature on a class of games called repeated zero-sum game with incomplete information, in section 7.1. This class of games is similar to the zero-sum POSG, but less general. For example, action selection in these games is defined to be non-simultaneous and agents observe the choices by the opponent directly. In section 7.2, we discuss works that identify a statistic that is similar to the one we use, for games where action selection is simultaneous.

We do not treat Nash memory for asymmetric games [Oliehoek et al., 2005] as it has already been discussed in detail in section 6.1.1.

7.1 Repeated Zero-Sum Games of Incomplete Information

There are a number of works from the game theory literature that present structural results on the value function of so-called ‘repeated zero-sum games with incomplete information’ [Mertens and Zamir, 1971; Ponssard, 1975; Ponssard and Sorin, 1980; Ponssard and Zamir, 1973]. These games are sequential, i.e., the agents act non-simultaneously, and have a static hidden state that is chosen at the start of the game. Based on this state (essentially a joint type, see section 2.1.2), each agent makes a private observation (i.e., individual type), and subsequently the agents take actions in turns thereby observing the actions of
the other player. This means that the incomplete information stems only from the static hidden state, and no new information arises during play (other than the actions of the opposing agent).

We disambiguate between incomplete games with information on one side [Ponssard and Zamir, 1973], cases with incomplete information on both sides where ‘observations are independent’ (i.e., where the distribution over joint types is a product of individual type distributions) [Ponssard, 1975] or dependent (general joint type distributions) [Mertens and Zamir, 1971; Ponssard and Sorin, 1980]. For these sorts of games, the value function has been shown to exhibit concave and convex properties in terms of the information distribution, similar to the structure we found for the zero-sum POSG.

The structural results, however, crucially depend on the alternating actions and the static state. This can be seen in (C.2.4), where we have repeated the main structural result of Ponssard and Zamir [1973]: the value function of repeated zero-sum games where both agents hold private information about the hidden state, expressed in terms of the information distribution, exhibits a concave and convex structure. The alternating maximizations and minimizations in (C.2.4) depend on the fact that the agents observe the actions of the opponent directly, i.e., every choice is based on the joint action-history of both agents and an individual observation at the start of the game. In the zero-sum POSG, this is not the case — generally, the agents receive no information about the actions of the opponents. Therefore, the found results do not extend to zero-sum POSGs directly. Nevertheless, the results by Ponssard et al. provide a good foundation for our theoretical results.

As the relevant literature is over 35 years old, we repeat parts of the proofs in notation that is familiar to us in appendix C. We refer an especially interested reader to [Ponssard and Zamir, 1973; Ponssard, 1975].

### 7.2 Multi-Agent Belief in Games with Simultaneous Actions

Recently, there has been increased interest in the identification of a multi-agent belief (i.e., a statistic for decision-making) in the collaborative Dec-POMDP setting that makes reasoning about these models more tractable. This is not surprising: the amount of practical applications of cooperative agents is endless, examples including rescue operations [Varakantham et al., 2009; Velagapudi et al., 2011] and industrial tasks [Carlin and Zilberstein, 2008]. As the Dec-POMDP model is similar to the zero-sum POSG, it is not unthinkable that solution methods for Dec-POMDPs that make use of plan-time statistics can be adapted for use in the zero-sum POSG.

However, in the Dec-POMDP setting, a probability distribution over observation histories and states is a better suited sufficient statistic for decision-making [Oliehoek,
Typically, this statistic is more compact than the distribution over AOHs that we use, and it does not depend on the initial belief $b^0$. However, we cannot use that statistic when we consider stochastic policies, as the observations do not provide sufficient information to determine past actions [Oliehoek, 2013]. Furthermore, whereas in the Dec-POMDP the statistics summarize a finite amount of past joint policies, there are infinitely many stochastic policies. As such, there will also be infinitely many plan-time sufficient statistics in a POSG, which renders the application of dynamic programming approaches that use the statistic impractical.

A recent paper that is similar to our work is by Nayyar et al. [2014] who introduce a so-called Common Information Based Conditional Belief — a probability distribution over AOHs and states, conditioned on common information — and use it to design a dynamic-programming approach for zero-sum POSGs. This method converts stages of the zero-sum POSG to Bayesian Games for which the type distribution corresponds to the statistic at that stage. However, since their proposed statistic is a distribution over joint AOHs and states, the statistic we use is more compact. Furthermore, Nayyar et al. do not provide any results regarding the structure of the value function, which is one of our main contributions.

Hansen et al. [2004] present a dynamic-programming approach for finite-horizon (general sum) POSGs that works by iteratively constructing sets of one-step-longer (pure) policies for all agents. At every iteration, the sets of individual policies are pruned by removing dominated policies. This pruning is based on a different statistic called multi-agent belief: a distribution over states and policies of other agents. Such a multi-agent belief is sufficient from the perspective of an individual agent to determine its best response (or to determine whether some of its policies are dominated). However, it is not a sufficient statistic for the past joint policy from a designer perspective, as it cannot be used to compute a joint strategy directly. The proposed plan-time sufficient statistic in this work, on the other hand, can be used to compute rational decisions at every stage of the game using (4.2.12) and (4.2.12).

A game-theoretic model that is closely related to the POSG model is the Interactive POMDP or I-POMDP [Gmytrasiewicz and Doshi, 2004]. In I-POMDPs, a (subjective) belief is constructed from the perspective of a single agent as a probability distribution over states and the types, $\zeta$, of all other agents. As the agents are rational, each individual AOH induces one type in the I-POMDP. Therefore, the belief of agent $i$ in the I-POMDP, which is a distribution $b(s, \zeta_j)$, can be seen to correspond to a conditional $\sigma^T_{c,i}(\tilde{\theta}_j | \tilde{\theta}_i)$ in the zero-sum POSG. Gmytrasiewicz and Doshi [2004] find that the value function expressed in this subjective belief-space is convex (which is similar to our structural results). However, knowing the value function from the perspective of one agent gives no guarantees about the value function from the perspectives of other agents. We have shown that in the zero-sum
case, the value function can be expressed in terms of the objective multi-agent belief that is shared by all agents. If we find the value function in marginal-space $\Delta(\theta^t_i)$ for a specific conditional $\sigma^t_{c,i}$ — corresponding to the perspective of one agent — then this does give guarantees about the value at particular (objective) statistics.

Hansen et al. [2011] show for various zero-sum stochastic games of infinite horizon that if the number of states is constant, an algorithm exists that can solve the game in polynomial time in the representation of the game. In these games, agents are able to observe the state directly, and the imperfect information stems only from the fact that the agents do not observe the choices of the opponent. As such, the games under consideration are less general than the zero-sum POSG, where agents also have uncertainty over the state.
Chapter 8

Conclusions

In this thesis we investigated several possibilities regarding the solving of two-player zero-sum Partially Observable Stochastic Games (POSG) of finite horizon using notions from literature on collaborative games. State-of-the-art solution methods for zero-sum POSG are able to find the rational joint strategy in polynomial time in the number of histories. However, as the number of histories is exponential in the number of actions and observations, solving such games quickly becomes infeasible. Clearly, there is a need for alternative solution methods. Motivated by this, we aimed to design methods that solve the zero-sum POSG as a sequence of smaller problems, namely, the identification of rational one-stage policies at every stage.

We provided new theory that enabled the design of such methods. We defined the value function of the zero-sum POSG in terms of so-called plan-time sufficient statistics (essentially distributions over concatenations of private information), and proved that it exhibits a particular structure at every stage of the zero-sum POSG: it is concave in marginal-space of the maximizing agent, i.e., the subspace of statistic-space that contains all probability distributions over private information of this agent, and convex in the marginal-space of the minimizing agent. Furthermore, we showed that the use of plan-time sufficient statistics allows for a reduction from the zero-sum POSG to a special type of Stochastic Game without observations which we call the Non-Observable Stochastic Game.

In an attempt to exploit the structural results, we proposed a heuristic method that performs a search in a subspace of statistic-space we refer to as conditional-space by selecting promising one-stage policies. We compared performance of this method to a random baseline and a state-of-the-art solution method that solves the zero-sum POSG using sequence form representation. We found that the proposed heuristic did not work well, and analyzed its shortcomings. To show that the underlying idea of performing a heuristic search in conditional-space is valid nonetheless, we repeated the experiments using a heuristic based on an existing solution method called Nash memory for asymmetric games. Results indicated that sequence form solving scales better than the heuristic methods we
Chapter 8. Conclusions

proposed. However, we observed that as the horizon grows, even sequence form solving quickly becomes infeasible.

Based on the experimental results, we argued for the use of heuristic search in the context of zero-sum POSGs, and proposed alternative heuristics. We believe that our theoretical results may serve as a first step towards the design of heuristic methods that solve the zero-sum POSG through identification of promising one-stage policies, and hope that such solution methods make reasoning in zero-sum games of incomplete information more tractable.
Appendix A

Proof of Lemmas Chapters 3 and 4

Proof of Lemma 3.2.1. We expand (3.1.3) in order to bring the marginal term to the front of the equation:

\[
Q_F(\sigma, \delta) \overset{\{3.1.3\}}{=} \sum_{\theta} \sigma(\theta) \sum_{a} \delta(a|\theta) R(\theta, a)
= \sum_{\theta_1} \sigma_{m,1}(\theta_1) \sum_{\theta_2} \sigma_{c,1}(\theta_2|\theta_1) \sum_{a_1} \delta_1(a_1|\theta_1) \sum_{a_2} \delta_2(a_2|\theta_2) R(\langle \theta_1, \theta_2 \rangle, \langle a_1, a_2 \rangle).
\]

\[\text{(A.1)}\]

A maximization over stochastic decision rules conditioned on the AOH \(\theta_1\) is equal to choosing a maximizing action for each of these AOHs. Thus, we can rewrite the best-response value function from (3.2.1) as follows:

\[
V_{F_{BR1}}(\sigma, \delta_2) = \max_{\delta_1} Q_F(\sigma, \langle \delta_1, \delta_2 \rangle)
\overset{\{A.2\}}{=} \max_{\delta_1} \left[ \sum_{\theta_1} \sigma_{m,1}(\theta_1) \sum_{\theta_2} \sigma_{c,1}(\theta_2|\theta_1) \sum_{a_1} \delta_1(a_1|\theta_1) \sum_{a_2} \delta_2(a_2|\theta_2) R(\langle \theta_1, \theta_2 \rangle, \langle a_1, a_2 \rangle) \right]
\]

\[\text{(A.2)}\]

As it is possible to write \(V_{F_{BR1}}\) as an inner product of the marginal distribution \(\sigma_{m,1}\) and a vector, \(V_{F_{BR1}}\) is linear in \(\Delta(\Theta_1)\) for all \(\sigma_{c,1}\) and \(\delta_2\). Analogously, \(V_{F_{BR2}}\) is linear in \(\Delta(\Theta_2)\) for all \(\sigma_{c,2}\) and \(\delta_1\). \(\square\)
Proof of Lemma 4.2.1. The proof is largely identical to the proof of correctness of sufficient statistics in the collaborative setting [Oliehoek, 2013]. For the final stage \( t = h - 1 \), it is easy to show that the sufficient statistic is sufficient:

\[
Q^*_{h-1}(\varphi_{h-1}, \bar{\theta}_{h-1}, \delta_{h-1}) = R(\bar{\theta}_{h-1}, \delta_{h-1}) = Q^*_h(\sigma_{h-1}, \bar{\theta}_{h-1}, \bar{\delta}_{h-1}).
\]

As induction hypothesis we assume that at stage \( t + 1 \), \( \sigma^{t+1} \) is a sufficient statistic, i.e.:

\[
Q^*_{t+1}(\varphi_{t+1}, \bar{\theta}_{t+1}, \delta_{t+1}) = Q^*_t(\sigma^{t+1}, \bar{\theta}_{t+1}, \delta_{t+1}). \tag{A.3}
\]

We aim to show that at stage \( t \), \( \sigma^t \) is a sufficient statistic as well:

\[
Q^*_t(\varphi^t, \bar{\theta}^t, \delta^t) = Q^*_t(\sigma^t, \bar{\theta}^t, \delta^t). \tag{A.4}
\]

We substitute the induction hypothesis into (4.2.3):

\[
Q^*_t(\varphi^t, \bar{\theta}^t, \delta^t) \overset{\{4.2.3\}}{=} R(\bar{\theta}^t, \delta^t) \sum_{\bar{\theta}^{t+1}} \sum_{\delta^{t+1}} \Pr(\bar{\theta}^{t+1}|\bar{\theta}^t, \delta^t) Q^*_t(\varphi^{t+1}, \bar{\theta}^{t+1}, \delta^{t+1})
\]

\[
\overset{\{A.3\}}{=} R(\bar{\theta}^t, \delta^t) \sum_{\bar{\theta}^{t+1}} \sum_{\delta^{t+1}} \Pr(\bar{\theta}^{t+1}|\bar{\theta}^t, \delta^t) Q^*_t(\sigma^{t+1}, \bar{\theta}^{t+1}, \delta^{t+1}) \tag{4.2.11}
\]

Furthermore, decision rules \( \delta_{1,pp}^{t+1} \) (based on the past joint policy) and \( \delta_{1,ss}^{t+1} \) (based on the sufficient statistic) are equal:

\[
\delta_{1,pp}^{t+1} \overset{\{4.2.5\}}{=} \arg\max_{\delta_{1}^{t+1}} \min_{\delta_{2}^{t+1}} \sum_{\bar{\theta}^{t+1}} \Pr(\bar{\theta}^{t+1}|\theta_0, \varphi^{t+1}) Q^*_t(\varphi^{t+1}, \bar{\theta}^{t+1}, \delta^{t+1})
\]

\[
\overset{\{\text{Def.4.2.2}\}}{=} \arg\max_{\delta_{1}^{t+1}} \min_{\delta_{2}^{t+1}} \sum_{\bar{\theta}^{t+1}} \sigma^{t+1}(\bar{\theta}^{t+1}) Q^*_t(\sigma^{t+1}, \bar{\theta}^{t+1}, \delta^{t+1}) \tag{4.2.12}
\]

Analogous reasoning holds for \( \delta_{2,ss}^{t+1} \). Therefore, by induction, \( \sigma^t \) is a sufficient statistic for \( \varphi^t, \forall t \in 0 \ldots h - 1 \).

Proof of Lemma 4.2.2. Given is that:

1. joint actions are equal to joint actions of the POSG,
2. joint types \( \theta \) correspond to joint AOHs \( \bar{\theta}^{h-1} \),
3. the initial distribution over joint types \( \sigma \) is equal to \( \sigma^{h-1} \).
At stage $t = h − 1$, the partial policy $\pi^{h − 1}$ contains only the joint decision rule $\delta^{h − 1}$. As such, we have:

\[
V_{h−1}(\sigma^{h−1}, \pi^{h−1}) = V_{h−1}(\sigma^{h−1}, \delta^{h−1})
\]

\[
\{4.2.16\} \quad Q^R_{t}^{(\sigma^{h−1}, \delta^{h−1})} \quad \{4.2.15\} \quad \sum_{\theta^{h−1}} \sigma^{h−1}(\theta^{h−1}) R(\theta^{h−1}, \delta^{h−1})
\]

\[
\{4.2.1\} \quad \sum_{\theta^{h−1}} \sigma^{h−1}(\theta^{h−1}) \sum_{a^{h−1}} \delta(a^{h−1}|\theta^{h−1}) R(\theta^{h−1}, a^{h−1}).
\]

By premises 1-3, this is equal to $Q_{\pi}$:

\[
Q_{\pi}(\sigma, \delta) \quad \{3.1.3\} \quad \sum_{\theta} \sigma(\theta) \sum_{a} \delta(a|\theta) R(\theta, a).
\]

As such, we have $Q_{\pi}(\sigma, \delta) = V_{h−1}(\sigma^{h−1}, \delta^{h−1}) = Q_{h−1}(\sigma^{h−1}, \delta^{h−1})$. From (3.1.3) and (4.2.17), it follows trivially that $V^*_{\pi}(\sigma) = V^*_{h−1}(\sigma^{h−1})$.

**Proof of Lemma 4.2.3.** We prove item 1 ($\sigma_{c,1}^{t+1}$ is not dependent on $\sigma_{m,1}^{t}, \delta_{1}^{t}$). We know that the statistic can be decomposed into marginal and conditional terms:

\[
\sigma^{t+1}(\theta^{t+1}) \quad \{4.2.8\} \quad \sigma^{t}(\theta^{t}) \delta^{t}(a^{t}|\theta^{t}) Pr(\sigma^{t}|a^{t}, \theta^{t})
\]

\[
= \sigma^{t}_{m,1}(\theta_{1}^{t}) \sigma^{t}_{c,1}(\theta_{2}^{t}, \delta_{1}^{t}) \delta_{2}^{t}(a_{1}^{t}|\theta_{1}^{t}) \delta_{2}^{t}(a_{2}^{t}|\theta_{2}^{t}) Pr(\langle a_{1}^{t}, a_{2}^{t} \rangle | \langle a_{1}^{t}, a_{2}^{t} \rangle, (\theta_{1}^{t}, \theta_{2}^{t}, \theta_{1}^{t}, \theta_{2}^{t})).
\]

Expanding the definition of $\sigma_{c,2}^{t+1}$ gives:

\[
\sigma^{t+1}_{c,1}(\theta_{2}^{t+1}|\theta_{1}^{t+1}) = \frac{\sigma^{t+1}(\langle \theta_{2}^{t+1}, \theta_{2}^{t+1} \rangle)}{\sigma^{t+1}_{m,1}(\theta_{1}^{t+1})} = \sum_{\theta_{2}^{t+1}} \sigma^{t+1}(\theta_{2}^{t+1}|\theta_{2}^{t+1})
\]

\[
= \sum_{\theta_{2}^{t+1}} \sigma^{t}_{m,1}(\theta_{1}^{t}) \sigma^{t}_{c,1}(\theta_{2}^{t}, \theta_{1}^{t}) \delta_{1}^{t}(a_{1}^{t}|\theta_{1}^{t}) \delta_{2}^{t}(a_{2}^{t}|\theta_{2}^{t}) Pr(\langle a_{1}^{t+1}, a_{2}^{t+1} \rangle | \langle a_{1}^{t+1}, a_{2}^{t+1} \rangle, (\theta_{1}^{t+1}, \theta_{2}^{t+1}))
\]

\[
= \sum_{\theta_{2}^{t+1}} \sigma^{t}_{m,1}(\theta_{2}^{t}) \delta_{1}^{t}(a_{1}^{t}|\theta_{1}^{t}) \sigma^{t}_{c,2}(\theta_{2}^{t}, \theta_{1}^{t}) \delta_{2}^{t}(a_{2}^{t}|\theta_{2}^{t}) Pr(\langle a_{1}^{t+1}, a_{2}^{t+1} \rangle | \langle a_{1}^{t+1}, a_{2}^{t+1} \rangle, (\theta_{1}^{t+1}, \theta_{2}^{t+1}))
\]

\[
= \sum_{\theta_{2}^{t+1}} \sigma^{t}_{c,1}(\theta_{1}^{t}) \delta_{1}^{t}(a_{1}^{t}|\theta_{1}^{t}) \delta_{2}^{t}(a_{2}^{t}|\theta_{2}^{t}) Pr(\langle a_{1}^{t+1}, a_{2}^{t+1} \rangle | \langle a_{1}^{t+1}, a_{2}^{t+1} \rangle, (\theta_{1}^{t+1}, \theta_{2}^{t+1}), (\theta_{1}^{t+1}, \theta_{2}^{t+1})).
\]

As $\delta_{1}^{t}$ and $\sigma_{m,1}^{t}$ are absent from (A.5), the conditional $\sigma_{c,1}^{t+1}$ is only dependent on the selected individual decision rule $\delta_{2}^{t}$ and the previous conditional $\sigma_{c,1}^{t}$. The proof for item 2 is analogous to the proof for item 1. 

\[82\]
Proof of Lemma 4.2.4. By the results of Lemma 3.2.1 and Lemma 4.2.2, we know the best-response value function \( V_{h-1}^{BR1} \) to be linear in \( \Delta(\Theta_1^{h-1}) \). For all other stages, we assume the following induction hypothesis:

\[
V_{t+1}^{BR1}(\sigma^{t+1}, \pi^{t+1}_2) = V_{t+1}^{BR1}(c^{t+1}_m, c^{t+1}_{c,1}, \pi^{t+1}_2) = \bar{\sigma}_{t+1}^{t+1} \cdot \nu[\sigma^{t+1}_c, \pi^{t+1}_2].
\] (A.6)

For the inductive step we aim to prove that at the current stage \( t \) the following holds:

\[
V_{t}^{BR1}(\sigma^{t}_m, \sigma^{t}_{c,1}, \pi^{t}_2) = \bar{\sigma}^t_{m,1} \cdot \nu[\sigma^{t}_c, \pi^{t}_2].
\] (A.7)

We expand the definition of \( V_{t}^{BR1} \). For notational convenience, we write \( \sigma^t \) instead of \( \sigma^{t}_m, \sigma^{t}_{c,1} \):

\[
V_{t}^{BR1}(\sigma^t, \pi^{t}_2) \overset{4.2.21}{=} \max_{\pi^t} V_{t}(\sigma^t, (\pi^{t}_1, \pi^{t}_2)) = \max_{\pi^t} \left[ Q_{t}^R(\sigma^t, (\delta^t_1, \delta^t_2)) + V_{t+1}(U_{ss}(\sigma^t, \delta^t), (\pi^{t+1}_1, \pi^{t+1}_2)) \right]
\overset{4.2.16}{=} \max_{\pi^t} \left[ Q_{t}^R(\sigma^t, (\delta^t_1, \delta^t_2)) + V_{t+1}(\sigma^{t+1}, (\pi^{t+1}_1, \pi^{t+1}_2)) \right]
= \max_{\delta^t_1} \left[ Q_{t}^R(\sigma^t, (\delta^t_1, \delta^t_2)) + \max_{\pi^t_1} \left[ V_{t+1}(\sigma^{t+1}, (\pi^{t+1}_1, \pi^{t+1}_2)) \right] \right]
\overset{4.2.21}{=} \max_{\delta^t_1} \left[ Q_{t}^R(\sigma^t, (\delta^t_1, \delta^t_2)) + V_{t+1}^{BR1}(\sigma^{t+1}, \pi^{t+1}_2) \right].
\] (A.8)

We make the decomposition of \( \sigma^t \) into the marginal and conditional terms explicit again. Immediate reward \( Q_{t}^R \) can be expanded similar to (A.2):

\[
Q_{t}^R(\sigma^{t}_{m,1}, \sigma^{t}_{c,1}, \delta^t_1, \delta^t_2) = \sigma^{t}_{m,1}(\tilde{\theta}^t_1)\sigma^{t}_{c,1}(\tilde{\theta}^t_2|\tilde{\theta}^t_1) \sum_{a^t_1} \delta^t_1(a^t_1|\tilde{\theta}^t_1) \sum_{a^t_2} \delta^t_2(a^t_2|\tilde{\theta}^t_2) R((\tilde{\theta}^t_1, \tilde{\theta}^t_2), (a^t_1, a^t_2)).
\] (A.9)

We expand \( V_{t+1}^{BR1} \) in order to bring the marginal distribution \( \sigma^{t+1}_m \) to the front:

\[
V_{t+1}^{BR1}(\sigma^{t+1}_m, \sigma^{t+1}_{c,1}, \pi^{t+1}_2) \overset{A.6}{=} \bar{\sigma}^{t+1}_{m,1} \cdot \nu[\sigma^{t+1}_c, \pi^{t+1}_2]
= \sum_{\tilde{\theta}^{t+1}_1} \bar{\sigma}^{t+1}_m(\tilde{\theta}^{t+1}_1) \nu[\sigma^{t+1}_c, \pi^{t+1}_2](\tilde{\theta}^{t+1}_1)
\overset{4.2.22}{=} \sum_{\tilde{\theta}^{t+1}_1} \sigma^{t+1}_m(\tilde{\theta}^{t+1}_1) \sum_{\tilde{\theta}^{t+1}_2} \sigma^{t+1}_{c,1}(\tilde{\theta}^{t+1}_2|\tilde{\theta}^{t+1}_1) \sum_{a^t_1} \delta^t_1(a^t_1|\tilde{\theta}^t_1) \sum_{a^t_2} \delta^t_2(a^t_2|\tilde{\theta}^t_2) \sum_{o^t_{a^t_1} a^t_{a^t_2}} \Pr((o^t_{a^t_1} a^t_{a^t_2})|\tilde{\theta}^t_1, \tilde{\theta}^t_2, (a^t_1, a^t_2)) \nu[\sigma^{t+1}_c, \pi^{t+1}_2]|(\tilde{\theta}^{t+1}_1).
\] (A.10)
Chapter A. Proof of Lemmas Chapters 3 and 4

Filling the expanded equations into (A.9) and factorizing gives:

\[ V_t^{BR1}(\sigma_{m,1}^t, \sigma_c^t, \pi_t) \quad \{A.21\} = \max_{\delta_t^1} \left[ Q_t R(\sigma_{m,1}^t, \sigma_c^t, \langle \delta_t^1, \delta_t^2 \rangle) + V_{t+1}^{BR1}(\sigma_{m,1}^{t+1}, \sigma_c^{t+1}, \pi_{t+1}) \right] \]

\[ \{A.10,A.12\} = \max_{\delta_t^1} \left[ \sum_{\theta_t^1} \sigma_{m,1}^t(\theta_t^1) \sum_{\theta_t^2} \sigma_{c,1}^t(\theta_t^2 | \theta_t^1) \sum_{a_t^1} \delta_t^1(a_t^1 | \theta_t^1) \sum_{a_t^2} \delta_t^2(a_t^2 | \theta_t^2) \right. \\
\quad \left. \left( R(\langle \theta_t^1, \theta_t^2 \rangle, \langle a_t^1, a_t^2 \rangle) + \sum_{o_t+1} \Pr(\langle o_t+1, a_t^1+1 \rangle | \langle \theta_t^1, \theta_t^2 \rangle, \langle a_t^1, a_t^2 \rangle) \mu_{[\sigma_{c,1}^{t+1}, \sigma_c^{t+1}]}(\theta_t^1+1) \right) \right] . \]

(A.13)

Note that the value vector at stage \( t + 1 \) is indexed by the conditional \( \sigma_{c,1}^{t+1} \). While this conditional is dependent on \( \delta_t^2 \), it is, by the result of Lemma 4.2.3, not dependent on \( \delta_t^1 \). Therefore we can remove the maximization over decision rules \( \delta_t^1 \) from the equation, by replacing it with a maximization over actions for each AOH. We rewrite (A.13) as follows:

\[ V_t^{BR1}(\sigma_{m,1}^t, \sigma_c^t, \pi_t) = \sum_{\theta_t^1} \sigma_{m,1}^t(\theta_t^1) \max_{a_t^1} \left[ \sum_{\theta_t^2} \sigma_{c,1}^t(\theta_t^2 | \theta_t^1) \sum_{a_t^2} \delta_t^2(a_t^2 | \theta_t^2) \right. \\
\quad \left. \left( R(\theta_t^1, a_t^1) + \sum_{o_t+1} \Pr(\langle o_t+1, a_t^1+1 \rangle | \theta_t^1, a_t^1) \mu_{[\sigma_{c,1}^{t+1}, \sigma_c^{t+1}]}(\theta_t^1+1) \right) \right] \]

\[ \quad \{4.2.22\} = \sum_{\theta_t^1} \sigma_{m,1}^t(\theta_t^1) \mu_{[\sigma_{c,1}^t, \pi_t]}(\theta_t^1) \{\text{vec. not.}\} = \sigma_{m,1}^t \cdot \nu_{[\sigma_{c,1}^t, \pi_t]} . \quad \text{(A.14)} \]

This corresponds to (A.7). Therefore, by induction, best-response value function \( V_t^{BR1} \) is linear in \( \Delta(\tilde{\Theta}_t^1) \) for a given \( \sigma_{c,1}^t \) and \( \pi_t^2 \), for all stages \( t = 0 \ldots h - 1 \). Analogously, \( V_t^{BR2} \) is a linear function in \( \Delta(\tilde{\Theta}_t^2) \) for a given \( \sigma_{c,2}^t \) and \( \pi_t^1 \), for all stages \( t = 0 \ldots h - 1 \). \( \square \)
Appendix B

Observation Functions and Reward Functions Test-Domains

B.1 Observation Functions

Table B.1: Observation matrix for the competitive tiger problem (continued on the next page). For the sake of concise notation, we write observation ‘left’ instead of ‘hear tiger left’, ‘right’ instead of ‘hear tiger right’.
### Chapter B. Observation Functions and Reward Functions Test-Domains Observation Functions

<table>
<thead>
<tr>
<th>$A_t$</th>
<th>$O_{t+1}$</th>
<th>$S_{t+1}$</th>
<th>$A_t$</th>
<th>$O_{t+1}$</th>
<th>$S_{t+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a_1$</td>
<td>$a_2$</td>
<td>$o_1$</td>
<td>$o_2$</td>
<td>$tiger$</td>
<td>$tiger$</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0.85</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0.15</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>block</td>
<td>block</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>left</td>
<td>0.85</td>
<td>0.15</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0.15</td>
<td>0.85</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>left</td>
<td>0.15</td>
<td>0.85</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>block</td>
<td>block</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>block</td>
<td>block</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>block</td>
<td>block</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>left</td>
<td>left</td>
<td>0</td>
<td>0</td>
<td>left</td>
<td>0</td>
</tr>
<tr>
<td>right</td>
<td>right</td>
<td>0</td>
<td>0</td>
<td>right</td>
<td>0</td>
</tr>
<tr>
<td>nothing</td>
<td>nothing</td>
<td>0</td>
<td>0</td>
<td>nothing</td>
<td>0</td>
</tr>
<tr>
<td>$A^t$</td>
<td>$O^{t+1}$</td>
<td>$S^{t+1}$</td>
<td>$A^t$</td>
<td>$O^{t+1}$</td>
<td>$S^{t+1}$</td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>-----------</td>
<td>-------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td></td>
<td>$a_1$</td>
<td>$a_2$</td>
<td>tiger</td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td>right</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>tiger</td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td>block</td>
<td>right</td>
<td>0 0</td>
<td>right</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>listen</td>
<td>right</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>right</td>
<td>0 0</td>
<td>right</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>open</td>
<td>right</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td>right</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>right</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>open</td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td>right</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>right</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>open</td>
<td>right</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td>right</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>right</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>open</td>
<td>right</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td>right</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>right</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>open</td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td>right</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>right</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>open</td>
<td>right</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td>right</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>right</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>left</td>
<td>0 0</td>
<td></td>
<td>left</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>nothing</td>
<td>0 0</td>
<td></td>
<td>nothing</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>
Table B.2: Observation matrix for the adversarial tiger problem. For the sake of concise notation, we write ‘same’ instead of ‘same door’ and ‘other’ instead of ‘other door’.

<table>
<thead>
<tr>
<th>$a_{\text{adventurer}}$</th>
<th>$a_{\text{tiger}}$</th>
<th>$O^{t+1}$</th>
<th>$S^{t+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>same</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>move</td>
<td>other</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>same</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>other</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>listen</td>
<td>same</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>other</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>same</td>
<td>0.6</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>other</td>
<td>0.4</td>
<td>0.6</td>
</tr>
<tr>
<td>move</td>
<td>same</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>other</td>
<td>0.6</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>same</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>other</td>
<td>0.4</td>
<td>0.6</td>
</tr>
<tr>
<td>listen</td>
<td>move</td>
<td>0.7</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>other</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>same</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>other</td>
<td>0.1</td>
<td>0.7</td>
</tr>
<tr>
<td>move</td>
<td>same</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>other</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>same</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>other</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>listen</td>
<td>move</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>other</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>same</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>other</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

88
B.2 Reward Functions

<table>
<thead>
<tr>
<th>( a_1 )</th>
<th>( a_2 )</th>
<th>left</th>
<th>right</th>
</tr>
</thead>
<tbody>
<tr>
<td>block</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>listen</td>
<td>-1</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>open left</td>
<td>-1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>open right</td>
<td>1</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( a_1 )</th>
<th>( a_2 )</th>
<th>left</th>
<th>right</th>
</tr>
</thead>
<tbody>
<tr>
<td>block</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>listen</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>open left</td>
<td>4</td>
<td>-2</td>
<td></td>
</tr>
<tr>
<td>open right</td>
<td>-2</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Table B.1: Reward matrix for the competitive tiger problem.

<table>
<thead>
<tr>
<th>( a_{\text{adventurer}} )</th>
<th>( a_{\text{tiger}} )</th>
<th>same</th>
<th>other</th>
</tr>
</thead>
<tbody>
<tr>
<td>listen</td>
<td>listen</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>move</td>
<td>move</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>open door</td>
<td>listen</td>
<td>-5</td>
<td>3</td>
</tr>
<tr>
<td>open door</td>
<td>move</td>
<td>-5</td>
<td>3</td>
</tr>
</tbody>
</table>

Table B.2: Reward matrix for the adversarial tiger problem.
Appendix C

Proofs Repeated Zero-Sum Games

Section C.1 treats zero-sum games with incomplete information on one side. Ponssard and Zamir [1973] have shown that the value function of this class of games exhibits a concave structure, and that it can be expressed in terms of the value function of a similar game. In section C.2, this result is extended to the zero-sum game where both agents hold private information, and neither of the agents knows the hidden state. The main result of Ponssard [1975] is that the value function of this class of games exhibits concave and convex properties in terms of the distribution over concatenations of private information.

C.1 Incomplete Information on One Side

Consider the problem of choosing an action in a one-shot game, where agent 1 has a set of types but agent 2 does not. This game is defined as follows.

Game C.1.1. Sequential game $\Gamma_{seq1}$ where only agent 1 has a type.

Definition:

- $I$ is the set of two agents,
- $A$ is the set of joint actions $a = (a_1, a_2)$,
- $\Theta_1$ is the set of types for agent 1,
- $\sigma \in \Delta(\theta_1)$ is a probability distribution over types of agent 1.
- $R : \Theta_1 \times A \to \mathbb{R}$ is the reward function mapping types and actions to payoff for agent 1.

Order of play:

1. ‘Nature’ chooses a type $\theta_1$ from $\Theta_1$ according to $\sigma$.
2. Agent 1 observes $\theta_1$, picks an action $a_1$ from $A_1$.
3. Agent 2 observes $a_1$ but not $\theta_1$, picks an action $a_2$ from $A_2$.
4. Both agents receive payoff according to their respective utility functions.

The Q-value function is defined as

$$Q_{seq1}(\sigma, (\delta_1, \delta_2)) \triangleq \sum_{\theta_1} \sigma(\theta_1) \sum_{a_1} \delta_1(a_1 | \theta_1) \sum_{a_2} \delta_2(a_2 | a_1) R(\theta_1, a_1, a_2).$$  \hspace{1cm} (C.1.1)
The value of the game for a given \( \sigma \) is then
\[
V_{seq1}(\sigma) \triangleq \min_{\delta_2} \max_{\delta_1} Q_{seq1}(\sigma, (\delta_1, \delta_2)).
\] (C.1.2)

Ponssard and Zamir [1973] show that \( V_{seq1} \) can be expressed in terms of the value function of a similar sequential game in which agent 1 does not use its private information. Let us define this game where agent 1 discards his information as \( \Gamma_{seq1}' \). The value function for this game is defined as a maximization over actions rather than decision rules, as agent 1 cannot give away his information through his actions and he cannot condition his decisions on observations:
\[
V_{seq1}'(\sigma) \triangleq \max_{a_1} \min_{a_2} \sum_{\theta_1} \sigma(\theta_1) R(\theta_1, a_1, a_2).
\] (C.1.3)

Let \( \text{Cav}_{\Delta(\Theta_1)} f(\sigma) \) be the minimal concavification of \( f(\sigma) \), i.e. the smallest concave function \( g(\sigma) \) on \( \Delta(\theta_1) \) that satisfies \( g(\sigma) \geq f(\sigma) \), \( \forall \sigma \in \Delta(\theta_1) \). We prove that the value function \( V_{seq1} \) can be expressed in terms of the value function \( V_{seq1}' \).

**Theorem C.1.2.** \( V_{seq1}(\sigma) = \text{Cav}_{\Delta(\Theta_1)} V_{seq1}')(\sigma) \)

**Proof** The proof consists of three steps. First, Ponssard et al. show that \( V_{seq1} \) is a concave function on \( \Delta(\Theta_1) \), which we show in Lemma C.1.3. These steps are largely identical to the steps we perform for the Family of Bayesian Games in chapters 3. Then, the equality \( V_{seq1}(\sigma) = \text{Cav}_{\Delta(\Theta_1)} V_{seq1}')(\sigma) \) is proved by splitting it into two inequalities which are shown to be true independently, in Lemma C.1.4 and Lemma C.1.5 respectively. We find that \( V_{seq1}(\sigma) = \text{Cav}_{\Delta(\Theta_1)} V_{seq1}')(\sigma) \).

Ponssard et al. go on to derive the rational decision rules for both agents, and extend the proof for the one-shot game \( \Gamma_{seq1} \) to the multi-stage setting. It turns out that for any point \( \sigma \in \Delta(\Theta_1) \), the marginal probability of choosing a particular action \( \Pr(a_1) = \sum_{\theta_1} \sigma(\theta_1) \delta_1^a(a_1|\theta_1) \) is directly related to the distance from \( \sigma \) to statistics that describe the concave function \( V_{seq1} \), as we show in Figure C.1.2. It is useful to know that agent 1 can act in such a way that the multi-agent belief \( \sigma \) changes in a useful way.

Interestingly, the value of a repeated zero-sum game of horizon \( h \) is found to be the following:
\[
V_{seq1}^h(\sigma) \triangleq h \cdot V_{seq1}(\sigma), \forall h \in \{1, \ldots, \infty\}.
\] (C.1.4)

This implies that a rational strategy for agent 1 ensures that he never gives away information about his type, otherwise the attained value at a following stage would be lower (of course, it also implies that he is not able to increase the uncertainty of his opponent through his actions, but this is trivially true). Intuitively it may seem more important to maximize immediate reward, but information is arguably more valuable in these games. For example,
if agent 2 learns the type of agent 1 through the chosen actions $a_1$ then agent 2 may be able to determine the hidden state of the game, making it easier for agent 2 to exploit agent 1. This is an important notion, as it validates the idea that the value function of a zero-sum game should be concave and convex in terms of the information distributions of the respective agents. We refer the reader to [Ponssard and Zamir, 1973] for the full proof.

**Lemma C.1.3.** $V_{seq1}(\sigma)$ is a concave function on $\Delta(\Theta_1)$.

**Proof** Examine the expected payoff $V_{seq1}(\sigma, \delta_1, \delta_2)$ given $\sigma$ and stochastic policies for both agents, $\delta_1$ and $\delta_2$ given in equation (C.1.1). Using the fact that for a given $\delta_2$, there is deterministic decision rule $\delta_1$ that is a best-response, we substitute the maximization over $\delta_1$ by a maximization over actions $a_1$, and $\delta_1(a_1|\theta) = 1$ for these best-response actions:

$$V_{seq1}(\sigma) = \min_{\delta_2} \sum_{\theta_1} \sigma(\theta_1) \max_{a_1} \sum_{a_2} \delta_2(a_2|a_1)R(\theta_1, \langle a_1, a_2 \rangle).$$

We show that this is a minimization over linear functions in $\Delta(\Theta_1)$. Let $f_{\delta_2}$ be defined as follows:

$$f_{\delta_2}(\theta_1) = \sum_{a_2} \max_{a_1} \delta_2(a_2|a_1)R(\theta_1, \langle a_1, a_2 \rangle).$$

We can then rewrite $V_{seq1}$ as:

$$V_{seq1}(\sigma) = \min_{\delta_2} \sum_{\theta_1} \sigma(\theta_1)f_{\delta_2}(\theta_1).$$

This shows that $V_{seq1}(\sigma)$ is the minimization of a set of linear functions in $\Delta(\Theta_1)$. Therefore, it is a concave function in $\Delta(\Theta_1)$.

**Lemma C.1.4.** $V_{seq1}(\sigma) \geq Cav_{\Delta(\Theta_1)} V_{seq1'}(\sigma)$

**Proof** Note that the difference between $V_{seq1}$ and $V_{seq1'}$ is that in the calculation of the latter agent 1 is not informed of $\theta_1$. The situation for agent 2 is unaltered. Thus, it makes
sense intuitively that the situation where information is withheld from agent 1 results in equal or lower payoff for agent 1.

Note that in $V_{seq1}'$, agent 1 can choose from a limited set of action. To prove this formally, let $R^δ2(θ_1, a_1) = \sum a_2 δ^2(a_2 \mid a_1)R(θ_1, a_1, a_2)$ be defined $\forall δ_2$ as the expected payoff for agent 1 given action and type of agent 1. Replacing the maximization over decision in the value function definition by a maximization over actions for every $θ_1$ gives us the following:

$$V_{seq1}(σ) = \min_{δ_2} \max_{θ_1} \sum_{a_1} σ(θ_1) \sum_{a_1} δ^1_1(a_1 \mid θ_1) \sum_{a_2} δ^2_2(a_2 \mid a_1)R(θ_1, a_1, a_2)$$

Choosing the maximizing action for every $θ_1$ gives equal or higher payoff for agent 1 than selecting a single action for all $θ_1$ based on the distribution $σ$ alone:

$$\sum_{θ_1} σ(θ_1) \max_{a_1} R^δ2(θ_1, a_1) \geq \max_{a_1} \sum_{θ_1} σ(θ_1)R^δ2(θ_1, a_1).$$

Therefore, we get:

$$V_{seq1}(σ) \geq V_{seq1}'(σ).$$

As we already know that $V_{seq1}$ is concave in $Δ(Θ_1)$, we get:

$$V_{seq1}(σ) \geq V_{seq1}'(σ),$$

$$V_{seq1}(σ) \text{ is concave in } Δ(Θ_1),$$

$$V_{seq1}(σ) \geq Cav_{Δ(Θ_1)} V_{seq1}'(σ).$$

**Lemma C.1.5.** $V_{seq1}(σ) \leq Cav_{Δ(Θ_1)} V_{seq1}'(σ)$

**Proof** Let $π_1$ be a mapping of actions $a_1 \in A_1$ to marginal probabilities $Pr(a_1)$:

$$π_1(a_1) = \sum_{θ_1} σ(θ_1)π(a_1 \mid θ_1) \quad \text{(C.1.5)}$$
Let $\sigma_{a_1}$ be a mapping from types $\theta_1$ to posterior probabilities $\Pr(\theta_1|a_1)$ for a given $a_1, \delta_1$:

$$\sigma_{a_1}(\theta_1) = \begin{cases} \frac{\pi(\theta_1|a_1|\theta_1)}{\pi_i(a_1)} & \text{if } \pi_i(a_1) \neq 0, \\ \text{arbitrary} & \text{otherwise.} \end{cases} \quad (C.1.6)$$

Rewriting (C.1.1) using (C.1.5) and (C.1.6) gives

$$Q_{seq1}(\sigma, \delta_1, \delta_2) = \sum_{\theta_1} \sum_{a_1} \sum_{a_2} \sigma(\theta_1)\delta_1(\theta_1|\theta_1)\delta_2(a_2|a_1)R(\theta_1, a_1, a_2)$$

$$= \sum_{\theta_1} \sum_{a_1} \sum_{a_2} \sigma_{a_1}(\theta_1)\pi_i(a_1)\delta_2(a_2|a_1)R(\theta_1, a_1, a_2)$$

$$= \sum_{a_1} \pi_i(a_1) \sum_{a_2} \delta_2(a_2|a_1) \sum_{\theta_1} \sigma_{a_1}(\theta_1)R(\theta_1, a_1, a_2).$$

By the minimax-theorem the value function for the game $\Gamma_{seq1}$ is now:

$$V_{seq1}(\sigma) = \max_{\delta_1} \min_{\delta_2} V_{seq1}(\sigma, \delta_1, \delta_2)$$

$$= \max_{\delta_1} \min_{\delta_2} \sum_{a_1} \pi_i(a_1) \sum_{a_2} \delta_2(a_2|a_1) \sum_{\theta_1} \sigma_{a_1}(\theta_1)R(\theta_1, a_1, a_2)$$

$$= \max_{\delta_1} \sum_{a_1} \pi_i(a_1) \min_{a_2} \sum_{\theta_1} \sigma_{a_1}(\theta_1)R(\theta_1, a_1, a_2).$$

It is obvious that picking a maximizing action $a_1'$ for each $\theta_1$ instead of following decision rule $\delta_1$ (contained in $\bar{\delta}_1$) will result in greater or equal payoff:

$$V_{seq1}(\sigma) = \max_{\delta_1} \sum_{a_1} \bar{\delta}_1(a_1) \min_{a_2} \sum_{\theta_1} \sigma_{a_1}(\theta_1)R(\theta_1, a_1, a_2)$$

$$\leq \max_{\delta_1} \sum_{a_1} \bar{\delta}_1(a_1) \max_{a_1'} \min_{a_2} \sum_{\theta_1} \sigma_{a_1}(\theta_1)R(\theta_1, a_1', a_2). \quad (C.1.8)$$

Using the fact that $\sigma_{a_1}(\theta_1)$ is independent of $a_1'$, it is possible to rewrite (C.1.8) using the definition of $V_{seq1'}(\sigma)$ in (C.1.3):

$$V_{seq1}(\sigma) \leq \max_{\delta_1} \sum_{a_1} \bar{\delta}_1(a_1) \max_{a_1'} \min_{a_2} \sum_{\theta_1} \sigma_{a_1}(\theta_1)R(\theta_1, a_1', a_2)$$

$$= \max_{\delta_1} \sum_{a_1} \bar{\delta}_1(a_1)V_{seq1'}(\sigma_{a_1}).$$

We know that $\sigma_{a_1} \in \Delta(\Theta_1)$. Therefore, $\sum_{a_1} \pi_i(a_1)V_{seq1}(\sigma_{a_1}) \leq \text{Cav } V_{seq1'}(\sigma)$: the left hand side of the equation is a linear combination of points in $V_{seq1'}(\sigma)$ and can never exceed values in $V_{seq1'}(\sigma)$. Then $V_{seq1}(\sigma) \leq \text{Cav } V_{seq1'}(\sigma)$ holds as well. \[\square\]
C.2 Incomplete Information on Both Sides

The structural results found for repeated zero-sum games where only one agent has private information are extended to a game where both agents have types in [Ponssard, 1975]. Similar to the game in section C.1, the agents are to solve the problem of acting in the game without revealing their type to the opponent, with the difference that here, both agents receive private information about the hidden state of the game. As a consequence, both agents have uncertainty about the hidden state. Let us first consider the one-shot version of the repeated zero-sum game. It is defined as follows.

Game C.2.1. $\Gamma_{\text{seq2}}$: Sequential game where both agents hold private information.

Definition:

- $I$ is the set of two agents,
- $\mathcal{A}$ is the set of joint actions $a = (a_1, a_2)$,
- $\Theta$ is the set of joint types $\theta = (\theta_1, \theta_2)$,
- $\sigma \in \Delta(\theta_1)$ is a probability distribution over types of agent 1.
- $R : \Theta \times A \rightarrow \mathbb{R}$ is the reward function mapping types and actions to payoff for agent 1.

Order of play:

1. ‘Nature’ chooses a joint type $\theta$ according to $\sigma$.
2. Agent 1 observes $\theta_1$, picks an action $a_1$ from $A_1$.
3. Agent 2 observes $\theta_2, a_1$ but not $\theta_1$, picks an action $a_2$ from $A_2$.
4. Both agents receive payoff according to their respective utility functions.

The probability distribution over joint types can be decomposed into a marginal distribution $\sigma_m$, and conditional distribution $\sigma_c$, just as we did in chapter 3. The value function is then defined as:

$$Q_{\text{seq2}}(\sigma, (\delta_1, \delta_2)) \triangleq \sum_{\theta_1} \sigma(\theta_1) \sum_{\theta_2} \sigma(\theta_2 | \theta_1) \sum_{a_1} \delta_1(a_1 | \theta_1) \sum_{a_2} \delta_2(a_2 | a_1, \theta_2) R(\theta_1, \theta_2, a_1, a_2)$$

$$V_{\text{seq2}}(\sigma) \triangleq \max_{\delta_1} \min_{\delta_2} Q_{\text{seq2}}(\sigma, (\delta_1, \delta_2)).$$

(C.2.1)

Ponssard et al. prove that this value function, like that of the game $\Gamma_{\text{seq1}}$ from section C.1, can be expressed in term of the value function of a similar game. Let $\text{Cav}_f(\sigma)$ be defined as the minimal concavification be the minimal concavification of $f(\sigma)$ over $\Delta(\Theta_1)$, i.e. the smallest concave function $g(\sigma)$ on $\Delta(\Theta_1)$ that satisfies $g(\sigma) \geq f(\sigma), \forall \sigma \in \Delta(\Theta_1)$. Similarly, let $\text{Vex}_f(\sigma)$ be the maximal convexification of $f(\sigma)$ over $\Delta(\Theta_2)$ for a given $\sigma$, i.e. the largest convex function $g(\sigma)$ on $\Delta(\Theta_2)$ that satisfies $g(\sigma) \leq f(\sigma), \forall \sigma \in \Delta(\Theta_2)$, respectively. Let us now define a new game in which agent 1’s choice is limited to a single action:
Chapter C. Proofs Repeated Zero-Sum Games  Incomplete Information on Both Sides

**Game C.2.2.** Sequential game where agent 1’s choice is limited to a single action $\Gamma_{\text{seq2}}^{a_1}$.

**Definition:**
- $I$ is the set of two agents,
- $a_1$ is the given action for agent 1,
- $A_2$ is the set of actions for agent 2,
- $\Theta$ is the set of joint types $\theta = (\theta_1, \theta_2)$,
- $\sigma \in \Delta(\Theta)$ is a probability distribution over types,
- $R : \Theta \times A \rightarrow \mathbb{R}$ is the reward function mapping types and actions to payoff for agent 1.

Its value function is defined as

$$V_{\text{seq2}}^{a_1}(\sigma) \triangleq \min_{\delta_2} \sum_{\theta_2} \sigma(\theta_2) \sum_{\theta_1} \sigma(\theta_1 | \theta_2) \sum_{a_2} \delta_2(a_2 | a_1, \theta_2) R(\theta_1, \theta_2, a_1, a_2).$$  \hfill (C.2.2)

Given this definition, Ponssard et al. prove the following Lemmas, which lead to a generalization of Theorem C.1.2.

**Lemma C.2.3.** $V_{\text{seq2}}(\sigma) = \text{Cav} \max_{\Delta(\Theta_1) a_1 \in A_1} V_{\text{seq2}}^{a_1}(\sigma)$

**Proof** First, Ponssard et al. show that $V_{\text{seq2}}$ is a concave function on $\Delta(\Theta_1)$, which we show in Lemma C.2.6. The proof is similar to the proof of concavity for the repeated zero-sum game where agent 2 has no types, from Lemma C.1.3. Then, the equality is proved by splitting it into two inequalities which are shown to be true independently, in Lemma C.2.7 and Lemma C.2.7 respectively. We find that $V_{\text{seq2}}(\sigma) = \text{Cav} \max_{\Delta(\Theta_1) a_1 \in A_1} V_{\text{seq2}}^{a_1}(\sigma)$. $\blacksquare$

**Lemma C.2.4.** $V_{\text{seq2}}^{a_1}(\sigma) = \text{Vex} \min_{\Delta(\Theta_2) a_2 \in A_1} \sum_{\theta_1} \sigma(\theta_1) \sum_{\theta_2} \sigma(\theta_2 | \theta_1) R(\theta_1, \theta_2, a_1, a_2)$

**Proof** First, Ponssard et al. show that $V_{\text{seq2}}$ is a concave function on $\Delta(\Theta_1)$, which we show in Lemma C.2.9. Then, the equality is proved by splitting it into two inequalities which are shown to be true independently, in Lemma C.2.10 and Lemma C.2.11 respectively. We find that $V_{\text{seq2}}^{a_1}(\sigma) = \text{Vex} \min_{\Delta(\Theta_2) a_2 \in A_1} \sum_{\theta_1} \sigma(\theta_1) \sum_{\theta_2} \sigma(\theta_2 | \theta_1) R(\theta_1, \theta_2, a_1, a_2)$. $\blacksquare$

**Theorem C.2.5.** $\text{Cav} \max_{\Delta(\Theta_1) a_1 \in A_1} \text{Vex} \min_{\Delta(\Theta_2) a_2 \in A_1} \sum_{\theta_1} \sigma(\theta_1) \sum_{\theta_2} \sigma(\theta_2 | \theta_1) R((\theta_1, \theta_2), (a_1, a_2))$

**Proof** We insert the results of Lemma C.2.3 into the result of Lemma C.2.4:

$$V_{\text{seq2}}(\sigma) = \text{Cav} \max_{\Delta(\Theta_1) a_1 \in A_1} \text{Vex} \min_{\Delta(\Theta_2) a_2 \in A_1} \sum_{\theta_1} \sigma(\theta_1) \sum_{\theta_2} \sigma(\theta_2 | \theta_1) R((\theta_1, \theta_2), (a_1, a_2)).$$  \hfill (C.2.3)
Ponssard [1975] show that iterative application of this result leads to the following value function definition for the horizon $h$ repeated zero-sum version of $\Gamma_{\text{seq}2}$:

$$V^h_{\text{seq}2}(\sigma) = \text{Cav} \max_{\Delta(\Theta_1)} R_{\text{ex}} \min_{\Delta(\Theta_2)} \cdots \text{Cav} \max_{\Delta(\Theta_1)} R_{\text{ex}} \min_{\Delta(\Theta_2)} \sum_{\theta_1} \sigma(\theta_1) \sum_{\theta_2} \sigma(\theta_2 | \theta_1) \sum_{a_1} \delta_1(a_1', \theta_1) \sum_{a_2} \delta_2(a_2 | a_1, \theta_2) R(\theta_1, \theta_2, a_1, a_2).$$

(C.2.4)

This is a generalization of the result of Theorem C.1.2. The main result here is that the value function is concave and convex in different subspaces of statistic-space at every stage of the game. This result is intuitive: as we move towards the edges of the space $\Delta(\Theta_1)$, agent 2 becomes more certain of the type of agent 1, and is therefore able to exploit agent 1 better, leading to a decrease in value. Similarly, if the uncertainty of agent 1 about the type of agent 2 decreases, agent 1 will be able to exploit agent 2 better, leading to an increase in value. In the repeated zero-sum game the space $\Delta(\Theta)$ does not change over time, unlike in the zero-sum POSG case, where the statistic-space at $t + 1$ is of higher dimensionality than the statistic-space at stage $t$. Therefore, the extension to the zero-sum POSG case is not straightforward. However, as we have shown in chapter 4, it is possible to prove the existence of similar properties at very stage, independently.

**Lemma C.2.6.** $V_{\text{seq}2}(\sigma)$ is concave over $\Delta(\Theta_1)$.

**Proof** This is a generalisation of Lemma C.1.3. We replace the maximization over decision rules $\delta_1$ by a maximization over actions for every $\theta_1$:

$$V_{\text{seq}2}(\sigma) = \min_{\delta_2} \max_{\delta_1} \sum_{\theta_1} \sigma(\theta_1) \sum_{\theta_2} \sigma(\theta_2 | \theta_1) \sum_{a_1} \delta_1(a_1', \theta_1) \sum_{a_2} \delta_2(a_2 | a_1, \theta_2) R(\theta_1, \theta_2, a_1, a_2).$$

(C.2.5)

Let $f_{\delta_2}$ be a function defined on $\Theta_1$ for all $\sigma_{c, \delta_2}$ as

$$f_{[\sigma_{c, \delta_2}]}(\theta_1) = \sum_{\theta_2} \sigma(\theta_1) \max_{a_2} \sum_{a_1} \sigma(\theta_2 | \theta_1) \sum_{a_2} \delta_2(a_2 | a_1, \theta_2) R(\theta_1, \theta_2, a_1, a_2).$$

Let $f_{\delta_1}$ be defined similarly. Substituting $f_{\delta_2}(\sigma)$ in the definition of the value function given in (C.2.5) gives

$$V_{\text{seq}2}(\sigma) = \min_{\delta_2} \sum_{\theta_1} \sigma(\theta_1) \max_{a_1} \sum_{\theta_2} \sigma(\theta_2 | \theta_1) \sum_{a_2} \delta_2(a_2 | a_1, \theta_2) R(\theta_1, \theta_2, a_1, a_2)$$

$$= \min_{\delta_2} \sum_{\theta_1} \sigma(\theta_1) f_{[\sigma_{\delta_1, \delta_2}]}(\theta_1)$$
This shows that $V_{\text{seq2}}(\sigma)$ is a minimization over linear functions on $\Delta(\Theta_1)$, one for each policy $\delta_2$. Therefore, $V_{\text{seq2}}(\sigma)$ must be concave over $\Delta(\Theta_1)$. \qed

**Lemma C.2.7.** $V_{\text{seq2}}(\sigma) \geq \text{Cav} \max_{\Delta(\Theta_1)} V^{a_1}(\sigma) \quad \forall \sigma \in \Delta(\Theta_1) \times \Delta(\Theta_2)$

**Proof** Given that the set of choices for the maximizing agent 1 in $\Gamma_{\text{seq2}}^{a_1}$ is a subset of its set of choices in $\Gamma_{\text{seq2}}$, it follows that

$$V_{\text{seq2}}(\sigma) \geq \max_{a_1 \in A_1} V^{a_1}(\sigma) \quad \forall \sigma \in \Delta(\Theta_1) \times \Delta(\Theta_2)$$

Combining this with the fact that $V_{\text{seq2}}(\sigma)$ is a concave function on $\Delta(\Theta_1)$, we know that for all $\sigma \in \Delta(\Theta_1) \times \Delta(\Theta_2)$:

$$V_{\text{seq2}}(\sigma) \geq \max_{a_1 \in A_1} V^{a_1}(\sigma),$$

and therefore

$$V_{\text{seq2}}(\sigma) \geq \text{Cav} \max_{\Delta(\Theta_1)} V^{a_1}(\sigma).$$ \qed

**Lemma C.2.8.** $V_{\text{seq2}}(\sigma) \leq \text{Cav} \max_{\Delta(\Theta_1)} V^{a_1}(\sigma) \quad \forall \sigma \in \Delta(\Theta)$

**Proof** Let $\delta_1$ be a mapping of actions $a_1 \in A_1$ to marginal probabilities $\Pr(a_1)$:

$$\delta_1(a_1) = \sigma(\theta_1) \delta(a_1 | \theta_1). \quad (C.2.6)$$

Let $\sigma_{a_1}$ be a mapping from types $\theta_1$ to posterior probabilities $\Pr(\theta_1 | a_1)$ for given $a_1$, $\delta_1$.

$$\sigma_{a_1}(\theta_1) = \begin{cases} \frac{\sigma(\theta_1) \delta_1(a_1 | \theta_1)}{\delta_1(a_1)} & \text{if } \delta_1(a_1) \neq 0, \\ \text{arbitrary} & \text{otherwise.} \end{cases} \quad (C.2.7)$$

Substituting (C.2.6) and (C.2.7) into (C.2.1) give:

$$V_{\text{seq2}}(\sigma) = \max_{\delta_1} \min_{\delta_2} \sum_{\theta_1} \sigma(\theta_1) \sum_{a_1} \delta_1(a_1 | \theta_1) \sum_{\theta_2} \sigma(\theta_2 | \theta_1) \sum_{a_2} \delta_2(a_2 | a_1, \theta_2) R(\theta_1, \theta_2, a_1, a_2)$$

$$= \max_{\delta_1} \min_{\delta_2} \sum_{a_1} \tilde{\delta}_1(a_1) \sum_{\theta_1} \sigma_{a_1}(\theta_1) \sum_{\theta_2} \sigma(\theta_2 | \theta_1) \sum_{a_2} \delta_2(a_2 | a_1, \theta_2) R(\theta_1, \theta_2, a_1, a_2).$$
Chapter C. Proofs Repeated Zero-Sum Games Incomplete Information on Both Sides

We know that \( \sigma_{a_1} \in \Delta(\Theta_1) \). This gives us

\[
V_{seq2}(\sigma) = \max_{\delta_1} \sum_{a_1} \bar{\delta}_1(a_1)V^{a_1}(\sigma).
\]

Inserting an additional maximization ensures that the following inequality holds:

\[
V_{seq2}(\sigma) \leq \max_{\delta_1} \sum_{a_1} \bar{\delta}_1(a_1) \max_{a'_1 \in A_1} V^{a'_1}_{seq2}(\sigma_{a_1}).
\]

By definition, \( \sum_{a_1} \bar{\delta}_1(a_1) = 1 \) and \( \sum_{a_1} \bar{\delta}_1(a_1)\sigma_{a_1} = \sigma \). Using the fact that \( V^{a'_1}(\sigma) \) is independent of the policy \( \delta_1 \), the following inequality must also hold

\[
\max_{\delta_1} \sum_{a_1} \bar{\delta}_1(a_1) \max_{a'_1 \in A_1} V^{a'_1}(\sigma_{a_1}, \sigma) \leq \text{Cav} \max_{\Delta(\Theta_1) \times \Delta(\Theta_2)} V^{a_1}(\sigma),
\]

and therefore \( V_{seq2}(\sigma) \leq \text{Cav} \max_{\Delta(\Theta_1) \times \Delta(\Theta_2)} V^{a_1}(\sigma) \).

**Lemma C.2.9.** \( V^{a_2}_{seq2}(\sigma) \) is convex over \( \Delta(\Theta_2) \), \( \forall \sigma \in \Delta(\Theta_1) \times \Delta(\Theta_2) \).

**Proof** By substitution using terms from C.2.7, and by the minimax theorem, the minimization over policies of agent 2 can be replaced by a minimization over actions:

\[
V_{seq2}(\sigma) = \max_{\delta_1} \min_{\delta_2} \sum_{\theta_2} \sigma(\theta_2) \sum_{\theta_1} \sigma(\theta_1 | \theta_2) \sum_{a_1} \bar{\delta}_1(a_1 | \theta_1) \sum_{a_2} \delta_2(a_2 | a_1, \theta_2) R(\theta_1, \theta_2, a_1, a_2)
= \max_{\delta_1} \min_{\delta_2} \sum_{\theta_2} \sigma(\theta_2) \sum_{a_1} \bar{\delta}_1(a_1 | \theta_1) \sum_{\theta_1} \sigma_{a_1}(\theta_1 | \theta_2) \sum_{a_2} \delta_2(a_2 | a_1, \theta_2) R(\theta_1, \theta_2, a_1, a_2)
= \max_{\delta_1} \sum_{\theta_2} \sigma(\theta_2) \sum_{a_1} \bar{\delta}_1(a_1) \min_{a_2} \sum_{\theta_1} \sigma_{a_1}(\theta_1 | \theta_2) R(\theta_1, \theta_2, a_1, a_2). \tag{C.2.8}
\]

Let \( f_{\delta_1}(\sigma) \) be a function defined for all marginalized policies for agent 1, \( \bar{\delta}_1 \), as

\[
f_{\delta_1}(\sigma) = \sum_{\theta_2} \sigma(\theta_2) \sum_{a_1} \bar{\delta}_1(a_1) \min_{a_2} \sum_{\theta_1} \sigma_{a_1}(\theta_1 | \theta_2) R(\theta_1, \theta_2, a_1, a_2).
\]

We only consider \( \sigma \) on every \( \Theta_1 \)-slice, so that the distribution over \( \Theta_1, \sigma_{a_1}(\cdot | \theta_2) \) in the equation, is equal for all \( \sigma \). It follows that for these distributions, \( f_{\delta_1} \) is defined over \( \Delta(\Theta_2) \) alone. We know that this function is linear: the proof of linearity is a straightforward generalization of the proof of Lemma C.1.3, and has its counterpart in the function \( f_{[\sigma_{a_1}, \delta_2]}(\theta_1) \) found in Lemma C.2.6. Substituting \( f_{\delta_1}(\sigma) \) in the definition of the value function given in (C.2.8) gives

\[
V_{seq2}(\sigma) = \max_{\delta_1} \sum_{\theta_2} \sigma(\theta_2) \sum_{a_1} \bar{\delta}_1(a_1) \min_{a_2} \sum_{\theta_1} \sigma_{a_1}(\theta_1 | \theta_2) R(\theta_1, \theta_2, a_1, a_2)
= \max_{\delta_1} f_{\delta_1}(\sigma).
\]

99
This shows that $V_{\text{seq2}}(\sigma)$ is a maximization over linear functions on $\Delta(\Theta_1)$, one for each policy $\delta_1$. Therefore, $V_{\text{seq2}}(\sigma)$ must be convex over $\Delta(\Theta_2)$. \hfill \Box

**Lemma C.2.10.** $V_{\text{seq2}}^{a_1}(\sigma) \leq \text{Vex}_{\Delta(\Theta_2)} \min_{a_2} \sum_{\theta_2} \sigma(\theta_2) \sum_{\theta_1} \sigma(\theta_1 | \theta_2) R(\theta_1, \theta_2, a_1, a_2)$.

**Proof** Note that a minimization over actions is equal to a minimization over pure policies, which form a subset of the total set of possible mixed policies. Therefore, values obtained by minimization over actions must be present in the set of values obtained for all mixed policies:

$$V_{\text{seq2}}^{a_1}(\sigma) = \min_{\delta_2} \sum_{\theta_2} \sigma(\theta_2) \sum_{\theta_1} \sigma(\theta_1 | \theta_2) \sum_{a_2} \delta_2(a_2 | a_1, \theta_2) R(\theta_1, \theta_2, a_1, a_2)$$

$$\leq \min_{a_2} \sum_{\theta_2} \sigma(\theta_2) \sum_{\theta_1} \sigma(\theta_1 | \theta_2) R(\theta_1, \theta_2, a_1, a_2).$$

In the previous Lemma, it was shown that $V_{\text{seq2}}^{a_1}$ is convex over $\Delta(\Theta_2)$. Thus,

$$V_{\text{seq2}}^{a_1}(\sigma) \leq \min_{a_2} \sum_{\theta_2} \sigma(\theta_2) \sum_{\theta_1} \sigma(\theta_1 | \theta_2) R(\theta_1, \theta_2, a_1, a_2),$$

$$V_{\text{seq2}}^{a_1}(\sigma) \text{ is convex over } \Delta(\Theta_2),$$

$$V_{\text{seq2}}^{a_1}(\sigma) \leq \text{Vex}_{\Delta(\Theta_2)} \min_{a_2} \sum_{\theta_2} \sigma(\theta_2) \sum_{\theta_1} \sigma(\theta_1 | \theta_2) R(\theta_1, \theta_2, a_1, a_2).$$

\hfill \Box

**Lemma C.2.11.** $V_{\text{seq2}}^{a_1}(\sigma) \geq \text{Vex}_{\Delta(\Theta_2)} \min_{a_2} \sum_{\theta_2} \sigma(\theta_2) \sum_{\theta_1} \sigma(\theta_1 | \theta_2) R(\theta_1, \theta_2, a_1, a_2) \forall \sigma \in \Delta(\Theta_1)$

**Proof** The proof is a straightforward extension of the proof of Lemma C.1.5. \hfill \Box
Bibliography


